Ejercicios para Resolver y Algunos Ejemplos

Programación II - Parte I

(Trabajos Prácticos que el alumno resuelve, incluye en la carpeta y/o explica en clase)

Índice

Entorno de Trabajo

En los laboratorios del colegio trabajaremos con **Python 3.7.9 64-bit** (mínimo) y generalmente usaremos **Spyder IDE 5.2.2** (mínimo).

Todos los ejemplos aquí explicados y desarrollados, fueron confeccionados, probados y ejecutados con el mencionado entorno. Tú puedes usar cualquier otro entorno que prefieras, aunque no se puede asegurar que funcionen exactamente igual, debido a la diferencia entre versiones y/o hardware.

Lee atentamente, encontrarás las pautas de trabajo y algunas cosas que te resultarán útiles a lo largo del año. Aprovéchalas.

Pautas de Trabajo

A continuación, un listado de enunciados (Trabajos Prácticos), que deberás resolver, agregar en tu carpeta y organizarlos de la siguiente forma:

a) Analiza y construye el diagrama de flujo.

b) Testea tu lógica, realizando la Prueba de Escritorio del diagrama de flujo. Recuerda que en este punto, es donde se encuentran los problemas de lógica que tu algoritmo presente, y de presentarse, corrige el diagrama para que funcione apropiadamente.

c) Una vez terminado el diagrama y Prueba de escritorio, escribe el código (programa) correspondiente en una PC (según tu diagrama de flujo), verifica el correcto funcionamiento (testing) e **imprímelo** para agregar el código en tu carpeta (funcionando bien e idéntico a tu Diagrama). Incluye número de ejercicio y enunciado en la parte superior de tu código (como un comentario), esto permitirá identificar de inmediato que programa es y que se pidió.

d) Finalmente en tu carpeta, a continuación del enunciado y cuando se pida, el Prompt (Tu comando o pedido de una tarea a tu I.A. preferida), debe estar el diagrama de flujo, prueba de escritorio y para finalizar, el segmento de hoja donde imprimiste el código. Recorta y pégalo a continuación o al costado del lugar donde realizaste el diagrama de flujo y prueba de escritorio.

> *Debes tener presente que*, el diagrama de flujo, prueba de escritorio y código, deben corresponderse. Si hubiera **alguna diferencia** entre ellos, **el ejercicio será rechazado** (se considera que está mal resuelto).
> Aprovecha que muchos de los diagramas y sus pruebas de escritorio, son resueltos durante las clases teóricas; y adicionalmente, dispones de una o dos clases semanales (clases prácticas) para realizar la programación (código) en una de las PC del laboratorio en el colegio.

e) **Resguarda** siempre **tus trabajos** en un **Pendrive** y en caso de perdida, evitarás hacer nuevamente todos los ejercicios para la evaluación y corrección de tu Carpeta.

f) La corrección del diagrama y prueba de escritorio será evaluada/corregida en tu carpeta, simultáneamente el código y Prompt, en una PC, donde deberás explicar el funcionamiento de tu programa, pudiendo tener que realizar algún cambio menor del código (como una evaluación en la PC).

g) La corrección, podrá realizarse dos semanas después (14 días) de la fecha de finalización indicada en cada ejercicio o tema.

h) **MUY IMPORTANTE**: Sugiero que resuelvas TODOS los ejercicios tu solo/a, sin embargo, puedes usar alguna IA (Inteligencia Artificial) que te ayude, así aseguras que TODOS los programas funcionan bien; te recomiendo guardar tus Prompt, serán necesarios. Igualmente tú deberás saber resolverlos, modificarlos y explicarlos (durante la corrección de tu carpeta, de los códigos contenidos en el Pendrive y en las evaluaciones). Recuerda que las evaluaciones serán en el aula (No podrás consultar: La carpeta, apuntes, celular ni la PC), donde tendrás que dibujar los diagramas de flujo, realizar las pruebas de escritorio, escribir la codificación Python y/o generar tus prompt en una hoja de papel.

Nro	Cosas Interesantes (Parte I)	Finaliza
	Primero es lo primero, comenzamos el año y desde ahora podemos decir que te va a gustar.. Escuchemos el Inicio..... Ingresar: Inicia.htm	
	Configurar Spyder: Puedes Configurar algunos detalles que te ayudarán cuando estés programando. Recomendado: Descarga Documento acá: Configurar_Spyder_para_Archivos_txt.pdf	
	Separador de código: Puedes marcar uno o varios segmentos en tu código para que visualmente lo identifiques: Coloca **"#%%"** pero sin las comillas, donde quieras separar tu código Esta secuencia de caracteres, trazará literalmente una línea que no interfiere con tu lógica o al ejecutarlo. Baja el ejemplo de la nube, analízalo y pruébalo. Es muy práctico. Baja el ejemplo desde acá: Codigo/90_Interesante_001_Raya_Separadora_del_Código.txt Pruébalo en tu **Spyder IDE**.	
	Comentario Resaltado en tu Código: Otra forma de insertar rápidamente un comentario y resaltarlo en tú código es: luego de escribir el comentario, simplemente presionas las dos teclas **"Control4"**, o marcas con el Mouse los renglones que quieres que sean parte del comentario y entonces presionas las dos teclas **"Control4"** ☺. Y.... **"Control5"** elimina el comentario ☹. *Este es el Comentario* **Control4** Así de simple, dos teclas y habrás insertado en tu código un comentario resaltado, Siempre será visto**.** # ========== # Este es el Comentario # ========== Baja el ejemplo desde la nube, analízalo y pruébalo. Es muy práctico. Baja el ejemplo: Codigo/90_Interesante_001_Comentario_Resaltado_en_el_Codigo.txt	
	Atajos o Códigos Genéricos en TU Entorno de Trabajo: Verifica en donde programas, cuales funcionan y cuales no. • **Ctrl + F:** Buscar texto en el archivo actual. • **Ctrl + H:** Buscar y reemplazar. • **Ctrl + Tab:** Cambiar entre pestañas abiertas. • **F11:** Activa/Desactiva Pantalla Completa (distrae menos al programar). • **Ctrl + Insert / Shift + Insert:** Copia desde cualquier aplicación y pega en cualquier aplicación (texto, imagen, etc) lo que necesites (**Shift**, en algunos teclados se visualiza con una fecha hacia arriba). • **Ctrl + Z / Ctrl + Y:** Deshacer y rehacer Ultimas Acciones realizadas en el código. Existen muchos otros atajos, que te facilitarán la tarea de escribir tu código. Busca y apréndelos.	

BREVE REPASO

Recordemos lo que es una Variable: Una variable en programación, es un espacio de memoria que se utiliza para almacenar y representar información. En términos más simples, es como una caja donde puedes guardar diferentes valores (números, texto, Valores lógicos, etc). Las variables son fundamentales en la programación, ya que te permiten manipular y trabajar con datos de forma dinámica. Por ejemplo, si quieres almacenar la edad de una persona, puedes crear una variable llamada "edad" y asignarle un valor numérico.

Una variable en programación debe seguir ciertas reglas al momento de ser nombrada. Estas reglas son:

a. El nombre de la variable debe comenzar con una letra o un guión bajo "_".
b. El nombre de la variable puede contener letras, números y guiones bajos.
c. No se pueden utilizar espacios en el nombre de la variable.

El nombre de la variable debe ser único y descriptivo.

Diagramas de Flujo (Parte 01): Los diagramas de flujo son representaciones gráficas de un algoritmo o proceso, de la secuencia de acciones que imaginamos. En programación estructurada (como en Python, C, etc.), se utilizan para visualizar la secuencia de pasos, decisiones y bucles que componen un programa. Se usan símbolos estándar para representar diferentes acciones a codificar: rectángulos para procesos, rombos para decisiones (condiciones), flechas para indicar el flujo de ejecución, etc. Su propósito principal es facilitar la comprensión, diseño y depuración de programas, haciendo más fácil la planificación antes de escribir el código. Permiten una representación visual clara y concisa del flujo lógico del programa, lo que ayuda a identificar errores o áreas de mejora antes de la implementación. Lon símbolos que usaremos son:

			
En Python, solo se usa en los diagramas de flujo. Con este símbolo indicaremos el inicio y final del programa.	Imprimir un dato print("Resultado: ", A)	Ejecutar una acción R = A + B	Lectura de un dato A = input("Nombre: ")

Nro	Ejercicios y Trabajos para la Carpeta - Repaso	Finaliza
1)	Investiga, amplía, explica y confecciona un pequeño ejemplo (Un mini programa para cada ejemplo, o un programa grande en el que explicas cada ejemplo) - Recuerda que cada Programa acompaña a su diagrama de flujo, y sin importar que parámetro se use, el símbolo del diagrama siempre será el mismo print( ) — Realiza un resumen de TODOS los parámetros que esta función puede usar, explicando en detalle que hace cada uno, junto con un pequeño programa ejemplo para visualizar el resultado. Puedes poner todos los ejemplos en un solo programa si los diferencias con un breve mensaje descriptivo. Por ejemplo, Que pasa aquí y que visualizaremos en la pantalla? — `print('Hola', end = ' ')` `print('mundo')` input( ) — Realiza un resumen de TODOS los parámetros que esta función puede usar, explicando en detalle que hace cada uno, junto con un pequeño programa ejemplo para visualizar el resultado. Puedes poner todos los ejemplos en un solo programa si los diferencias con un breve mensaje descriptivo.	10/03
2)	Hacer un programa (Diagrama de flujo y código Python), que pregunte el nombre del usuario en la consola y después que el usuario lo introduzca, muestre por pantalla la cadena **¡Hola <nombre>!**, donde <nombre> es el nombre que el usuario haya introducido.	10/03
3)	Operaciones Aritméticas Que Debes Repasar: Desde la Nube puedes descargar una breve reseña de las operaciones aritméticas que usaremos este año. El documento contiene una rápida descripción y un ejemplo de cada una (diagrama de flujo y código). Descargar desde: Operadores Aritmeticos Python 01.pdf	10/03 

Analiza, y ejecuta los códigos en la PC. Asegúrate de comprender que hace y como lo hace.

Fecha de finalización 17/03

Cuando Leemos Números: En Python, cuando leemos datos, que ingresados desde teclado usando input(), estos siempre se reciben como texto o "strings", sin importar si el usuario escribe números o letras. Esto significa que, cuando queramos leer o ingresar números, aunque el usuario escriba números, al ingresarlos, Python los entenderá como caracteres o cadenas de caracteres.

Para poder leer los datos y luego hacer operaciones matemáticas (sumar, restar, etc.) o comparaciones numéricas, siempre necesitaremos transformar estos datos al tipo adecuado, como "**int**" para números enteros, "**float**" para números decimales, etc.

Veamos algunos ejemplos de código:

Lectura y transformación de un numero Entero o int	

Ejemplo 01
Lee el dato como texto
	dato = input("Ingresa un número entero: ")
	Numero = int(dato) # Transforma el dato a entero
	print(f"El número ingresado es: {Numero}")

Lectura y transformación de un numero con Parte decimal o float	

Ejemplo 02
Lee el dato como texto
	dato = input("Ingresa un número decimal: ")
	Numero = float(dato) # Transforma el dato a decimal
	print(f"El número ingresado es: {Numero}")

Hay que mencionar que, los usuarios pueden cometer errores al escribir los datos en el teclado y generar errores en el programa, que en ese caso, Python detendría el programa. Este problema puede ser prevenido y corregido si controlamos que los datos ingresados tengan el formato debido, y le indicamos a Python que debe hacer en caso de que se produzca el error.

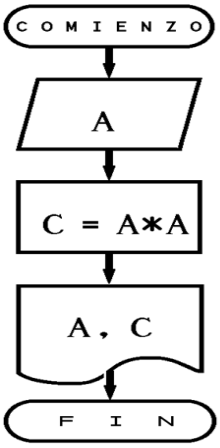
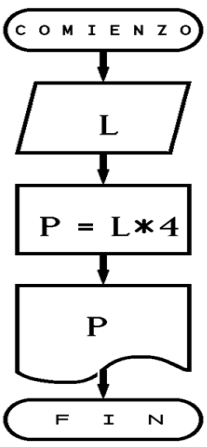
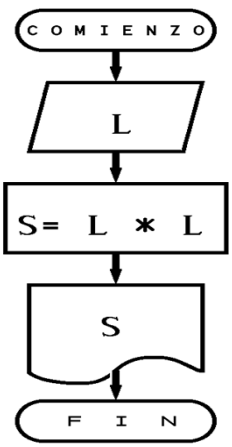
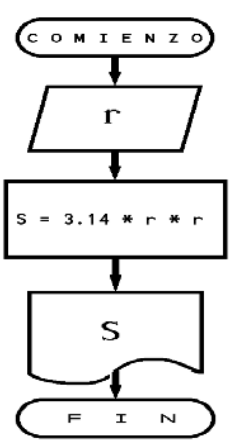
Lectura, Control y transformación de un número Entero o int	

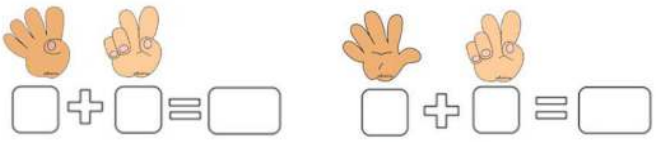
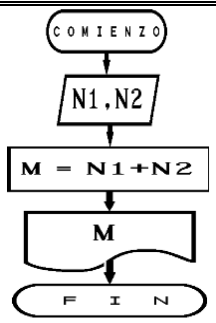
Ejemplo 03	dato = input("Ingresa un número Entero: ") **# Lee Cadena**
	try:
	  Numero = int(dato) **# Convierte a entero**
	  print("El número leído es:", Numero)
	except ValueError as e**:**
	  print("Error: número ingresado NO es int")
	  print(f'Error: {e}")

Lectura, Control y transformación de un número float	

Ejemplo 04	dato = input("Ingresa un número Real: ") **# Lee Cadena**
	try:
	  Numero = float(dato) **# Convierte a decimal**
	  print("El número leído es:", Numero)
	except ValueError as e**:**
	  print("Error: número ingresado NO es float")
	  print(f'Error: {e}")

4) Analiza los diagramas de flujo y para cada uno, escribe un enunciado válido, codificarlo y probarlo en una PC. 17/03

Diagrama A	Diagrama B	Diagrama C	Diagrama D
			

5)	Dado el diagrama de Flujo, analiza y crea el enunciado correspondiente. Para finalizar codifícalo. Pega el enunciado, diagramas y código en tu carpeta. Dejo pistas para la prueba de escritorio y creación del enunciado  . 		17/03

Analiza, y ejecuta los códigos en la PC. Asegúrate de comprender que hace y como lo hace.

Fecha de finalización 24/03

Conversión de Tipos, "type castin" o simplemente "casting".

El proceso de transformar datos a un tipo determinado en Python se denomina "conversión de tipos" o "type casting". Y existen dos tipos de conversiones: La implícita o automática y la conversión explícita.

Tipos de conversión

Conversión implícita: Es cuando Python automáticamente transforma un dato de un tipo a otro, por ejemplo, al sumar un entero con un número decimal:

```
Entero = 5
Decimal = 2.5
Resultado = Entero + Decimal  # Convierte automáticamente `entero` a `float`
print(type(Resultado))  # Resultado: <class 'float'>
```

Conversión explícita: Es cuando el programador utiliza funciones específicas para transformar un dato de un tipo a otro:

```
texto = "42"
numero = int(texto)  # Convierte de string a entero
print(type(numero))  # Resultado: <class 'int'>
```

Funciones comunes de conversión explícita en Python.

Python dispone de muchas funciones para la transformación de datos. A continuación te muestro las más comunes:

```
int( ): Convierte a entero.
float( ): Convierte a número decimal.
str( ): Convierte a cadena de texto.
bool( ): Convierte a valor booleano.
```

Analiza, y ejecuta los códigos en la PC. Asegúrate de comprender que hace y como lo hace.

Fecha de finalización 24/03

Mostramos o Imprimimos Datos (Números, Cadenas, Etc).

Independientemente de como hayamos leído o transformados los números, cuando los mostramos o imprimimos en el monitor, es el momento en que aportamos claridad para que se comprendan los resultados. Podemos darle una forma agradable a la vista, sin cambiar su verdadero formato.

Números Enteros (int): Hay varias cosas que podremos hacer, y más que decir, es mostrar, así que veamos directamente en el código.

```
num = 42
print(f"'{num:5d}'\n")  # Salida: " 42" (3 espacios antes del número)
print(f"'{num:05d}'\n") # Salida: 00042
print(f"'{num:<5d}'\n")  # Salida: "42 " # Alineación Izquierda con espacios
print(f"'{num:^5d}'\n")  # Salida: " 42 " Centrar el número en un campo de ancho fijo
# Separador de Miles
num = 1000000
print(f"'{num:,}'\n")  # '1,000,000' (con coma)
```

```
print(f"'{num:_}'\n")  # '1_000_000' (con guión bajo)
print(f"'{num:,}'\n".replace(",", "."))  # Salida: '1.000.000' (con Punto)

# Signo Explícito
num_pos = 42
num_neg = -42
print(f"'{num_pos:+d}'\n")  # '+42' (siempre muestra el signo)
print(f"'{num_pos:+5d}'\n")  # '+42' (siempre muestra el signo)

print(f"'{num_neg:+d}'\n")  # '-42'
```

Descarga de la nube programa completo:	Codigo/14_Formateo_de_Cadenas_010_Impresion_01a_int_01.txt

Números Reales (float): Con este tipo de números hay varios trucos, algunos simples y otros avanzados. Acá te muestro todos, luego, cuando transcurra el año, siempre podrás regresar y repasarlos

```
print("Ejemplo 01: redondea número float a cantidad de decimales elegida")
Num = 3.1415926535
Valor = round(Num, 2)
print(f'Numero Redondeado con 2 decimales: '{Valor}'\n")  # Imprime: 3.14

print("Ejemplo 02:")
# Imprime: 3.14 - permite especificar cantidad de decimales
print(f'Numero Formateado con 2 decimales: '{Num:.2f}'\n")

print("Ejemplo 03:")
# Espesifica cantidad total de dígitos incluido el punto, y cantidad de decimales,
Num = 12.3456
print(f"'{Num:6.2f}'\n")   # Imprime: 12.35

print("Ejemplo 04:")
# completando con ceros a la izquierda
print(f"'{Num:06.2f}'\n")   # Imprime: 012.35

print("Ejemplo 05:")
# SIEMPRE SE MUESTRA EL SIGNO
print(f"'{Num:+.2f}'\n") # Imprime: +3.14 (siempre muestra el signo pos o neg)

print("Ejemplo 06:")
# Justifica el numero a la izquierda, ocupando la cantidad de digitos señalada
print(f"'{Num:<10.2f}'fin'\n") # Imprime: '3.14 fin' (se llena con espacios a la derecha)
#============================================================
print("Ejemplo 07a:")
Num = 1234567.8914
# Uso de la coma decimal y punto para miles
def  NumArgentino_01( Numero):
    Valor = f"{ Numero:,.2f}".replace(",", "X").replace(".", ",").replace("X", ".")
    return Valor

NA = NumArgentino_01(Num)
print(f"'{NA}'\n") # Imprime: 1.234.567,89
#============================================================
print("Ejemplo 07b:")
def NumArgentino_02(Numero, decimales):
    formato = f"{{:,.{decimales}f}}"  # Construcción dinámica del formato
    Valor = formato.format(Numero).replace(",", "X").replace(".", ",").replace("X", ".")
    return Valor

NA = NumArgentino_02(Num,2)
print(f"'{NA}'\n") # Imprime: 1.234.567,89

#============================================================
# Mira que pasa si le digo que quiero cero decimales
NA = NumArgentino_02(Num,0)
print(f"'{NA}'\n") # Imprime: 1.234.567,89
```

Descarga de la nube programa completo:	Codigo/14_Formateo_de_Cadenas_010_Impresion_01a_float_01.txt

Cadenas de Caracteres (Texto): Con las cadenas hoy muchas funciones y/o métodos, aca te dejo las que mas usaremos en clase. Descarga el programa ejemplo y analízalo:

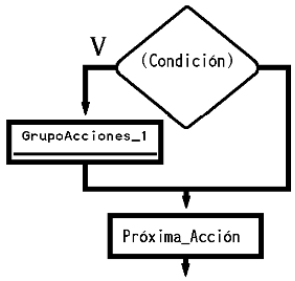
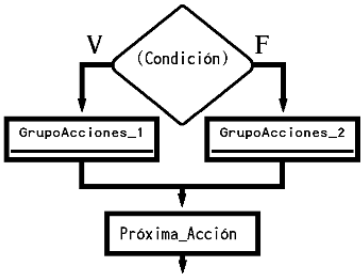
Descarga de la nube programa completo:	Codigo/14_Formateo_de_Cadenas_010_Impresion_01a_Cadenas_01.txt

Los otros tipos de datos los iremos viendo en el camino.

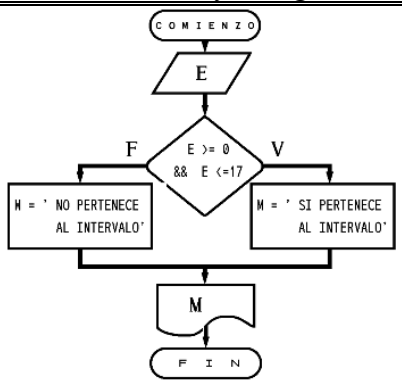
bool( ): Booleans.
Y un monton de otros que acá no mencionaré

Diagramas de Flujo (Parte 02): Uso del "if" y del "while"

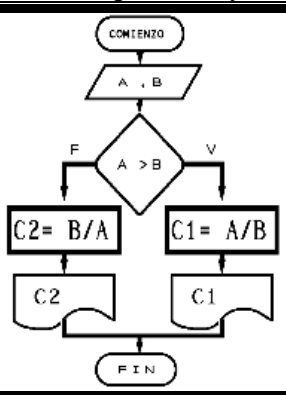
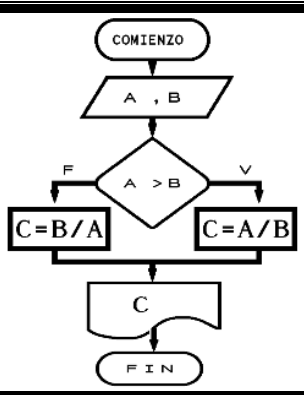
Importante: Siempre recuerda que, un diagrama de flujo describe la lógica *pura* que debes desarrollar en tu algoritmo, representando los pasos y decisiones de manera abstracta. Por lo tanto, **no se enfoca en las particularidades sintácticas de los distintos lenguajes de programación**, tampoco en las formas específicas en que se definen o declaran las variables, ni los tipos de datos o las estructuras de control propias de cada lenguaje. Esto permite que cada programador, al implementar el algoritmo en un lenguaje específico (Python, Java, C++, etc.), defina y/o declare las variables de la manera más apropiada y eficiente según las convenciones y características de dicho lenguaje. El diagrama de flujo es la hoja de ruta lógica, y la implementación en código es la traducción a un lenguaje concreto.

		
Condición Simple	Condición Doble o Compuesta	Repetición (Mientras)

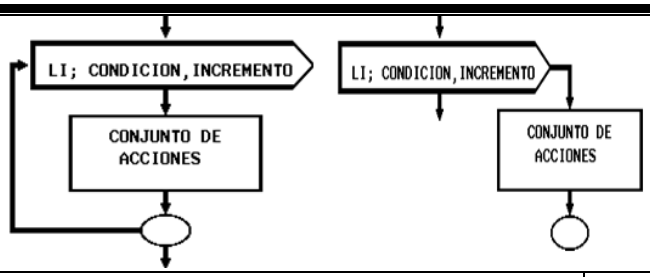
if condición:
GrupoAcciones_01
Proxima_Accion
	if condición:
	GrupoAcciones_01
	else:
	GrupoAcciones_02
		while condición:
		GrupoAcciones
		Primera_Accion_Fuera_Del_Ciclo

Nro	Ejercicios Simples, primero en el pizarrón y luego en la PC		Finaliza
6)	Leer dos números, calcular y mostrar la suma. Recuerda, esto ya lo vimos.		24/03
7)	Luego de leer dos números, calcular y mostrar el resultado de la división entre el número mayor y el número menor. Hacer diagrama de flujo, prueba de escritorio y código.		24/03
8)	Analiza el Diagrama, Escribe el enunciado y Codifícalo. Pega los diagramas en tu carpeta. Acá lo importante es el uso de una variable del tipo cadena de caracteres o string. 		24/03
9)	Hacer un programa (Diagrama de Flujo, Prueba de escritorio y codificación) que me permita leer 3 números, y luego buscar y mostrar el mayor de los tres.		31/03
10)	Leer un número e informar si es divisible por dos y por tres al mismo tiempo		31/03

Si por alguna razón, una clase destinada a la corrección de carpetas no se pudiera concretar, la corrección y/o trabajos de quienes quedaran pendientes, será desplazada a la case inmediata posterior a la planificada.

Nro	Ejercicios Simples, primero en el pizarrón y luego en la PC			Finaliza
11)	Analiza ambos Diagramas y explica si hacen cosas similares o no. Escribe el enunciado correspondiente y Codifícalos (Pega los diagramas del ejercicio en tu carpeta). Dividendo | Divisor Cociente Resto			31/03
12)	**Este ejercicio tiene varias partes. Resuélvelas a todas**: **a)-** Leer 2 (dos) números enteros por teclado, y luego informar por consola (el monitor) si estos números son iguales o distintos. **b)-** Leer 3 (tres) números enteros por teclado, y luego informar por consola (el monitor) si estos números son iguales o distintos. **c)-** Leer 3 números y luego mostrarlos en orden ascendente (de menor a mayor) **d)-** Analiza los diagramas y explícalos, escribe los enunciados y Codifícalos en tu PC, recuerda incluirlo en tu carpeta. Descarga los diagramas de la nube: Diagramas/O_Analisis_01.png; Diagramas/O_Analisis_02.png; Diagramas/O_Analisis_03.png			31/03
13)	Hacer un programa que calcule el importe que se debe pagar por una compra. El programa solicitará al cajero que ingrese el Precio Unitario del producto, la cantidad de unidades que el cliente está comprando y el IVA asociado a ese tipo de producto **1:** Exento (cero%), **2:** 10,5% o **3:** 21%. Calcular y mostrar en el monitor el importe que deberá pagar el cliente.	Importe = PU*Cant IVA = Importe * IVA Total = Importe + IVA		31/03

Diagramas de Flujo (Parte 03): Ciclo "For"



Ciclo repetitivo for

(Investiga que tiene muchas otras opciones además de lo que te muestro)

Tú Código	Y en el monitor veas:
for i in range(5): print(i)	0 1 2 3 4

Opciones de Uso y Descripción	Código	En el Monitor
a)- Bucle "**for**" recorriendo una secuencia numérica. Para estos casos, Python pone a nuestra disposición "**range**", donde:		
➢ range(max): números enteros consecutivos que empiezan en 0 y terminan en "max - 1" (el anterior al último).	for i in range(5): print(i)	0 1 2 3 4
➢ range(min, max): para usar con números enteros consecutivos que empiezan en "min" y terminan en "max - 1" (el numero anterior al último).	for num in range(2, 9, 2): print(num)	2 4 6 8
➢ range(min, max, step): números enteros consecutivos que empiezan en "min", terminan en "max - 1" (el anterior al último) y los valores se van incrementando de "step" en "step".	for num in range(0, 11, 2): print(num)	0 2 4 6 8 10

b)- Recorriendo una lista: Material de consulta.

```
Elementos = [4, 78, 9, 84]
c=0
for n in Elementos:
    print(c, " => ", n)
```

```
4
78
9
84
```

c)- Recorrer los elementos de un **diccionario**. Dado que un diccionario está compuesto por pares clave/valor, hay distintas formas de iterar sobre ellas.

Acá Puedes Ampliar tus Conocimientos en Diccionarios Python

Diccionarios_Creacion_y_Uso_Parte_01.pdf
Diccionarios_Creacion_y_Uso_Parte_02.pdf

```
Elementos = {'A': 4, 'E': 3, 'I': 1, 'O': 0}
for k, v in Elementos.items():
    print('k = ', k, ', v = ', v)
```

```
k=A, v=4
k=E, v=3
k=I,  v=1
k=O, v=0
```

```
Elementos = {'A': 4, 'E': 3, 'I': 1, 'O': 0}
for k in Elementos:
    print(k)
```

```
A
E
I
O
```

```
Elementos = {'A': 4, 'E': 3, 'I': 1, 'O': 0}
for v in Elementos.values():
    print(v)
```

```
4
3
1
0
```

d)- En ocasiones, puedes no usar variable para iterar. En estos casos se usa el caractér "_" (guión bajo) para indicar esta situación. Sin embargo, no lo recomiendo, ya que puede generar errores y conflictos con algunas librerías usadas en tu programas.

```
Elementos = [5, 78, 9, 84]
cont = 0
for _ in Elementos:
    cont += 1
print("Se contaron ", cont, "elementos")
```

```
Se contaron 4 elementos
```

e)- También es posible alterar la iteración de un bucle **for** en Python. Para esto se pueden usar las sentencias "**break**" y "**continue**". Pero, es importante destacar que recomiendo **NO USARLAS**, ya que rompen las estructuras de control, pudiendo introducir errores en el código muy difíciles de encontrar corregir:

- **break**: se utiliza para finalizar y salir el bucle, por ejemplo, si se cumple alguna condición.
- **continue**: salta al siguiente paso de la iteración, ignorando todas las sentencias que le siguen y que forman parte del bucle.

```
Elementos = [2, 4, 5, 6, 7, 8, 9, 3, 4]
for n in Elementos:
    if n == 7:
        break
    else: print(n)
print("Encontré el elemento ",n)
```

```
Prueba este ejemplo tal como esta, y luego elimina el 7 de los Elementos y prueba nuevamente
```

```
Elementos = [1, 2, 3, 4, 5, 8, 6]
for n in Elementos:
    if n == 3:
        break
    print(n)
else:
    print('No se encontró el número 3')
```

```
Prueba este ejemplo tal como esta, y luego elimina el 3 de los Elementos y prueba nuevamente
```

El ciclo "**for**", también te permite usar el bloque "**else**" que solamente se ejecutará, siempre y cuando no se haya ejecutado la sentencia "**break**" dentro del bloque del "**for**".

Nro	Enunciados - Ciclos For y While	Finaliza
14)	Analiza los Diagramas, escribe los enunciados y Codifícalos. Pega todos los diagramas y códigos en tu carpeta. Descarga los diagramas desde la nube. Diagrama 1: Diagramas/Par_Cociente_A.png Diagrama 2: Diagramas/While_50.png Diagrama 3: Diagramas/Maximo_50.png Diagrama 4: Diagramas/May_Par.png Diagrama 5: Diagramas/May_Que.png	07/04
15)	Este ejercicio tiene dos partes, resolver cada una como un ejercicio separado. Leer por teclado una serie de 10 números, calcular y mostrar la suma. **(No Usar Listas)**. a) Resolver con **While** b) Resolver con **for** **Recuerda la Sintaxis** While Condición: for i in range(máximo):	07/04
16)	Leer por teclado una serie de números. Se pide calcular e imprimir la suma de todos los elementos, la lectura de números termina cuando se lea/ingrese un valor menor que 1. **(No Usar Listas)**.	07/04

<table>
<tr><th>Nro</th><th>Enunciados - Ciclos For y While</th><th>Finaliza</th></tr>
<tr><td>17)</td><td>Hacer un algoritmo que permita al usuario ingresar una serie de valores, y calcule el promedio de todos los números ingresados. El programa termina cuando se lea/ingrese un valor menor que cero. (No se puede usar listas). La sumatoria de los números debe realizarse mientras el programa los va leyendo).</td><td>07/04</td></tr>
<tr><td>18)</td><td>Analiza los Diagramas, escribe los enunciados y pega los códigos junto con los diagramas en tu carpeta. Descarga y usa los diagramas de la nube.
<table>
<tr><td>Diagrama 1:</td><td>Diagramas/Mayores_que.png</td><td>Diagrama 2:</td><td>Diagramas/50_Multi.png</td></tr>
<tr><td>Diagrama 3:</td><td>Diagramas/Suma_50.png</td><td>Diagrama 4:</td><td>Diagramas/Suma_Pares.png</td></tr>
</table></td><td>14/04</td></tr>
<tr><td>19)</td><td>Leer por teclado una serie de números. Se pide calcular e imprimir la suma de todos los elementos menores que 15 y el promedio de los restantes. La serie termina cuando se lea un valor menor que 1. (No Usar Listas).</td><td>14/04</td></tr>
<tr><td>20)</td><td>Leer por teclado una serie de pares ordenados con el formato (Código, Valor). Se pide calcular el promedio de todos los valores, cuyos Códigos asociados sean pares. Se identifica el final de la serie de pares, cuando algún código sea igual a 0 (cero). Realizar el código y prueba de escritorio.
<table>
<tr><td>Diagrama:</td><td>Diagramas/Promedio_Codigos_Pares.png</td></tr>
</table></td><td>14/04</td></tr>
<tr><td>21)</td><td>Este ejercicio tiene dos partes, resolver cada una como un ejercicio separado. Buscar el mayor valor contenido en una serie de 10 números leídos. (No Usar Listas).
a) Resolver con While
b) Resolver con for
<table>
<tr><th>Sintaxis</th></tr>
<tr><td>While Condición:</td></tr>
<tr><td>for i in range(máximo):</td></tr>
</table></td><td>14/04</td></tr>
<tr><td>22)</td><td>Mostrar el Mayor valor encontrado en una serie de números enteros, que se leerán por el teclado. La lectura de números debe terminar cuando se encuentre algún valor mayor que 100. A continuación puedes descargar desde la nube, una forma (hay muchas) en que puedes resolver este ejercicio. Codifícalo y agrégalo a tu carpeta (diagrama, prueba de escritorio y código)
<table>
<tr><td>Diagrama:</td><td>Diagramas/Mayor_Serie_Numeros_02.png</td></tr>
</table></td><td>21/04</td></tr>
<tr><td>23)</td><td>Buscar y mostrar el menor valor encontrado en una serie de números ingresados por el usuario. El ingreso de números debe terminar cuando se lea un número menor que 1. (No Usar Listas).</td><td>21/04</td></tr>
<tr><td>24)</td><td>Para cada uno de los códigos, realiza el diagrama de flujo correspondiente y confecciona un enunciado apropiado.
<table>
<tr><th>Código A</th><th>Código B</th></tr>
<tr><td><pre>
Suma = 0

for i in range(10):
    Cadena = "Ingresar el " + str(i+1) + " Valor: "
    Num = int(input(Cadena))
    Suma = Suma + Num

print("Las suma de los 10 numeros es: ", Suma)
</pre></td><td><pre>
Val = int(input("Ingrese un numero: "))

Calculo = Val # Inicializo variable para calculo
for i in range(1, Val):
    Calculo = Calculo * i # Calculo Factorial

# Muestro numero y factorial
print("Número: ", Val, " - Factorial: ", Calculo)
</pre></td></tr>
<tr><th>Código C</th><th>Código D</th></tr>
<tr><td><pre>
# Ingreso el numero
Nro = int(input("Ingrese un numero: "))
if Nro > 1:
    Primo = True
    for i in range(2, Nro):
        if Nro % i == 0 and i < Nro :
            Primo = False
    if Primo == True:
        print("El Número: ", Nro, "SI Es Primo")
    else:
        print("El Número: ", Nro, "NO Es Primo")
else:
    print("El numero ingresado debe se mayor que 1")
</pre></td><td><pre>
print("Terminar proceso con 0 o Numero negativo:")
numero = int(input("Núm: "))

while numero > 0 :
    Suma = 0
    for i in range(1,numero+1):
        if numero % i == 0 :
            Suma = Suma + i
    print("\nMuestro resultados y continuo: ")
    print("--")
    print("Suma de los divisores del número es:", Suma, "\n")
    print("--")
    print("Terminar proceso con 0 o Numero negativo:")
    numero = int(input("Núm: "))
</pre></td></tr>
</table></td><td>21/04</td></tr>
</table>

Reconocer los Tipos de Datos

Esto seguro que te servirá para mucho de lo que en este año tendremos que hacer. Analízalo!

Este código te permitirá identificar, el dato leído a que tipo de datos perteneces o el número que estás usando, que tipo es, o si es una cadena, etc.

Recuerda que cuando lees desde el teclado, siempre lees una cadena de caracteres, aunque esos caracteres puedan ser dígitos o letras, pero cuando trabajas con variables y especialmente cuando recibes parámetros, podrás reconocer cada tipo de datos. Analiza los ejemplos, hay mucho por descubrir.

```
numero = 123.456
print (numero)
print (type(numero))
if isinstance(numero, complex):
    print (numero, ' es un número complejo')
elif isinstance(numero, float):
    print (numero, ' es un número es real')
elif isinstance(numero, int):
    print (numero, ' es un número es entero')
elif isinstance(numero,str):
    print (numero, ' es una cadena')
# etc...
else:
    print(numero,' es un completo misterio')
```

Recuerda incluirlo en la carpeta :)

Código 1:	Codigo/90_Interesante_001_Reconoce_Caracter_a_que_tipo_Pertenece.txt

Código 2:	Codigo/90_Interesante_001_Reconociendo_Tipos_de_Numeros.txt

Caracteres Unicode: También puedes reconocer y clasificar los caracteres en tu programa, mediante las categorías de caracteres Unicode, una clasificación muy usada y extremadamente útil (Por ejemplo **los emojis** entre otros). Puedes ampliar descargando desde la nube el siguiente documento:

Características Caracteres Unicote:	Caracteres_Unicode_Caracteristicas.pdf

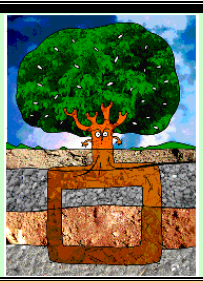
Programa Python para reconocer y clasificar los caracteres, descarga de la nube y analiza.

Código 3:	Codigo/90_Interesante_001_Reconociendo_Informacion_del_Caracter.txt

21/04

Nro	Enunciados donde se Reconocen Tipos de Datos	Finaliza
25)	Hacer un programa en el que se lean tres números (x,y,z) y a continuación, según sea el valor del primero, se calcule y muestre: - Si x= 1 Calcular la suma de los otros dos R = Y + Z - Si x= 2 Calcular el productos de los dos restantes R= Y*Z - Si x = 3 Calcular La raíz Cuadrada de la suma de los cuadrados de los dos números restantes: $$R = \sqrt{Y^2+Z^2}$$ Importante: Primero asegúrate de que "x" es un número, y además, si x tuviera cualquier otro valor, informar que se trata de un error y leer nuevamente ese valor de "x", luego continuar con la lectura de los otros dos valores.	28/04
26)	Este ejercicio tiene dos partes, resolver cada una como un ejercicio separado: Escribir un programa que solicite al usuario ingresar un número "**n**", que deberá ser entero y positivo. Verificar que "n" cumpla con las condiciones, y en caso de error, avisar del problema y leer nuevamente el número. $R = \frac{n(n+1)}{2}$ **Luego:** **a)-** Resolver la ecuación dada y mostrar el resultado por el monitor. **b)-** Calcular y mostrar la sumatoria de los resultados de la ecuación para todos los valores de "n" consecutivos y enteros, comprendidos entre 0 (cero) y 20 (veinte).	28/04
27)	Leer por teclado una serie de Números Positivos. Verificar que los números leídos cumplan con la condición, los números que no cumplan con la condición, deberán ser descartados, avisando el problema y leyendo otro número en su lugar. Se pide calcular imprimir la suma de todos los números validos (la serie termina cuando se ingrese un cero y en este caso no se considera como error). Recuerda que No puedes Usar Listas. Los números positivos son aquellos que son mayores que cero, mientras que los números negativos son aquellos que son menores que cero. Y el cero que es?	28/04 
28)	Desarrollar un programa en el que se lean por teclado una serie de números y mientras se ingresan, se calcule la suma de los números que sean múltiplos de 2, busque el mayor de los múltiplos de 5 y el promedio de los múltiplos de 3. El programa termina cuando se ingresa un número menor que 1 (uno), y recién al terminar la lectura, mostrar los resultados del proceso. Verificar que los datos leídos sean realmente números y en caso de ingresar caracteres, informar del error, ignorar el dato y leer nuevamente. Recuerda que todavía No puedes Usar Listas.	28/04

Nro	Enunciados Variados - De Todo un Poco.	Finaliza
29)	El programa debe ser capaz de leer una serie de números enteros positivos (verificar que los números cumplan con la condición). **a)** El ingreso de los datos termina cuando se lea un número menor que 1. **(No Usar Listas)**. **b)** Mientras se ingresan los números, debes seleccionar el mayor y menor valor de los valores que sean múltiplos de 2. **c)** Al finalizar la carga de números y antes de terminar la ejecución del programa, calcular y mostrar el promedio todos los números que sean múltiplos de 3 y la suma los múltiplos de 5. **d)** Informando además cuantos valores fueron ingresados.	28/04
30)	**Programa tiene dos partes. Resuelve ambas por separado.**  **a)-** Hacer un programa que lea un numero, calcule y muestre el numero pero incrementado en el 1,5% de su valor. **b)-** El banco **"Terroba S.A."** pagará un interés acumulado diario del 1,5% del importe depositado por el ahorrista. Si transcurre un año (365 días) desde el momento del deposito, se debe calcular: **b.1)-** Cual será el interés porcentual acumulado total recibido, **b.2)-** El importe total acumulado que ganó (recibe) en concepto de intereses, a lo largo del periodo transcurrido y.. **b.3)-** Cuanta plata dispone ahora en su cuenta el ahorrista, luego del periodo transcurrido. El importe depositado por el ahorrista será ingresado al comenzar el programa. **Recuerda la Sintaxis** While Condición: for i in range(máximo):	05/05
31)	Crear un algoritmo que sea capaz de leer un número y calcular la mitad y repetir el proceso de la división (cálculo de la mitad) mientras que el resultado de la división, sea mayor que 1/3 (un tercio). Al terminar, informar cuántas veces se repitió el calculó de la mitad del número leído.	05/05
32)	**Intervalos.** Este programa tiene tres partes. Resuelve las tres. **A)-** Acá podrás ver la forma de preguntar si un número cualquiera, pertenece a un intervalo dado. Baja el código, analízalo y Explícalo detalladamente en tu carpeta y luego continúa con el presente ejercicio. Codigo/02 Intervalos 000 Formas de Preguntar 01.txt **B)-** Realizar un programa (sin usar listas), que sea capaz de detectar e informar que intervalo de valores tiene una mayor ocurrencia (se repite más) dentro de una serie de temperaturas informadas por un Sensor (termómetro) instalado un una central meteorológica ubicada en un sector de la campiña. Los valores correspondientes a los intervalos estudiados son los siguientes: - **Intervalo a**: Menos infinito hasta cero, sin incluirlo (X < 0). - **Intervalo b**: 0 <= X < 20. - **Intervalo c**: 20 <= X < 40. - **Intervalo d**: Todo Valor X >= 40. El programa termina cuando se ingrese un valor igual a cero. **C)-** Modificar el programa para que informe que porcentaje de mediciones tiene cada intervalo, respecto del total las mediciones registradas (el 100%).	05/05
33)	El alumno debe crear un algoritmo (diagrama de flujo) y posterior codificación con Python, en el que hay que ingresar de a uno, una serie de números y avisar por monitor, cual o cuales de los números leídos tienen su doble igual a su cuadrado. Antes de finalizar el programa informar Cuántas veces se produjo este evento y cuanto valores se leyeron durante la ejecución del programa. El último valor (NO procesable) deberá ser el 0.	05/05

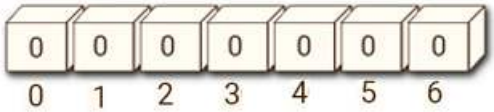
Nro	Enunciados Variados - De Todo un Poco.	Finaliza
34)	**ACÁ** Deberás Incluir el Trabajo práctico de matemática: Cálculo de la Raíz Cuadrada de un número Positivo. A continuación la dirección donde puedes obtener el documento con la explicación del proceso para realizar el cálculo (lo explicado en clase de matemática), una vez seguro del proceso, realizar diagrama y codificación en Python del problema. (Deberás presentar el programa funcionando, y usando la lógica explicada). TP_Matematica_10_Raiz_Cuadrada.pdf 	05/05
	Recuerda que a partir hoy, debes comenzar con los preparativos para que trabajemos con IA "**Práctico Adicional B**" Revisa en la página del curso, que hacer y fechas limites.	

Vectores / Listas. Comenzamos a Usar esta Herramienta (Jueves 07 de Mayo).

Definición, Carga y Diagrama de un Vector Estático/Dinámico:
(Agregar estos programas a la Carpeta y Pendrive)

a)- Estático de 100 posiciones. Se crean los lugares al definir y en el mismo acto se guarda un cero (Valor Inicial).

```
CantElem = 100 # Cantidad elementos que contendrá el Vector
Val_Inic = 0 # Que contendrá inicialmente cada elemento
A =[Val_Inic for i in range(CantElem)]
```



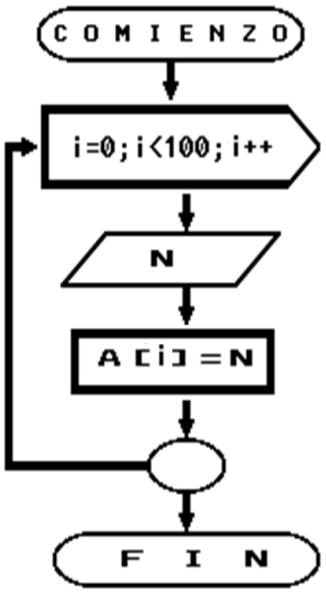
Y sigue hasta 99

```
for i in range (CantElem):
    N=input("Ingrese un numero: ")
    A[i] = N
```

b)- Dinámico, para cargar luego 100 elementos. Acá se Crea cada lugar (posición) en el mismo momento en que se guardar el elemento.

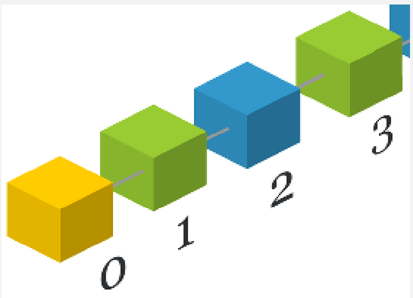
```
CantElem = 100
A = [ ]  # Define Vector Vacío

for i in range (CantElem):
    N=input("Ingrese un numero: ")
    A.append(N)  # Guarda
```



Clase Teórica

Recordemos como usarlos.



En programación, una lista es una estructura de datos que nos permite almacenar múltiples valores en una sola variable. Puedes pensar en una lista como una colección ordenada de elementos. Estos elementos pueden ser de cualquier tipo de dato, como números, cadenas de texto, booleanos e incluso otras listas.

Dependiendo del lenguaje de programación usado, es común, encontrar que a las listas también se las llame o identifique como Arreglos, Vectores, Tablas, etc.

En Python, las listas se crean utilizando corchetes "**[ ]**", entonces una lista vacía se define:

```
Lista = [ ]
```

O también podemos definir una lista que ya contenga algunos elementos, y los elementos se separan por comas:

```
mi_lista = [1, 2, 3, 4, 5]
```

Aquí hemos creado una lista llamada "**mi_lista**" que contiene los números del 1 al 5. Cada elemento en la lista tiene una posición específica, conocida como índice. El primer elemento tiene el índice 0 (cero), el segundo tiene el número de índice 1 (uno) y así sucesivamente.

Entonces, puedes acceder a los elementos individuales de una lista utilizando su índice. Por ejemplo:

```
print(mi_lista[0])  # Imprime el primer elemento: 1
print(mi_lista[2])  # Imprime el tercer elemento: 3
```

También puedes modificar los elementos de una lista asignándoles un nuevo valor:

```
mi_lista[1] = 10
print(mi_lista)  # Imprime la lista modificada: [1, 10, 3, 4, 5]
```

Además de acceder y modificar elementos individuales, las listas tienen muchas otras operaciones útiles. Algunas de ellas son:

- Agregar elementos a una lista:

```
mi_lista.append(6)   # Agrega el número 6 al final de la lista
mi_lista.insert(2, 7)   # Inserta el número 7 en el índice o posición 2 (dos) de la lista.
```

- Eliminar elementos de una lista:

```
mi_lista.remove(3)   # Eliminará de la lista, el primer elemento cuyo contenido es igual a 3.
del mi_lista[3]  # Elimina de la Lista el elemento cuyo índice sea 3.
mi_lista.pop(0)   # Elimina y devuelve el elemento en el índice o posición 0 (cero) de la lista.
```

- Obtener la longitud de una lista:

```
longitud = len(mi_lista)   # Devuelve la cantidad de elementos en la lista.
```

- Sabias que:

```
mi_lista = ['Juan', 'Pedro', 'Laura', 'Carmen', 'Susana']

print(mi_lista[0])  # Muestra Juan (la primera posición es la 0)
print(mi_lista[-1]) # Muestra Susana
print(mi_lista[1]) # Muestra Pedro
print(mi_lista[2])  # Muestra Laura
print(mi_lista[-2]) # Muestra Carmen
```

- Lo Importante: A continuación te dejo algunos programas con más ejemplos.

Baja Código 01:	Codigo/07_Listas_Vectores_000_Declara_Define_A_Estaticos_01.txt
Baja Código 02:	Codigo/07_Listas_Vectores_000_Declara_Define_B_Dinamicos_01.txt

- Y Algo más: Puedes inicializar un vector al momento de definirlo, generando o leyendo los datos que en el guardarás. Esto ya se había mostrado anteriormente, pero acá tienes varios ejemplos. Es interesante, Analízalos:

Baja Código 03:	Codigo/07_Listas_Vectores_000_Declara_Define_C_Genera_Datos_Inicializacion_01.txt

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- Investigar que hacen y como funcionan:

Función/Método	Muy Breve descripción para referencia
clear( )	Elimina todos los elementos de la lista
index(x)	Retorna el índice basado en cero del primer elemento cuyo valor sea igual a x
count(x)	Retorna el número de veces que x aparece en la lista.

- **Investiga** el uso de "**random**" aplicado a listas. Dejo Programa Ejemplo, bájalo de la nube y analízalo.

Baja Código 04:	Codigo/90_Interesante_003_Generacion_Numeros_Aleatorios_B.txt

Nro	Enunciados con Listas (vectores) - Cantidad de elementos Pre-Definida y algo más.	Finaliza

Nota Importante: Será VALIDO para TODO EL AÑO LECTIVO, a menos que se indique lo contrario. 07/05

- No esta permitido el uso de los Comandos:

• break	Salta Fuera ciclo	• continue	Salta Fuera ciclo	•	

- No esta permitido el uso de las Funciones:

• sorted( )	Ordena Elementos	• max( )	Mayor Elemento	• min( )	Menor Elemento
• mean( )	Promedia	• sum( )	Suma Elemento	• break	Salta Fuera ciclo

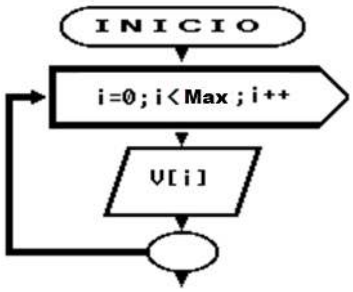
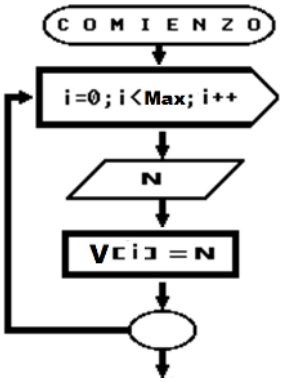
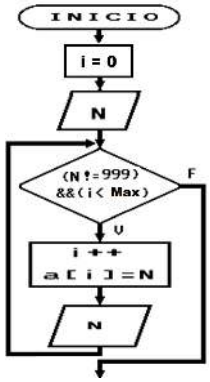
- No esta Permitido el uso del Método: :

• sort( )	Ordena elementos del Vector	• index( )	Índice de la primera ocurrencia de un valor	• count( )	

Se preguntará en clase: Que hacen, Por que no los puedes usar (por ahora) y Cual es la diferencia entre una **Función** y un **Método.** Pregunta al profesor y/o Investiga.

Clase Teórica

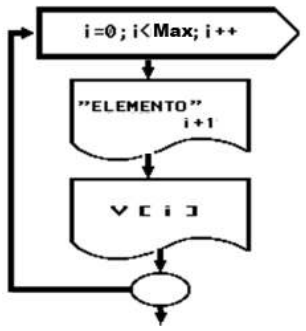
Importante: Recuerda que un diagrama de flujo describe la lógica *pura* que debes desarrollar en tu algoritmo, representando los pasos y decisiones de manera abstracta. Por lo tanto, **no se enfoca en las particularidades sintácticas de los distintos lenguajes de programación**, como las formas específicas en que se definen o declaran las variables, los tipos de datos o las estructuras de control propias de cada lenguaje. Esto permite que cada programador, al implementar el algoritmo en un lenguaje específico (Python, Java, C++, etc.), defina y/o declare las variables de la manera más apropiada y eficiente según las convenciones y características de dicho lenguaje. El diagrama de flujo es la hoja de ruta lógica, y la implementación en código es la traducción a un lenguaje concreto. 07/05
A continuación veremos algunos diagramas de flujo, que independientemente de la forma en que definas los vectores con Python, podrás analizar su lógica. Y claro, a continuación, siempre podrás agregar más cosas a tú programa.

Carga de un vector (Formato 01)	Carga de un vector (Formato 02)	Carga de un vector (Formato 03)
 Otras Acciones o Finaliza.	 Otras Acciones o Finaliza.	 Otras Acciones o Finaliza.
Diagramas/Vector_Parte_Carga_01.png	Diagramas/Vector_Parte_Carga_02.png	Diagramas/Vector_Parte_Carga_03.png

Ahora puedes analizar algunas acciones básicas para trabajar con los vectores. Recuerda que ya se deben encontrar definidos y cargados en memoria.

Nro	Enunciados con Listas (vectores) - Cantidad de elementos Pre-Definida y algo más.	Finaliza

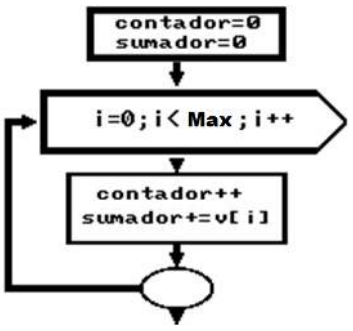
Muestra elementos (Formato 01)



Otras Acciones o Finaliza.

Diagramas/Vector_Parte_Muestra_01.png

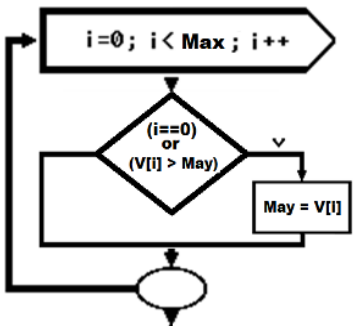
Suma/Cuenta Elementos



Otras Acciones o Finaliza.

Diagramas/Vector_Parte_Suma_01.png

Mayor Valor (Formato 01)



Otras Acciones o Finaliza.

Diagramas/Vector_Parte_May_Elem_01.png

35) **Resuelve** cada una de las tres partes como un ejercicio independiente. — 07/05

a)- Cargar una lista (vector) con 10 números enteros positivos y al finalizar la carga, recorrer el vector mostrando cada uno de los números cargados por el monitor.

Recuerda la Sintaxis
While Condición:
for i in range(máximo):

b)- Modificar el programa anterior, y verifica que cada numero cargado sea realmente entero y positivo. Si al leer un número detectas que no cumple con la condición, avisa del error y lee otro. Asegúrate que los números cargados en el vector cumplan con la condición.

c)- Modifica el programa anterior para que al finalizar la carga, y antes de listar los números cargados en el monitor, informes cuantos números que no cumplieron con la condición debiste rechazar y pedir que sean reingresados.

36) Analiza cada uno de los diagramas de flujo, escribe los enunciados y codificalos. Recuerda agregar el enunciado en la codificación y pegar el diagrama en tu carpeta. Descarga los diagramas de la nube: — 12/05

Diagrama 1:	Diagramas/Vec_Pos_May_Men.png
Diagrama 2:	Diagramas/Vec_Suma_Elem.png
Diagrama 3:	Diagramas/Vec_Mayor_Elem_02.png

Habiendo analizado los diagramas, que opinas? Están completos? Cumplen su cometido?

37) **Resuelve** cada una de las partes como un ejercicio independiente. — 12/05

a)- Cargar una lista con 15 números enteros positivos y al finalizar la carga, calcular e informar la suma de todos los elementos.

b)- Modificar el problema anterior, para que una vez terminada la carga de datos, calcule y muestre el promedio de todos los números cargados en la Lista.

38) **Resuelve** cada una de las partes como un ejercicio independiente. — 12/05

a)- Cargar una lista con 20 números enteros positivos y al finalizar la carga, buscar y mostrar el mayor y menor de los números cargados.

b)- Modificar el programa anterior (punto a), para que al finalizar la carga, muestre la diferencia (resta) del numero (elemento) mayor y el numero (elemento) menor.

c)- Modificar el programa del punto "a", para que luego de cargar los números en el vector, informe por pantalla, cuantos números son mayores que el promedio de todos los cargados y cuantos son menores o iguales que el promedio.

39) Para cada uno de los códigos, realiza el diagrama de flujo correspondiente y confecciona un enunciado apropiado. — 12/05

Código A

```
# Definimos lista Vacía
Valor = [ ]
# Definimos cantidad de elementos a procesar
Cantidad = 10
print("Ingrese", Cantidad, " números.")
for i in range(Cantidad):
    texto = "Ingrese el " + str(i+1) + " Número: "
```

Código B

```
Valor = [ ]
Cant = 0
Suma = 0
Texto = "Ingrese "+str(Cant + 1) + " Elem (Cero termina): "
Nro = int(input(Texto))
while Nro != 0 and Cant < 10:
    Cant += 1
```

Nro	Enunciados con Listas (vectores) - Cantidad de elementos Pre-Definida y algo más.	Finaliza

```
    Valor.append(int(input(texto)))
Mayor = 0
for i in range(Cantidad):
    if Mayor < Valor[i] or i == 0:
        Mayor = Valor[i]
print("\nEl mayor elemento del verctor es: ", Mayor)
```

```
    Valor.append(Nro)
    texto = "Ingrese " + str(Cant + 1) + " Elem (Cero termina):"
    Nro = int(input(Texto))
for i in range(Cant):
    Suma += Valor[i]
Promedio = float(Suma/Cant)
print("\nEl promedio de los ", Cant," elementos: ", Promedio)
```

Nro	Enunciados con Listas (vectores) - Cantidad Máxima de elementos (Puede haber menos)	Finaliza
40)	Para el siguiente código, realiza el diagrama de flujo correspondiente y confecciona un enunciado válido.	19/05

```
Empleados = [ ]
Maximo = 10
Finaliza = ['F', 'FIN']

i=0
texto = "Ingrese Nombre alumno " + str(i+1) + " (Enter o Fin para terminar): "
Nombre = input(texto)
while Nombre and Nombre.upper() not in Finaliza and i < Maximo:
    i = i + 1
    Empleados.append(Nombre)
    texto = "Ingrese Nombre alumno " + str(i+1) + " (Fin para terminar): "
    Nombre = input(texto)

if i > 0:
   tamaño = len(Empleados)
   for i in range(tamaño):
       print("Nombre: ", i+1," : ", Empleados[i])
else:
    print("No se ingresaron Datos")
```

41)	**Resuelve** cada una de las tres partes como un ejercicio independiente. **a)-** Cargar una lista con un máximo de 10 números enteros, la carga debe terminar si algún valor es menor o igual a cero, o la cantidad de valores ingresada sea 10. Al finalizar la carga, recorrer el vector mostrando cada uno de los números cargados por el monitor. **b)-** Modificar el programa anterior, y verifica que cada numero cargado sea realmente entero. Si al leer un valor detectas que no cumple con la condición, avisa del error y lee otro. Asegúrate que los números cargados en el vector cumplan con la condición. (Los números rechazados no deben contarse como números cargados en la lista) **c)-** Modificar nuevamente el programa, para que al finalizar la carga, y antes de listar los números cargados en el monitor, informes cuantos números que no compilan con la condición debiste rechazar y pedir que sean reingresados. (Los números rechazados no deben contarse como números cargados en la lista). **Recuerda la Sintaxis** While Condición: for i in range(máximo):	19/05
42)	**Resuelve** cada una de las partes como un ejercicio independiente. **a)-** Cargar una lista con un máximo de 15 números enteros, la carga debe terminar si algún valor es menor o igual a cero, o la cantidad de valores ingresada sea 15. Al finalizar la carga, recorrer el vector mostrando cada uno de los números cargados por el monitor. **b)-** Modificar el problema anterior, para que una vez terminada la carga de datos, calcule y muestre el promedio de los números cargados. Debes prevenir la división por cero. **Te dejo un rápido ejemplo que hace parte de lo solicitado:** Bájalo de: Codigo/07_Listas_Vectores_002_Suma_Elementos.txt	19/05
43)	**Resuelve** cada una de las partes como un ejercicio independiente. **a)-** Cargar una lista con un máximo de 20 números enteros positivos, la carga debe terminar si algún valor es menor o igual a -1 (menos uno), o la cantidad de valores ingresada sea 20. Al finalizar la carga, recorrer el vector mostrando cada uno de los números cargados por el monitor.	19/05

Nro	Enunciados con Listas (vectores) - Cantidad Máxima de elementos (Puede haber menos)	Finaliza
	b)- Modificar el programa anterior, para que al finalizar la carga, muestre la diferencia (resta) del numero (elemento) mayor y el numero (elemento) menor. **c)-** Luego de cargar los números en el vector, informa por pantalla, cuantos números son mayores que el promedio de todos los cargados y cuantos son menores o iguales que el promedio de todos los cargados. Debes prevenir la división por cero.	

Nro	Enunciados con Listas (vectores) - Sin Cantidad definida de elementos (Simplemente Cargas)	Finaliza
44)	**Resuelve** cada una de las tres partes como un ejercicio independiente. **a)-** Cargar una lista con N números enteros positivos, siendo N ingresado por el usuaria al comenzar el programa y antes de realizar la carga de los elementos del vector. Al finalizar la carga de todos los elementos, recorrer el vector mostrando cada uno de los números cargados por el monitor. **b)-** Modificar el programa anterior, y durante la carga verifica que cada número sea realmente entero y positivo. Si al leer un número detectas que no cumple con la condición, avisa del error y lee otro. Asegúrate que los números cargados en el vector cumplan con la condición. **c)-** Modifica el programa anterior para que al finalizar la carga, y antes de listar los números cargados en el monitor, informes cuantos números que no compilan con la condición debiste rechazar y pedir que sean reingresados. **Recuerda la Sintaxis** While Condición: for i in range(máximo):	19/05
45)	**Resuelve** cada una de las tres partes como un ejercicio independiente. **a)-** Cargar una lista con números enteros, dicha carga de datos debe terminar si algún valor es menor o igual a cero. Al finalizar la carga de datos, recorrer el vector mostrando cada uno de los números cargados por el monitor. (Los números rechazados no deben contarse como números cargados en la lista) **b)-** Modificar el programa anterior, para que puedas verificar que cada numero cargado sea realmente entero. Si al leer un valor detectas que no cumple con la condición, avisa del error y lee otro. Asegúrate que los números cargados en el vector cumplan con la condición. (Los números rechazados no deben contarse como números cargados en la lista) **c)-** Modificar nuevamente el programa, para que al finalizar la carga, y antes de listar los números cargados, informes por monitor, cuantos números que no cumplían con la condición debiste rechazar y pedir que sean reingresados. (Los números rechazados no deben contarse como números cargados en la lista)	19/05

FUNCIONES. (Termina Repaso, Comienzan Nuevos Temas)

Clase Teórica

Una función no es más que un bloque de código (líneas de código) aislado (como un programa aparte, separado del programa principal) que lleva a cabo una tarea específica. Ahora explicado formalmente, podemos decir que, una función permite definir un bloque de código reutilizable que se puede ejecutar dentro del programa, tantas veces como sea necesario (lo escribes una sola vez y lo usas muchas veces). **Las funciones** te permiten crear soluciones modulares y DRY para problemas complejos.

(Dry): "No Repitas". *En un principio, estaba destinado a reducir la repetición de patrones de software, reemplazándolos con abstracciones o utilizando la normalización de datos (eliminar repeticiones) para evitar la redundancia.*

*En un principio, DRY se establece como "**Cada pieza de conocimiento** debe tener una representación autorizada, única e inequívoca dentro de un sistema". El principio ha sido formulado por Andy Hunt y Dave Thomas en su libro The Pragmatic Programmer. Lo aplican de manera bastante amplia para incluir "esquemas de bases de datos, planes de prueba, el sistema de construcción, incluso documentación". Cuando el principio DRY se aplica con éxito, la modificación de cualquier elemento*

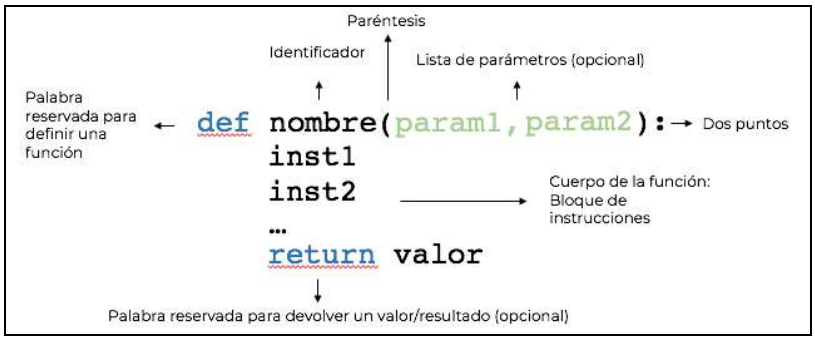
26/05

individual en un sistema, no requiere un cambio en otros elementos lógicamente no relacionados. Además, todos los elementos que están lógicamente relacionados cambian de manera predecible y uniforme, por lo tanto, se mantienen sincronizados. Además de usar métodos y subrutinas en su código, Thomas y Hunt confían en generadores de código, sistemas de compilación automáticos y lenguajes de secuencias de comandos para observar el principio DRY en todas las capas.

Si bien Python ya proporciona muchas funciones integradas, como **print( )** y **len( )**, también nosotros como programadores, podemos definir nuestras propias funciones, que podremos usar en los proyectos.

Una de las grandes ventajas de usar funciones en nuestro programa (proyecto), es que reduce el número total de líneas de código del proyecto, por lo tanto más fácil de entender y corregir.

Como Definir una Función.



Para definir una función, comenzamos con la instrucción **def,** luego le damos un nombre descriptivo de función (como queremos que se llame), y para esto aplican las mismas reglas que para el nombre de las variables, continuamos con el paréntesis de apertura y cierre. Como toda estructura de control en Python, la definición de una función, finaliza con dos puntos (:) y el algoritmo que la compone, irá identado con 4 espacios. ***Veamos un ejemplo muy simple:***

Descarga Diagrama:	Diagramas/09_Funciones_001_Saludo_001_Ejemplo_01.png

```
def  Mi_Primera_Funcion( ):
    print( "Hola Mundo" )
```

Y como se mencionó antes, en nuestro programa, podremos usar la función, tantas veces como sea necesario, simplemente llamándola. Puedes bajar de la nube el ejemplo completo, el programa que tiene la función y el programa que la usa. Cuando lo pongas en el IDE podrás ver y analizar lo que aparece en la pantalla.

Descarga Diagrama:	Diagramas/09_Funciones_001_Saludo_001_Ejemplo_01_Completo.png
Descarga Código:	Codigo/Codigo/09_Funciones_001_Saludo_001_Ejemplo_01.txt

```
def  Mi_Primera_Funcion( ):
    print( "Hola Mundo" )

print("\n----- Y Ahora el Programa --------")

Mi_Primera_Funcion()  # Programa y llamada a la Función
```

MUY IMPORTANTE: Por claridad, antes y después de la definición de una función, deja al menos una o dos líneas en blanco.

Los Parámetros

A una función, le podemos **entregar datos** para que los procese, estos datos **se llaman Parámetros**.
Entonces, formalmente, un parámetro es un valor que la función espera recibir cuando sea llamada (invocada), a fin de ejecutar acciones con ellos. Una función puede recibir uno o más parámetros (que irán separados por una coma) o ningún parámetro, si no hacen falta (tal como sucedió en el primer ejemplo).

Los parámetros, si es que los hay, se indican entre los paréntesis, a modo de variables, a fin de poder utilizarlos como tales, dentro de la misma función (como siempre, puedes bajar de la nube el programa completo para analizar).

Diagrama:	Diagramas/Ejemplo_Primeras_Funciones_02.png
Bajar código en:	Codigo/09_Funciones_001_Saludo_001_Ejemplo_02.txt

```
def Saludar(nombre):
    mensaje = "\nHola " + nombre + "!"
    print(mensaje)
```

Una variante del mismo programa seria (como siempre, puedes bajar de la nube el programa completo para analizar). Debes notar que el diagrama es el mismo que en el programa anterior, puedes explicar el porque?:

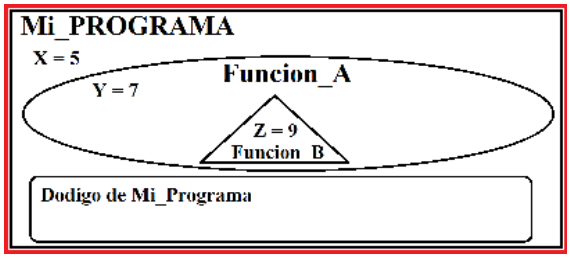
Diagrama:	Diagramas/Ejemplo_Primeras_Funciones_02.png
Bajar código en:	Codigo/09_Funciones_001_Saludo_001_Ejemplo_03.txt

```
def Saludar(nombre):
    print(f"Hola {nombre}!")
```

Algo muy importante y que siempre hay que recordar:

- Los parámetros que una función recibe, solo podrán ser usados por ella y dentro de ella (como parte del código que se escriba dentro de ella), es decir, podrán ser usadas de igual forma que cualquier variable que se defina dentro de la función, "variables de ámbito local" o simplemente "**Variables locales**".
- Si la definición de una función incluye parámetros, cuando llames a la función, le debes proporcionar el mismo número de parámetros y en el mismo orden que los espera.

Aprende a Diferenciar los Parámetros de las Variables Globales y Variables Locales: Tal como se acaba de definir, un parámetro es por donde ingresan los datos a la función, y las variables locales son aquellas que se declaran dentro de una función y solo son accesibles dentro de esa función. Esto significa que su alcance está limitado a la función en la que se definen, y no pueden ser utilizadas fuera de ella. **Por otro lado**, las variables globales son aquellas que se declaran en el bloque de código que contiene la función, es decir, fuera de la función y pueden ser accedidas y modificadas desde cualquier parte del bloque de código donde se declararon.



Analicemos un Ejemplo: MiPrograma (Representado por la imagen de la derecha), tiene la Variable "X" (Variable Global para el programa) que podrá ser usada en cualquier parte del programa (dentro del Cuadrado).
Luego, Tenemos la Funcion_A (representada por el óvalo), que en su interior tiene la Variable "Y" (Variable Local a la Funcion_A), que podrá ser utilizada en cualquier parte de la Funcion_A, incluso dentro de la Funcion_B (que está incluida dentro de la Funcion_A), pero no la podremos usar en el programa.
Finalmente Está la Funcion_B (Función Local de la Funcion_A) que tiene en su interior la Variable "Z", que es local a la Funcion_B, y solo podrá ser usada dentro de la Funcion_B. Hay otros aspectos que deberemos contemplar, pero esto lo veremos en clase.

Tipos de Parámetros que Recibe una Función

Este punto si no lo comprende de inmediato no importa, regresa luego y repásalo. Más adelante será importante. Ahora al inicio no.
Los parámetros se diferencian según el tipo de datos que sean: Parámetros Mutables y Parámetros Inmutables.
Los Inmutables, como int, float, str, tupla: También llamados Parámetros por Valor. Si se modifica el parámetro dentro de la función, en realidad se crea un nuevo objeto en la memoria. El valor fuera de la función permanece sin cambios.
Los Mutables, como list, dict, set: También llamados Parámetros por referencia. Si se modifica el parámetro dentro de la función (por ejemplo, agregando o cambiando un elemento de una lista), el cambio afecta al objeto original fuera de la función.

Diagrama:	Diagramas/Ejemplo_Tipos_Parametros_01.png
Código Ejemplo:	Codigo/09_Funciones_005_Parametros_Mutables_e_Inmutables_001_Ej_01_Lista_e_int.txt

Código	Veras en el Monitor

def modificar_lista(lista):
  lista.append(4) **# Esto modifica el objeto original**
def ModificarNumero(numero):
  numero += 1 **# Se crea nuevo objeto, no afecta original**
Mi_Lista = [1, 2, 3]
Mi_Numero = 10
print(f"Numero Original: {Mi_Numero}")
ModificarNumero(Mi_Numero) **# Número original 10**
print(f"Numero Final: {Mi_Numero}")
	Numero Original: 10
	Numero Final: 10
	
	Lista Original: [1, 2, 3]
	Lista Final: [1, 2, 3, 4]

```
print(f"Lista Original: {Mi_Lista}")
modificar_lista(Mi_Lista) # La lista es Alterada
print(f"Lista Final: {Mi_Lista}")
```

Nro	Enunciados con Funciones	Finaliza
46)	Primer Ejemplo completamente desarrollado. Analízalo detalladamente y luego inclúyelo en la carpeta. **Enunciado:** Hacer un programa en el que se lean dos números para calcular y mostrar la suma. Estos ejercicios serán resueltos en clase. **Tú los completas con diagramas de flujo y agregas en la carpeta**. Debes usar las tres funciones y cuidado con los parámetros. - LeeNumero( ). - SumaDosNumeros( ). - MuestraNumero( ).	O4/06

Diagramas/Ejemplo_Primeras_Funciones_03.png

Codigo/09_Funciones_002_Suma_Dos_Numeros_Eje_01.txt

```
# A este lo Llamaremos Programa 1 (Uno)
def Lee_Numero():
    Num = int(input("Ingresar Numero Entero: "))
    return(Num)
def Suma_dos_Numeros(N1,N2):
    R = N1 + N2
    return(R)
def Muestra(Numero):
    print("\nLa suma es: ", Numero)
# Este es el Programa. Analiza como usar las funciones:
A = Lee_Numero()
B = Lee_Numero()
Suma = Suma_dos_Numeros(A,B)
Muestra(Suma)
```

Diagramas/Ejemplo_Primeras_Funciones_04.png

Codigo/09_Funciones_002_Suma_Dos_Numeros_Eje_02.txt

```
# A este lo Llamaremos Programa 2 (Dos)
def Lee_Numero():
    Num = int(input("Ingresar Numero Entero: "))
    return(Num)

def Suma_dos_Numeros(N1,N2):
    R = N1 + N2
    return(R)

def Muestra(Numero):
    print("\nLa suma es: ", Numero)

# Este es el Programa. Analiza como usar las funciones :
Muestra(Suma_dos_Numeros(Lee_Numero(),Lee_Numero()))
```

47)	Hacer un programa que lea dos números y muestre el mayor de ambos. IMPORTANTE: Usted debe **Crear y usar las funciones definidas por el usuario**: **a-** Leer un número( ) **b-** Mostrar mayor( ) **c-** Busca mayor( ) En el Código y la carpeta (con un comentario), identifica si las **variables usadas son** Parámetros o variables **Locales** y/o **Globales.**	O4/06
48)	Hacer una función (y el programa que la usa) que permita saber, si en el programa principal, se puede o no realizar una división o habrá error al dividir por cero. **Recuerda** hacer la división (si es posible) y que si el denominador es cero, el proceso tendrá un error. Si el enunciado no te queda claro, consulta con el profesor. **Te puede servir:** Prevenir División por cero — Codigo/09_Funciones_004_Prevenir_Division_por_Cero_01.txt	O4/06
49)	Hacer un programa en el que se leen tres números positivos, y se muestra la Raíz cuadrada del mayor. Deberás crear una función que reciba como parámetro un número y retorna la raíz. IMPORTANTE: Debes crear y usar las siguientes funciones: **a-** Lee un número. **b-** Busca el mayor número. **c-** Calculo de la Raíz Cuadrada. Debes usar el método aprendido en matemática. d- Muestra un número. A la función **Muestra un número,** deberás usarla para mostrar la raíz cuadrada calculada. En la carpeta (con un comentario), identifica si una **variable** es **Local** o **Global.**	O4/06
50)	Hacer un programa que, lea una serie de números (la serie termina cuando se lea un cero). Este programa debe seleccionar e imprimir el mayor de todos los números leídos. IMPORTANTE: El programa deberá contener y usar las siguientes funciones: **a-** LeeUnNumero( ). **b-** BuscaMayorNumero( ).	09/06

Nro	Enunciados con Funciones	Finaliza
	c- MuestraUnNumero( ). A la función **Muestra un numero,** deberás usarla para mostrar el mayor valor encontrado.	
51)	Realiza un programa que lea y calcule el promedio de 3 números. Utilizar las funciones: **a)** LeeUnNumero( ). **b)** Promedio( ). **c)** MuestraResultados( ). El objetivo es realizar el programa con la menor cantidad de líneas posibles. Colocar las funciones en el lugar donde envías los parámetros (Críptico). Recuerda que lo hicimos en clase, solo debes probarlo y/o usarlo nuevamente en la PC. Realiza la prueba de escritorio y recuerda que esta debe coincidir con el diagrama y codificación. **En la carpeta** (con un comentario), **identifica si** una **variable** es **Local** o **Global.**	09/06
52)	Realizar un programa que permita calcular y mostrar el promedio de una serie de números. Terminará el ingreso de números cuando algún elemento de la serie sea un valor negativo. Recuerda que debes usar las siguientes funciones. **a)** LeerUnNumero( ). **b)** Suma( ). **c)** PuedoDividir( ) **# Verifica que no se divida por cero.** **d)** Promedia( ). **e)** MuestraResultado( ). Importante: Recuerdas que puedes usar listas.... Debes realizar el diagrama, prueba de escritorio y codificación (como siempre, que todas las partes coincidan, o será considerado como MAL).	09/06

OPCIONAL

Funciones como Parámetros: Este tema es Opcional, aunque muy interesante.
Una función, puede recibir otras funciones como Parámetros, o dicho de otra forma: Las funciones También pueden ser enviadas como parámetros. Esta funcionalidad es práctica y suele ser muy utilizada, siendo en este momento, simplemente mencionada, dejándote un ejemplo para que lo analices y si es de tu agrado, lo utilices en donde consideres necesario hacerlo.

Bajar código en:	Codigo/09_Funciones_005_Parametros_con_Funciones_001_Ej_01_Paso_Una_Funcion.txt

Nro	Cosas Interesantes (Parte II)	Finaliza
	Memoria Ocupada: Puedes Obtener la memoria en "bytes" que ocupa una estructura de datos o variable. Pero cuidado, no confundas Cantidad de elementos con Tamaño Ocupado. Baja desde acá: Codigo/90_Interesante_015_Memoria_Usada_01_Estructuras_A.txt	
	Diccionarios: Un diccionario en Python, es una estructura de datos que sirve para almacenar pares de claves y valores. Resumiendo: Es como un "diccionario real", donde buscas una palabra (clave) para encontrar su significado (valor). Cada clave actúa como un "nombre" único que te permite acceder a su valor asociado. Analiza este ejemplo:	

Ejemplo 01

```
contactos = { "Ana": "123-456",
              "Luis": "987-654",
              "Sofía": "456-789"
             }

#%% En EL Programa
print(contactos["Ana"]) # Devuelve: 123-456
```

Ejemplo 02

```
descripciones = { "S": "Supervisor",
                  "U": "Usuario",
                  "V": "Visitante"
                 }

#%% En EL Programa
print(descripciones["U"]) # Devuelve: Usuario
```

Puedes Bajar Otros Ejemplos, es un programa completo y muchos ejemplos, podrás

Nro	Cosas Interesantes (Parte II)	Finaliza

analizar prácticamente todas las alternativas de este ejemplo:

Descarga desde:	Codigo/Codigo/25_Utiles_001_Diccionario_Tipo_01_A_Declaracion_y_Uso.txt
Acá más sobre Diccionarios:	Diccionarios_Creacion_y_Uso_Parte_01.pdf
Acá más sobre Diccionarios:	Diccionarios_Creacion_y_Uso_Parte_02.pdf

Colores en el Texto (Parte 1): Si en tus programas debes resaltar algunas palabras o textos, acá te dejo una forma simple y rápida de hacerlo.

Colorama es un Paquete (parte de una librería) que, actúa como una capa que traduce códigos de escape ANSI (utilizados para colorear texto en terminales de sistemas tipo Unix) a un formato que también es compatible con Windows. Esto significa que puedes usar colores en cualquier sistema operativo sin preocuparte por compatibilidad.

Colorama funciona en terminales estándar, pero no en interfaces gráficas o IDEs que no soporten códigos de escape (como algunas versiones antiguas de IDLE). En terminales personalizadas (como ciertas extensiones de VSCode o terminales remotas), podrías necesitar probar la compatibilidad.

Puedes bajar de la nube el programa con ejemplos, muy simples de aplicar.
(ya sabes como usar el link).

Bájalo desde:	Codigo/90_Interesante_004_Colores_en_el_Texto_01_Colorama_Parte_I.txt

SEPARAR DATOS Leídos dentro de UNA CADENA

Nro	Enunciados con Separación de Datos (Campos)	Finaliza

Clase Teórica

Único Separador de Campo: Como separar o individualizar los datos, cuando se ingresan en una solo línea y separados por algún caractér, por ejemplo una coma. Analiza y confecciona un enunciado válido para el siguiente código (ejemplo):

16/06

```
Linea = input("Ingrese tres números Enteros separados por una COMA: ")
A => Linea = Linea.translate({ord(' '): None})  #elimino espacios si los hubiera
B => N1, N2, N3 = Linea.strip().split(",")  # Separa datos
if N1.isdigit() and N2.isdigit() and N3.isdigit(): # Verifico que los caracteres sean dígitos
    N1 = int(N1) # lo transformo en numero
    N2 = int(N2) # lo transformo en numero
    N3 = int(N3) # lo transformo en numero
    print("\nSumando los Números Leídos: ", N1,"+ ", N2,"+ ", N3,"= ",N1+N2+N3)
else: print("\nAlgunos Datos Son Incorrectos")
```

Explicación

Linea, es una variable que contiene la cadena de caracteres que leímos (La podríamos haber llamado Cadena, o Cajita).

Renglón A: Elimina (reemplaza por "**None**", es decir nada) Todos los espacios en blanco que encuentre en la cadena de caracteres. (No siempre servirá)

Renglón B: Esta línea toma la cadena (Para este ejemplo, ya sin espacios de la línea anterior), intenta eliminar (si encontrara) espacios en blanco al comienzo y al final de la cadena, y a continuación, la divide la cadena resultante en partes, usando el separador indicado (la coma) como marca, eliminándola en el proceso (elimina las comas) y finalmente guarda cada parte en las variables N1, N2 y N3. **Analicemos cada parte:**

.strip(): Quita cualquier espacio extra que pudiera haber **al inicio o al final** de la cadena (aunque la línea anterior ya hizo gran parte de este trabajo), también elimina caracteres no visibles como saltos de línea "`\n`", tabulaciones "`\t`", etc.

.split(","): Una vez tenesmos el renglón limpio, busca las comas (,) que se usa para separar cada datos (campo) y corta la cadena cada vez que encuentra una (separando los

Clase Teórica

datos (campos) leídos. Esto crea una lista con las partes separadas por comas.

Finalmente, los elementos de esa lista se asignan, uno por uno, en las variables que nosotros ponemos para que sean guardados: N1, N2 y N3.

Interesante: En ves de tres variables, podrías haber utilizado una lista, donde automáticamente quedarían almacenados los datos (uno en cada posición)

A continuación, puedes bajar dos ejemplos completos de la nube, analízalos, seguro te ayudarán:

Ejemplo 01:	Codigo/08_Archivos_004_TXT_000_Separo_Datos_Leidos.txt
Ejemplo 02:	Codigo/08_Archivos_004_TXT_000_Limpio_Datos_Leidos.txt

Sugerencia: también investiga y comprende que hacen los métodos: **rstrip( ) y lstrip( )**.

Dos o Más Separadores de Campo Alternativos: Cuando el separador de campo no es único y tienes dos o más alternativas, por ejemplo, un espacio y/o una coma, etc:

```
import  re
Linea = "Manzana,   Pera  ::::: Banana Uva"
Resultado = re.split(r"[,:\s]+", Linea.strip())
print(Resultado)   # ['Manzana', 'Pera', 'Banana', 'Uva']
```

Baja el ejemplos de la nube y analizarlo (ahí encontraras la explicación del funcionamiento), seguro te ayudará:

Ejemplo:	Codigo/08_Archivos_004_TXT_000_Limpio_Datos_Leidos_con_DOS_Separadores.txt

53) Este Ejercicio tiene dos partes, resuélvelas.

a)- Cuando trabajamos con Python, que diferencia existe entre una función y un método? pueden hacer lo mismo?

b)- Analiza los Diagramas, si puedes, modifícalos y usa alguna función. Escribe los enunciados y Codifícalos. Pega todos los diagramas (indicando cual es el original y cual el modificado) y códigos en tu carpeta.

(Descarga los diagramas desde la nube)

Diagrama 1:	Diagramas/Par_Resta.png
Diagrama 2:	Diagramas/Par_Cociente_B.png

54) Realizar un programa que lea por teclado, una lista con datos (renglón a renglón). Cada renglón representa un día y contiene las 3 (tres) temperaturas registradas durante ese día (el programa deberá leer las temperaturas como una terna ordenada, donde los valores están separados por una coma). Es decir, un renglón representa las temperaturas de un día. El proceso de lectura termina cuando se lean las 3 temperaturas correspondientes a un día igual a cero.

T1	T2	T3
T1	T2	T3
...	...	...

A continuación te dejo ejemplos en donde se usan/cargan 3 vectores paralelos (Dinámicos y Estáticos) Analízalos, seguro te ayudaran.

Imagen Ilustrativa:	Diagramas/Cargar_3_Vectores_Numericos.png
Vector Estático:	Codigo/07_Listas_Vectores_006_Carga_Paralelos_01_Estaticos_A.txt
Vector Dinámico:	Codigo/07_Listas_Vectores_006_Carga_Paralelos_02_Dinamicos_A.txt

El programa deberá procesar, buscar, calcular y mostrar:

Recuerda la Sintaxis
While Condición:
for i in range(máximo):

a) ¿Cuantos días de duración tiene el experimento?
b) ¿Número de día registra la temperatura promedio más alta?
c) ¿Que número de día tiene la mayor amplitud térmica? **(Amplitud = Mayor - Menor)**

Usar las siguientes funciones (en el orden que consideres necesario)**:**
- AmplitudTermica( ).
- TemperaturaPromedio( ).
- LeerUnNumero( ). <= **Puedes usar e**sta función, que lee un número, si te resulta útil.
- LeerTerna( ). <= **O puedes usar e**sta otra función si LeerUnNumero( ) no te sirve

Numera cada línea de tu programa y ponerle un comentario o texto explicativo. Y en la carpeta (con un comentario), identifica si una **variable** es **Local** o **Global.**

55) En el instituto de enseñanza **"Aprenderemos S.A"**, el preceptor de cada curso, escribe en hoja de papel, un listado que contiene el legajo y tres notas de cada uno de sus alumnos (ver imagen) correspondientes a las tres evaluaciones del período controlado de una materia.

LISTADO DE NOTAS			
Num. Legajo	Nota 1	Nota 2	Nota 3

Debes realizar un programa que permita **calcular y mostrar** la siguiente información:

a)	El promedio de cada uno de los alumnos.
b)	El promedio del curso, correspondiente a cada una de las 3 evaluaciones (promedio de cada columna, correspondiente a una evaluación). Siempre debes prevenir la División por cero.
c)	Promedio general del curso (el promedio de TODAS las notas de TODOS los alumnos)
d)	Informar el legajo y promedio del alumno que tiene el promedio individual mayor

Información Importante para realizar el programa.

-	La carga de datos, se realiza renglón a renglón (datos separados con una coma) y termina cuando se ingrese un legajo 0 (cero).
-	Los legajos, son números enteros que pueden contener entre 1 (uno) y 3 (tres) dígitos.
-	Todas las notas, son números enteros comprendidos entre 0 y 10.
-	El **promedio** de cada evaluación, **muéstralo** como número real (float) **con 2 decimales**.
-	El **promedio** de cada alumno, **muéstralo** como número real (float) **con 2 decimales**.

El Programa deberá contener y usar las siguientes funciones:

-	LeerDatosDeUnAlumno( ). **Recuerda** controlar que los datos leídos sean correctos
-	PromedioPorAlumno( ).
-	MuestraNumeroConTextoEplicativo( ).
	Esta última función será usada para los cuatro puntos solicitados (a, b, c, y d).

FRACCIONES CON PYTHON - Opcional

Python incluye un módulo llamado "fractions" que permite trabajar con fracciones de manera sencilla y eficiente. Este módulo proporciona la clase "Fraction", que representa números racionales como el cociente de dos enteros: un numerador y un denominador.

El uso de fracciones, es una herramienta poderosa para realizar cálculos precisos con números racionales. La clase "Fraction" te permite realizar operaciones matemáticas complejas de manera sencilla y eficiente.

Bajar de la nube el documento completo:

Documento:	Fracciones con Python.pdf

Operaciones Básicas con Fracciones:

a) Suma y Resta: Puedes sumar y restar fracciones de forma natural. Python se encarga de encontrar un denominador común automáticamente.

b) Multiplicación y División: Multiplicar fracciones es tan simple como multiplicar los numeradores entre sí y los denominadores entre sí. Para dividir, simplemente multiplicas por la fracción inversa.

c) Conversión a Decimal: Las fracciones pueden convertirse fácilmente a su representación decimal utilizando la función "float()". Esto es útil para cálculos que requieren precisión decimal.

d) Simplificación Automática: Una de las características más útiles de la clase "Fraction" es que simplifica automáticamente las fracciones.

e) Comparaciones: Compara fracciones usando los operadores estándar de comparación, como mayor que, menor que, etc.

f) Acceso a Numerador y Denominador: La clase "Fraction" proporciona acceso directo al numerador y denominador a través de sus atributos "numerator" y "denominator."

g) Detectar si una variable contiene una fracción.
A continuación te dejo un programa ejemplo donde encontraras ejemplos de cada una de las cosas que puedes hacer con las fracciones. Bájalo de la nube y analízalo. Es muy fácil usar las fracciones con Python.

Programa:	Codigo/13_Fracciones_001_Operaciones_A_Ejemplos_Operaciones_Iniciales.txt
Programa:	Codigo/13_Fracciones_001_Operaciones_B_Minimo_Comun_Multiplo_01.txt

Nro	Enunciados con Fracciones	Finaliza
56) OPCIONAL	Crear un programa que, **visualice un menú que permita elegir alguna de las siguientes opciones**: **a)-** Leer y Sumar dos fracciones: En esta opción se piden dos fracciones y se muestra el resultado. **b)-** Leer y Restar dos fracciones: En esta opción se piden dos fracciones y se muestra la resta. **c)-** Leer y Multiplicar dos fracciones: En esta opción se piden dos fracciones y se muestra el producto. **d)-** Leer y Dividir dos fracciones: En esta opción se piden dos fracciones y se muestra el cociente. **e)-** Salir. Usar la función "LeeFraccion( )" y la función "MuestraFraccion( )". **Y colores distintos en el menú**.	

Vectores/Listas POSICIONAMIENTO DIRECTO.

Repasando

Repasando Distintas Formas de Definir, Guardar y recorrer Vectores

Acá veremos y/o repasaremos formas/métodos para definir listas/vectores, almacenar sus elementos y recorrerlos.

a)- Definir una lista (vector) con 10 posiciones (que contendrán números enteros), y cuando se estén reservando (para el posterior uso) las posiciones del vector, inicializar en ese momento con un valor especifico, por ejemplo cero (Pregunta lo que no entiendas).

```
1  CantElem = 3 # Cantidad de elementos que contendrá el Vector
2  Val_Inic = 0  # Que contendrá inicialmente cada elemento
3  V_01 = [Val_Inic for i in range(CantElem)] # Creación/Definición del Vector
```

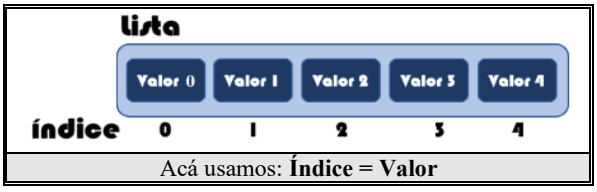
b)- Has pensado en guardar/cargar nombres de personas en vez de números.

A continuación encontrarás un ejemplo con varias opciones para cargar los datos. Si buscas ideas, descárgalo de la nube, analízalo, toma lo que necesites y adáptalo.

Código Ejemplo	Codigo/07_Listas_Vectores_000_Declara_Define_A_Estaticos_01.txt

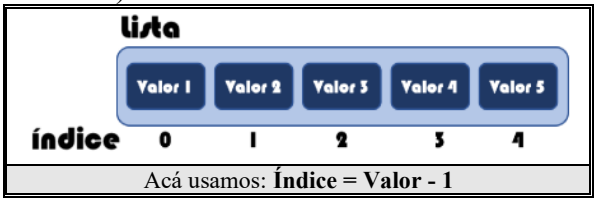
Clase Teórica

Introducción: En este punto, nos enfocaremos en el posicionamiento directo, una técnica que nos permite acceder a elementos específicos de una lista (vector) relacionando el contenido con la posición o índice. Para esto, deberemos asociar algún dato de cada elemento con su posición o numero de orden dentro de la lista. **Por ejemplo**, imaginemos una carrera de bicicletas, en la que cada bicicleta/competidor tiene un numero único e irrepetible, entonces podríamos guardar la información de la bicicleta/competidor numero 1 (uno) en la posición 1 (uno) de nuestra lista, la del competidor 2 en la posición 2, y así sucesivamente. Esto nos permite acceder a la información de cada competidor en forma directa, rápida y eficiente, **entonces** si conocemos el numero de competidor, podremos usarlo (como índice) para acceder a esa posición del vector (nuestra lista), que contendrá toda la información requerida. Seguro puedes decir: Pero no hay Competidor 0 (cero). En este caso tienes dos alternativas:



Acá usamos: **Índice = Valor**

- **a)** Dejas el primer elemento (el de la posición 0) vacío.
- **b)** Simplemente, restas 1 (uno) al numero de competidor, y logras acceder con la misma facilidad a cada posición conociendo el numero de competidor.



Acá usamos: **Índice = Valor - 1**

Lo único que debes tener presente, es que debes conocer de antemano cuantos elementos

Clase Teórica

o posiciones usarás en la lista, **es decir**, cuantos competidores hay en el evento, y crear una lista que contenga esa cantidad de elementos (un lugar para cada competidor) y luego guardar la información requerida en cada posición.

Ejemplo (Diagrama y Código Python):

Las explicaciones de clase, escríbelas en tu carpeta y al pie del código, como un comentario en tu programa, así siempre podrás leerla cuando se te pregunte durante la corrección.

- Analiza detalladamente el diagrama y código Python. Explica que hace y como funciona el segmento de código?
- Que uso puedes darle? Plantea dos enunciados como ejemplo.

```
Cantidad = 10
V=[ 0 for i in range(Cantidad)]
Mensaje = "Ingresa numero entre 0 y " + str(Cantidad) + " (Cero termina): "
n=int(input(Mensaje))
while n>0 and n<Cantidad:
    V[n] += 1
    n=int(input(Mensaje))
for i in range(Cantidad):
    print(f"El numero: {i} , Se leyó: {V[i]} veces.")
```

Diagrama:	Diagramas/07_Listas_Vectores_016_Posicionamiento_Directo_01_Cuenta_Repeticiones_B.png
Código Completo:	Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Cuenta_Repeticiones_B.txt

Nro	Enunciados para Usar Posicionamiento Directo en Listas (Vectores)	Finaliza

57) Este es un solo ejercicio, ejecuta paso a paso las cosas solicitadas:

a)- Define Vector (Lista) que contenga 101 Posiciones, y cada posición esté inicializada en 0 (cero).

b)- Leer por teclado números enteros positivos mayores o igual que cero y menores que 100 (cien). Cada número que lees, te indica en que posición del vector deberás poner un 1 (uno). El proceso de lectura, deberá terminar cuando leas (ingreses) cualquier valor menor que cero o mayor o igual que 100.

c)- Una vez hayas terminado con la lectura de los números, recorre el vector e informa por pantalla, la posición de los elementos que están en 1 (uno).

d)- La respuesta a la pregunta que se formulará a continuación, escríbela en tu carpeta y al pie del código de tu programa, como un comentario de tu programa, así siempre podrás leerla cuando se te pregunte durante la corrección. **Hay alguna similitud/relación entre el informe que realizas por el monitor y los números que ingresaste por teclado? Por que? Explica minuciosamente.**

58) Este programa tiene dos partes, en la primera deberás pensar un poco, pero en la segunda parte encontraras diagramas para que analices, adaptes y completes el código.

Primera Parte: En una carrera de bicicletas, antes de iniciar la competencia, los corredores forman una fila y se los anota en el sistema, entregándole a cada participante un cartel con el número, el que le tocó en la fila (uno al primero, dos al segundo, etc.). El programa debe:

a) Cargar en memoria el número de corredor y su nombre. La carga termina con un "**ENTER**" - Sin nombre.

b) Una vez cargados los datos en memoria, el operador debe poder leer por teclado el número del corredor y el sistema le informará el nombre por el monitor. Esta opción termina cuando se lea un número de participante menor o igual a cero.

c) Al finalizar la competencia, el programa debe permitir cargar los tiempos de cada corredor. Para esto usar un vector paralelo, en el que guardaras los tiempos leídos de cada corredor. Esta opción termina cuando se lea un número de participante menor o igual a cero.

d) Y al finalizar el programa, realizar un listado por monitor, donde muestres el nombre y el tiempo de cada corredor.

Segunda Parte: Modificar la primera parte (Puedes usar funciones) para que al iniciar el programa muestre un menú de selección, con las siguientes opciones:

a) Cargar en memoria los nombres de los participantes de la competencia (La carga termina con un "**ENTER**" - Sin nombre).

b) Dado el número del corredor y el sistema le informará el nombre por el monitor (esta opción termina cuando se lea un numero de participante menor o igual a cero).

c) Leer el número **de un corredor, entonces** el sistema informará el nombre por el monitor y pedirá el tiempo que tardo en llegar a la meta (esta opción termina cuando se lea un número de participante menor o igual a cero).

d) Realizar listado en el que muestre para cada participante (en orden numérico que se asigno a cada uno):

Numero	Nombre	Tiempo

Nro	Enunciados para Usar Posicionamiento Directo en Listas (Vectores)	Finaliza

e) Ingresando el Numero del Corredor, el sistema le permite corregir el nombre si fue mal cargado. (piensa, piensa no es necesario hacerlo otra vez).

f) Ingresando el Numero del Corredor, el sistema le permite corregir el Tiempo del participante en llegar a la meta (piensa, piensa no es necesario hacerlo otra vez).

g) FIN. Terminar el proceso.

IMPORTANTE: El menú debe contener un color para cada opción. - A continuación, encontraras el diagrama de un programa similar que usa el menú, y en cada opción llama a la función correspondiente (alternativa Práctica). Ahora analízalo y adáptalo.

Diagrama:	Diagramas/Uso_Menu_de_Seleccion_10.png

59)

Día Juliano: correspondiente a una fecha, es un número entero que indica los días que han transcurrido desde el 1 de enero del año con que estemos trabajando.
Detectar Año Bisiesto: Será un año Bisiesto, cuando el numero de año con 4 cifras sea divisible por 4 y no sea múltiplo de 100 o sea múltiplo de 400.

a)- Habiendo comprendido la definición de "**Día Juliano**" y comprendido la lógica para detección de un Año Bisiesto, se pide que hagas un programa, que dada una fecha, calcule y muestre el día Juliano, pero cuidado, ya que debes controlar que todos los datos sean apropiadamente leídos. A continuación encontraras algunos programas que te servirán para tu trabajo. Bájalos de la nube, analízalos y adáptalos para tu programa.

Año Bisiesto:	Codigo/09_Funciones_003_Anio_Bisiesto_01_Reconocimiento.txt
Días Por mes:	Codigo/09_Funciones_003_Dias_en_Cada_Mes_01_Reconocimiento.txt
Día Juliano:	Codigo/09_Funciones_003_Fecha_a_Dia_Juliano_01_Calculo.txt

b)- Usando los conocimientos ya adquiridos, hacer un programa que dado un día Juliano, calcule y muestre por monitor la fecha original (Con formato día/mes).

60)

Antes del Receso Invernal

Ejercicio Complementario resuelto en clase. (Para los que siempre quieren usar Tablas/Vectores).

Esto puede ayudar:	Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Cuenta_Repeticiones_A.txt
Imperdible:	Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Cuenta_Repeticiones_B.txt
Ver también:	Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Elimina_Repeticiones_C.txt

En la empresa "Curiosos S.A.", el departamento de personal, confecciona un listado en el que se colocan los nombres de todos los empleados y un número que corresponde al número de orden que tuvo cada empleado al momento de anotarse en el listado. Cargar este listado en memoria usando un vector, donde el número de orden del empleado sea la posición (índice) que ocupa y realizar el fin de carga con el nombre de algún empleado Nulo - No lo ingresa y presiona "**Enter**".

Lista de Empleados
1- Paula
2- Alejandro
3- Pedro
4- Susana
5- José
Etc.

Por otro lado, en la cantina, anotan cada vez que hay una venta, el número del empleado (para identificarlo) y la plata que gastó en esa compra.
Cargar también el listado de compras en memoria, usando un segundo vector, donde el número de orden del empleado sea la posición (índice) que ocupa.

Finalizar el proceso, ingresando un Número de orden menor o igual a cero.

Listado en la Cantina	
Empleado	Importe
4-	$ 32,50
1-	$ 121,25
5-	$ 83,00
4-	$ 54,10
1-	$ 17,25
5-	$ 35,00
Etc	

En base a la información que se reunió entre los dos listados, se quiere saber:

a) Cuanta plata gasto cada empleado.
b) Cuantas compras realiza cada empleado en el periodo controlado.
c) Nombre del Empleado que gasto más en todo el periodo controlado.
d) El importe promedio de compra (sumar los importes de todas las compras y dividir por la cantidad total de compras anotadas).

Ultima Clase Repaso

Para la Próxima clase, deberás traer entendido, sabido y practicado desde tu casa, lo que esta un poquito más abajo: "Archivos de Texto" y "Creación Manual de Archivos de Texto Básicos". Es muy simple, solo tienes que usar un editor de texto. **Lee y estudia lo que se encuentra a continuación**.

VACACIONES DE INVIERNO

ARCHIVOS DE TEXTO.

A un archivo informático lo podemos imaginar como un cuaderno virtual donde puedes guardar diferentes tipos de información, como documentos de texto, imágenes, videos o música.

Un archivo es como tener un armario donde guardas tus cosas digitalmente.

Una descripción formal, será: Un archivo informático es un conjunto de datos almacenados en un dispositivo de almacenamiento, como un Disco duro de computadora, PenDrive, (etc), que puede contener texto, imágenes, audio, video o cualquier otra información.

Desde acá, estudiaremos dos tipos de archivos o ficheros: Los archivos de texto y los archivos binarios.

Un archivo de texto, contiene caracteres que son legibles por el ser humano y están guardados con una codificación (ASCII, UTF-8, …).

61)	Investiga, escribe en tu carpeta y explica detalladamente que significa: **"Un archivo de texto**, contiene caracteres que son legibles por el ser humano y están guardados con una codificación (ASCII, UTF-8, …)". Que son esas codificaciones?	

Una pregunta Frecuente en las evaluaciones: Es lo mismo un archivo UTF-8 que un archivo Caracteres Unicode?.
Respuesta: un archivo UTF-8 y un archivo de caracteres Unicode, no son exactamente lo mismo. Unicode es un estándar de codificación de caracteres que define un amplio conjunto de caracteres, incluyendo letras, números, símbolos (emojis), etc., de diferentes idiomas. UTF-8 es una forma de codificar esos caracteres Unicode en un archivo. Piensa en Unicode como el alfabeto y UTF-8 como la forma en que escribes ese alfabeto en un papel. Puedes usar otros métodos de codificación como UTF-16 o Latin-1, **pero**, UTF-8 es muy común por su eficiencia y compatibilidad. Un archivo UTF-8 contiene datos codificados, y esos datos representan caracteres Unicode. En resumen, todos los archivos UTF-8 usan Unicode, pero no todos los archivos Unicode están codificados como UTF-8.

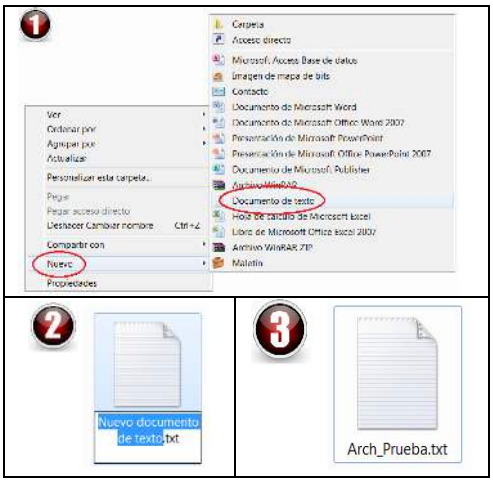
Características de los Caracteres Unicote:	Caracteres_Unicode_Caracteristicas.pdf

Un archivo binario, (al contrario que un archivo de texto) está compuesto por un flujo de bytes que solo tienen sentido cuando son leídos por una PC y que podrán ser interpretados por los programas y/o aplicaciones para los que son creados. Un ejemplo de este tipo de archivos son los archivos que contienen imágenes, música o incluso los mismos programas ejecutables (pre compilado o compilado).

Como crear un archivo TXT con Windows. Si bien esto no tiene que ver con la programación, y mucho menos con cualquier comando Python, es algo que necesitarás cuando comencemos a estudiar el procesamiento de estos archivos. En la primera etapa, comenzaremos con la lectura, luego la escritura y finalizando el año con, *la lectura, procesamiento de los datos y escritura de los resultados en un nuevo archivo*.

Creación Manual de Archivos de Texto Básico (Clase Teórica del Martes 22/Julio).

A continuación encontrarás el paso a paso. Practica creado varios, ya que los necesitarás muy pronto.



a) Primero nos posicionamos en la carpeta donde queremos crear el Archivo de texto.

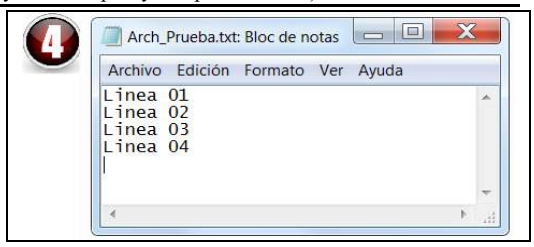
b) Presiona Botón derecho del Mouse, esto provocará que una ventana emergente de Windows se abra. Ahora, en la ventana emergente, elije "**Nuevo**" y luego "**Documento de Texto**". Ver imagen 1.

c) En el acto, aparecerá el icono de un archivo de texto, y esperará a que ingresemos un nombre, el que sea de nuestra preferencia. Ver imagen 2. (**poner un nombre sin espacios en blanco, por que algunas versiones de Python, podrían presentar problemas**)

d) Escribimos el nombre de archivo que usaremos en nuestro programa, ya que será el nombre que nuestro programa busque. Ver imagen 3.

e) Recuerda anotar la dirección o ruta de acceso del archivo, ya que también lo necesitarás (**Si una vez creado el archivo le haces un clic, luego botón derecho, propiedades, podrás ver la ubicación o ruta de acceso a tu archivo recién creado**).

f) **REPITO.** Recuerda anotar la dirección o ruta de acceso para llegar hasta tu archivo, ya que también lo necesitarás (**Si una vez creado el archivo le haces un clic, luego botón derecho, propiedades, podas ver la ubicación o ruta de acceso de este archivo**).



g) Escribir datos en el Archivo de Texto: Simplemente haces un doble clic en el archivo, o lo abres con el **Block de Notas** de Windows. Una vez abierto el archivo, procedes a escribir en cada línea lo que quieras y esperas leer desde tu programa Python. Con un "**ENTER**", das por terminada cada una de las líneas y continúas escribiendo en la siguiente. Ver imagen 4.

Puedes acceder al documento completo que contiene el instructivo de creación manual de archivos de Texto y el código básico Python como ejemplo para leer archivo. Bájalo de la nube y agrégalo a tu carpeta.

Documento:	Archivos_txt_Como_Usarlos.pdf

También deberías configurar el **IDE** de tu **Spyder** para que trabajes más cómodo. A continuación encontrarás el instructivo de configuración.

Documento:	Configurar_Spyder_para_Archivos_txt.pdf

Listo, has terminado. Ya puedes usar el archivo de texto desde tu programa Python. Y Ahora a Resolver ejercicios en la PC.

Nro	Enunciados con Lectura de Archivos - Creados Manualmente	Finaliza
62)	Este ejercicio tiene dos partes, realízalas. Si algo no comprendes o no recuerdas de lo visto en clase, pregunta. a)- Crea manualmente el archivo "personas.txt", el que debe contener un nombre en cada renglón. Abre el archivo de texto y copias los nombres que están a tu derecha (uno en cada renglón), grabas el archivo, cierras y listo. Recuerda poner el nombre al archivo. b)- Bajas de la nube el diagrama y código para leer el archivo de texto "personas.txt" (recién creado) que contiene listado de Nombres de personas (un nombre en cada renglón). Recuerda que puedes usar el instructivo. O simplemente usar lo que vimos en clase. Antonio Andrea Juan Susana Pedro Melanie Agustina Fernando Paula Victoria Iván Diagrama: Diagramas/08_Archivos_001_TXT_Plano_Lectura_001.png Programa: Codigo/08_Archivos_001_TXT_Plano_Lectura_001.txt Si no logras crear apropiadamente tu archivo "**personas.txt**", por esta vez, puedes bajar el archivo de la nube y probarlo, pero deberías usar el tuyo, así que practica creándolo. Archivo "**Personas.txt**": Archivos/Personas.txt	
63)	Este ejercicio consta de tres partes: La creación de un archivo y dos programas que usan el archivo. Desarrolla y explica cada una de las partes en forma independiente. En estos dos programas en particular No uses Listas. **Parte A:** Crear manualmente el archivo de texto "**Numeros_01.txt**", el que debe contener un número en cada línea. Estos números deberán ser enteros, mayores que 0 (cero) y menores o iguales que 100 (cien). No puede haber números repetidos. *Usa el listado de números que tienes a tu derecha.* **Parte B:** A continuación, hacer un programa Python que, lea el archivo "**Numeros_01.txt**" línea a línea. Y además (sin usar listas) debe calcular y mostrar el promedio de todos los números. Recuerda, siempre debes evitar dividir por Cero. **Parte C:** Modifica el programa anterior, para que busque y muestre el mayor número y su número de orden (posición) que tiene al ser leído en el archivo. **Importante,** el programa debe contener la siguiente función: "def ***MuestraNumero_ConTextoExplicativo(texto, num)***" Es decir, recibe como parámetro el número a mostrar, junto con la explicación de lo que es ese número. Por ejemplo, puede mostrar en el monitor *"El promedio es: 9"* 17 20 2 10 5 9 1 15 7 19 3 6 99	

Nro	Enunciados con Lectura de Archivos - Creados Manualmente	Finaliza
64)	Este ejercicio consta de varias partes: La creación de un archivo, dos programas que usan el archivo. Desarrolla y explica cada una de las partes en forma independiente. Acá si debes usar Listas.  **Parte A:** Crear manualmente el archivo de texto "**Numeros_02.txt**", el que debe contener un número en cada línea. Estos números deberán ser enteros, mayores que 0 (cero) y menores o iguales que 100 (cien). No puede haber números repetidos. *Puedes usar el archivo que creaste en un programa anterior o crearlo nuevamente y practicar*. **Parte B:** A continuación, crear un programa con Python, que sea capaz de leer el archivo "**Numeros_01.txt**", línea a línea y cargar en una lista todos los números leídos. Luego de cerrar el archivo, recorre el vector mostrando todos los elementos contenidos. **Parte C:** Modificar el programa anterior, para que luego de cerrar el archivo, recorra el vector para: - Calcular y mostrar el promedio de todos los elementos, y .. - Buscar/mostrar el mayor numero y posición que tiene en el vector. - Informa por monitor, que Números (de los comprendidos en el intervalo 1 - 100) NO SE ENCONTRARON en el archivo? (Por este apartado, pregúntaselo a tu Profesor o analiza el ejemplo - Acá usamos Posicionamiento Directo - Descarga el archivo de números que lo usaremos) IMPORTANTE, el programa debe usar las siguientes funciones: - BuscaMayor( ). - MuestraNumero_ConTextoExplicativo()	

Recuerda Como Leer Archivos

```
Camino = ""
Nombre = "Archi.txt"
Archivo_Fisico = Camino + Nombre
Archivo_Logico = open(Archivo_Fisico, "r")

Linea = Archivo_Logico.readline()
while Linea:
   print(Linea) # Registro completo
   Valor = Linea.strip().split(",")
   print(Valor) # Campo
   Linea = Archivo_Logico.readline()
Archivo_Logico.close() # Siempre cierro
# Aca se codifican otras acciones
```

Diagramas/08_Archivos_001_TXT_Plano_Lectura_001.png

Archivo usado:	Archivos/Numeros_02.txt
Programa Ejemplo:	Codigo/08_Archivos_001_TXT_Plano_Lectura_002_Posiciona_Directo.txt

Esto Debes Saberlo	Ventana de Comandos o Símbolo del sistema (Windows): Aunque a primera vista pueda parecer un poco intimidante, con sus letras blancas sobre fondo negro, te aseguro que es una herramienta increíblemente poderosa y versátil. Piensa en ella como el "cerebro" de tu computadora, donde puedes darle instrucciones directas y precisas. No te preocupes si no sabes por dónde empezar; todos lo hemos hecho. Esta guía es para ti, para que descubras paso a paso cómo dominar estos comandos aplicables y hacer que tu Windows haga exactamente lo que necesitas. ¡Prepárate para sentirte como un verdadero experto en tu propio sistema! Antes de la existencia de Windows (pregunta a tu profe) esta era prácticamente la única forma de manejar y trabajar en una PC. Bajar de la Nube: CMD_Ventana_de_Comandos_o_Simbolo_del_Sistema.pdf	
65)	En este programa usaremos nuevamente el archivo "**Numeros_01.txt**" recientemente creado. Este es un solo ejercicio, ejecuta paso a paso las cosas solicitadas: **a)-** Define un Vector (Lista) que contenga 100 elementos, y cada elemento debe estar inicializado en 0 (cero). **b)-** A continuación leer el archivo **"Numeros_01.txt",** que contiene en cada registro un número entero positivo mayor o igual que cero y menor que cien. El número que lees en cada registro del archivo, te indicará en que posición del vector debes poner un 1 (uno). **c)-** Una vez hayas terminado con la lectura del archivo y lo haya cerrado, recorre el vector e informa por pantalla, la posición de los elementos que están en 1 (uno). **d)-** La respuesta a la pregunta que se formulará a continuación, escríbela en tu carpeta y también al pie del código, como un comentario de tu programa, así siempre podrás	

Nro	Enunciados con Lectura de Archivos - Creados Manualmente	Finaliza

leerla cuando se te pregunte durante la corrección.
Hay alguna similitud entre el informe que realizarás por el monitor y los números contenidos en el archivo? Por que? Explica minuciosamente.

66) La respuesta a este ejercicio, escríbela en tu carpeta y también al pie del código (como un comentario en tu programa), así siempre podrás leerla cuando se te pregunte durante la corrección.

Tu debes:

- Si hubiere algún comando que no conoces, investiga y anota en tu carpeta.
- Analizar detalladamente y explicar que hace y como funciona el segmento de código?
- Que uso puedes darle? plantea un enunciado coherente al ejercicio.

Completa el final!

Si algo no comprendes pregunta al profesor. Luego con tus palabras completas el ejercicio.
A continuación tienes la dirección donde bajar el código completo del programa ejemplo y el archivo que debes usar:

```
Cantidad = 100
V=[ 0 for i in range(Cantidad)]  # El vector será desde posición 0 a la 99
Camino = ""  # Si está vacía, el archivo debe estar junto con tu programa
Nombre = "ArchiVector_01.txt"
Archivo_Fisico = Camino + Nombre
Archivo_Logico = open(Archivo_Fisico, "r")

Linea = Archivo_Logico.readline()
while Linea: # Mientras se haya leido una Linia del Archivo
   print(Linea)
   NroSensor, Valor = Linea.strip().split(",")
   if NroSensor.isdigit() and Valor.isdigit():
      NroSensor = int(NroSensor)
      Valor = int(Valor)
      print(f"Campos Separados y Convertidos a Numero: {NroSensor} - {Valor}")
      if(Cantidad > NroSensor and NroSensor > 0):  # Que controla Acá?
            V[NroSensor - 1] += 1
      else: print (f"Numero de Sensor Invalido: {NroSensor}.")
   else: print("\nAlgunos Datos Son Inconrrectos.")
   print("=======================================")
   Linea = Archivo_Logico.readline()
Archivo_Logico.close() # SIEMPRE Cierro el archivo (MUY IMPORTANTE)
```

Código Completo:	Codigo/08_Archivos_003_TXT_Plano_Lectura_004_Posicionamiento_Directo_Vector.txt
Archivo:	Archivos/ArchiVector_01.txt

67) **Recomendación:** Usar Posicionamiento Directo en Vectores. Este ejercicio consta de dos partes. Primero, la creación de un archivo y segundo, un programa que usa el archivo al que podemos dividirlo en varias partes. Recuerda, acá si **debes usar Listas**.

Primera Parte: Crear manualmente el archivo de texto "**Numeros_02.txt**", el que debe contener un número en cada línea. Estos números deberán ser enteros, mayores o iguales que 0 (cero) y menores que 100 (cien). SI puede haber números repetidos, tantos como quieras (agrega números repetidos en el archivo, en forma desordenada, pero siempre respetando la existencia de un solo número en cada línea).

```
17
20
2
10
5
9

1
15
7
19
3
6
17
```

Segunda Parte: Hacer una programa que lea el archivo recién creado "**Numeros_02.txt**" y realice las siguientes acciones:

a)- Informe todos los números posibles (cualquiera entre el cero y el noventa y nueve, ambos incluidos) Si el número esta incluido en el archivo y cuantas veces se repitió cada número.

b)- Muestre un listado, ordenado de menor a mayor, (Si usas Posicionamiento directo, no es necesario que sepas ordenar el vector) con los números que se encontraron en el archivo, pero elimina (no muestres) las repeticiones.

```
20
10
5
9
1
20
2
```

c)- Informar cual o cuales de los números contenidos en el intervalo (0 - 99), no se encuentran cargados en el archivo.

Donde corresponda, usar las siguientes funciones:
- MuestraNumeroConExplicacion( )
- MuestraCantidadDeRepeticiones( )
- InformaNumerosFaltantes( )

```
10
5
9
1
15
7
19
3
6
17
```

Te dejo un diagrama de flujo, fíjate si hace algo de lo solicitado en el enunciado:

Diagrama:	Diagramas/Archivo_Vector_Cuenta_Numeros_Repetidos_01.png

También tienes algunos programas de ejemplo, que te pueden ayudar. Analízalos

Codigo/02_Intervalos_001_Verificar_Pertenencia_01.txt
Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Cuenta_Repeticiones_A.txt
Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Cuenta_Repeticiones_B.txt
Codigo/07_Listas_Vectores_016_Posicionamiento_Directo_01_Elimina_Repeticiones_C.txt

Nro	Enunciados con Lectura de Archivos - Creados Manualmente	Finaliza

68) Este ejercicio tiene cuatro partes, resuelve cada parte como un ejercicio independiente.

a)- Crear manualmente el archivo "**Registro_01.txt**", el cual contiene 10 (diez) registros (si hubiere repeticiones ignóralas):
Simplemente cópialos en el archivo de texto, graba y cierra. Cuida de no agregar un "**Enter**" adicional o cualquier otra cosa.

```
01,Primera Línea
02,Segunda Línea
03,Tercera Línea
04,Cuarta Línea
05,Quinta Línea
02,Línea Repetida
06,Sexta Línea
07,Séptima Línea
08,Octava Línea
02,Línea Repetida
09,Novena Línea
10,Décima Línea
```

b)- Ahora usaremos el archivo recién creado "**Registro_01.txt**". Tu programa deberá leer este archivo y mostrar en el monitor, todos los registros leídos, tal cual se leen.

Repasar: **SEPARAR LOS DATOS Leídos en UNA SOLA Línea**.

c)- Modificar el programa anterior, para que muestre los registros, pero con los campos separados. Recuerda que el separador de campo es la "," (coma) y cada registro tiene dos campos:

Numero	Texto

d)- Modificar el ultimo programa, para que los datos de los 10 registros, se almacenen en un vector, y utiliza el campo "Numero" como índice del vector (es el campo clave) y guarda el campo "Texto)". Una vez finalizada la carga del vector y cerrado el archivo, muestra todos los elementos contenidos en el vector. Recuerda que si encuentras repetido el "Numero" (campo clave) no debes cargar el texto ya que seria una repetición. Informa al finalizar cuantas repeticiones encontraste

Si no logras crear apropiadamente tu archivo "**Registro_01.txt**", por esta vez, puedes bajarlo de la nube y probarlo, pero debes usar el tuyo, así que practica.

Archivo:	Archivos/Registro_01.txt
Repasa, como Separar los Campos:	Codigo/08_Archivos_004_TXT_000_Separo_Datos_Leidos.txt

IMPORTANTE: piensa, reflexiona y escribe en tu carpeta:
- Es necesario siempre eliminar los espacios en blanco al separar los campos de un registro?
- Siempre hay que convertir los campos a numero entero?

69) Este ejercicio tiene cuatro partes, resuelve cada parte como un ejercicio independiente.

a)- Crea manualmente el archivo "**Registro_02.txt**".
Simplemente cópia los datos en el archivo de texto, graba y cierra. Cuida de no agregar un "**Enter**" adicional o cualquier otra cosa.

```
3,5,9,1
1,3,3,2
0,0,0,0
2,3,1,5
4,2,5,1
7,5,2,5
3,7,5,9
```

b)- Ahora usaremos el archivo recién creado "**Registro_02.txt**". Tu programa deberá leer este archivo y mostrar en el monitor, todos los registros leídos, muéstralos tal cual se leen.

Repasar: **SEPARAR LOS DATOS Leídos en UNA SOLA Línea**.

c)- Modificar el programa anterior, para que muestre los registros, pero con los campos separados. con el siguiente formato:

Campo 1: Valor 1	Campo 2: Valor 2	Campo 3: Valor 3	Campo 4: Valor 4

Recuerda que el separador de campo es la "," (coma) y cada registro tiene cuatro campos
d)- Modifica el último programa para que calcule y muestre el promedio de cada registro. Cuidado con el registro que no contiene valores (todos los campos en cero), este no debe ser procesado.

Si no logras crear apropiadamente tu archivo "**Registro_02.txt**", esta vez, puedes bajarlo de la nube y probarlo, pero debes usar el tuyo, así que practica creándolo.

Archivo:	Archivos/Registro_02.txt
Repasa como Separar los Campos:	Codigo/08_Archivos_003_TXT_Plano_Lectura_003_Registro.txt

IMPORTANTE: piensa, reflexiona y escribe en tu carpeta:
- Es necesario siempre eliminar los espacios en blanco al separar los campos de un registro?

Nro	Enunciados con Lectura de Archivos - Creados Manualmente	Finaliza
	- Siempre hay que convertir los campos a numero entero? - Siempre debes evitar dividir por Cero ?	
70)	En un colegio, para realizar un sorteo, se entregan números a cada uno de los participantes, para este fin, se utiliza un talonario de 100 números (001 al 100). Cada participante elige el número que siente que le traerá suerte, por este motivo los números no se entregaron correlativamente y quedan algunos sin entregar. El encargado de repartir los números, confecciona un listado para mantener un registro de quien tiene cada número. **Tu Debes hacer:** **Primero,** Crear manualmente el archivo "**Rifas_01.txt**" el que en cada registro contiene dos campos: Numero de rifa y Nombre del dueño. Número | Nombre Puedes simplemente crear el archivo copiando el listado que se encuentra a la derecha, y luego en tu programa: **a)-** Define e inicializa un vector de 100 posiciones, el que en cada posición tendrá únicamente un nombre. **b)-** Lee el archivo y carga en el vector, los nombres, pero cuidado, el número de rifa, te indicará en que posición del vector guardarás el nombre del propietario del número. **c)-** Crea un menú, que te permitirá realizar las siguientes acciones: **- 0 Termina Programa.** **- 1 Listado que muestra cada número y nombre del Poseedor (si fué vendido).** **- 2 Confecciona un listado con los números no entregados.** **- 3 Informa cuantos números se entregaron.** **- 4 Cuantos números quedan disponibles para entregar.** El menú deberá usar un color distinto para cada opción. 7,Alvaro 85,Maite 12,Marcela 34,Margarita 1,Teresa 99,Andrés 54,Angel 45,Horacio 17,Roxana 4,Santiago 36,Paula 21,Benjamín 29,Alberto 62,Beltran 19,Juan 81,Micael 43,Tatiana 32,Florencia 5,Alida 11,Augusto 16,Victorio 72,Gaston 68,Cintia	
71)	En una carrera, se controlaron y registraron los tiempos de todos los atletas al finalizar cada una de las etapas. Quedando toda la información contenida en el archivo "**Competencia_01.txt**". Cada registro contiene los siguientes campos: Tramo | Metros | NumCorredor | Tiempo Descripción de los campos:  **Tramo:** Representa el número de tramo y en total son 10 (diez) numerados de uno en uno. **Metros:** Es tamaño del tramo, medido en metros. **NumCorredor:** Número de corredor. En la competencia participaron 25, numerados de uno en uno. **Tiempo:** Tiempo, medido en segundos, que el corredor tardó en recorrer el tramo completo. El separador de campo usado en cada registro es "/" Realiza un programa que, procese los datos del archivo dado y encuentre: **a)** Numero del corredor que gano la competencia (El que tenga la menor Suma de todos sus tiempos). **b)** El corredor más lento del evento (la mayor Suma de todos sus tiempos).. **c)** Cuantos Kilómetros corrieron los competidores en total. **d)** Cual fue la velocidad media (Km/h) del ganador de la competencia (Suma los Kilómetros / Suma de horas). **Comentarios Importantes**: - El archivo esta ordenado por tramo y luego por Número de corredor. Analízalo?. - Puedes usar un vector donde sumaras los tiempos de cada corredor. - En una variable entera, podrás sumar las distancias recorridas por cada corredor (todos corren igual distancia) Puedes descargar el archivo o crearlo tu mismo: Archivos/Competencia_01.txt Hay casos en que el caractér "/"(Barra diagonal o Slash) leído o grabado en registros de tu archivo como separador de campo, es interpretado erróneamente, generando un problema en tu programa. En estos casos	

Nro	Enunciados con Lectura de Archivos - Creados Manualmente	Finaliza
	simplemente cámbialo por una coma "," (en el archivo o en tu registro). Pregunta a tu profesor como debes proceder. :-)	

Un extra a saber, cuando abres archivos de Texto para Lectura (encoding="utf-8)

Hasta el momento, se ha explicado (y siempre funcionó) que a un archivo podemos abrirlo para lectura así:

```
Archico_Logico = open( Archivo Fisico, "r")
```

Esto asume que tu archivo ha sido escrito con las reglas definidas por "utf-8", ahora bien, si quieres evitar posibles fallos o errores de lectura y asegurarte que realmente se lo lea e interprete con esta norma, puedes abrir tus archivos de texto agregando el parámetro: **encoding="utf-8"**. Es decir ábrelo así:

```
Archico_Logico = open( Archivo Fisico, "r", encoding="utf-8")
```

Explicación: El parámetro encoding="utf-8" indica a tu programa cómo interpretar los caracteres del archivo (la "codificación de texto").

Al poner ese parámetro, le dices a Python: "lee este archivo como texto en UTF-8".
Que resulta ser el estándar actual en la mayoría de los archivos y sistemas.
Y por las dudas, estarás garantizando que se lean bien los acentos (á, é), eñes (ñ) y símbolos especiales.

Y si NO PONES ese parámetro, Si tu archivo solo tiene letras normales (a-z, A-Z) y números, funcionará igual. Pero si el archivo tiene acentos, ñ o símbolos raros, **podría dar el error** (UnicodeDecodeError) o mostrarte caracteres extraños.

Crear Archivos con Programas Python - (Clase Teórica: xx/xx2026).

Trabajo explicado y desarrollado en clase. Hasta este punto, los archivos de texto los creaste manualmente, según se indicó en el instructivo correspondiente.

Ahora deberás escribir en la PC, el programa desarrollado en el aula, que realmente sea capaz de crear (escribir) el Archivo usado en los Ejercicios anteriores (Es el mismo archivo de texto que antes creaste manualmente, y contiene un número en cada línea, pero sin repeticiones).
Una vez hayas codificado el programa, verifica que funcione correctamente, visualizando el contenido del nuevo archivo.

Nro	Trabajo con Archivos - Lectura y Creación - Resumen de Clase	Finaliza
72)	Para la clase teórica de esta semana, deberás traer impreso a clase el documento "**Resumen_Archivos.pdf**". Lo usaremos frecuentemente. **Inclúyelo en la Carpeta con el número de Ejercicio Correspondiente**. Documento: Resumen_Archivos.pdf	

Modos de Apertura de Archivos:

r	Lectura
w	Escritura
a+	Podrás leer o escribir en el archivo. Si el archivo aún no existe, se crea uno nuevo. Si escribes, el puntero se posiciona al final del archivo. El texto recién escrito se agregará al final, a continuación de los datos escritos anteriormente.
	Hay otros modos de apertura - Por ahora solo veremos estos.

Lugar donde se encuentra y nombre (Cualquier Modo)	**Camino = ""** # Si dejo en blanco, Python entiende que es la carpeta de trabajo **Nombre = "Personas.txt"** # Nombre Físico del archivo (el que veo en el disco) **Archivo_Fisico = Camino + Nombre** # Armo Dirección más nombre de Archivo La variable **Archivo_Fisico**, contiene toda la información referente al archivo (dentro de la PC)

Apertura para Lectura o Escritura (Modo 01)	**Código Completo:** Codigo/08_Archivos_001_TXT_Plano_Lectura_001.txt **Archivo_Logico = open(Archivo_Fisico, "'MODO APERTURA")** Acá los comandos que Usaré mientras trabajo con el archivo Acá los comandos que Usaré mientras trabajo con el archivo La variable **Archivo_Logico**, Nos permite Manejar y decir que hacer con el **Archivo_Fisico**
Apertura para Lectura o Escritura (Modo 02)	**Código Completo:** Codigo/08_Archivos_002_TXT_Plano_Escritura_002.txt **with open(Archivo_Fisico, 'MODO APERTURA') as Archivo_Logico:** Acá los comandos que Usaré mientras trabajo con el archivo Acá los comandos que Usaré mientras trabajo con el archivo La variable **Archivo_Logico**, Nos permite Manejar y decir que hacer con el **Archivo_Fisico**
Proceso de Lectura de Archivos (Modo 01) **Registro Completo**	**Código Completo:** Codigo/08_Archivos_003_TXT_Plano_Lectura_001.txt **Linea = Archivo_Logico.readline()** while Linea: # Mientras Línea contenga algo print(Linea) #El archivo ya Tiene un \n al final de cada línea **Linea = Archivo_Logico.readline()** Realiza la lectura del archivo y el registro (contenido en la variable **Linea**), es una cadena de Caracteres. hay que tener cuidado antes de usarla y ver que partes se usaran.

Proceso de Lectura de Archivos (Modo 02) **Separando Campos**	**Programa:** Codigo/08_Archivos_003_TXT_Plano_Lectura_003_Registro.txt
	Linea = Archivo_Logico.readline()
	while Linea:
	Nro_01, Nro_02, Nro_03, Nro_04 = Linea.strip().split(",") #Separa Campos
	print("Campos Separados: ",Nro_01, Nro_02, Nro_03, Nro_04)
	# Acá hago lo que se deba hacer con cada uno de los campos
	Linea = Archivo_Logico.readline() # Leo nuevamente y regreso al While

Proceso de Escritura El Registro Completo (Modo 01)	**Programa:** Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_001_Registro.txt **Nombre = input("Nombre: ")** **while Nombre:** #Mientras haya leído Nombre **Registro = Lee_Datos(Nombre)** **Archivo_Logico.write(Registro)** **Nombre = input("Nombre: ")** **Función "Lee_Datos()"** def Lee_Datos(Nom): N1 = input("Primera Nota: ") N2 = input("Segunda Nota: ") N3 = input("Tercera Nota: ") Linea = Nom + "," + N1 + "," + N2 + "," + N3 + "\n" return Linea
Control de Lectura	# Cuando estoy leyendo Cadena de Caracteres y Quiero saber si se leyó algo **if not Linea:** **# Pregunto: Esta Vacía?** print("Cuando NO SE PUDO LEER")
Finaliza y no hay nada que hacer	**Archivo_Logico.close()** # Siempre cierro el archivo

73) **Resuelto y explicado en Clase.** Hacer un programa en el que se cree el archivo "**Nombres_00.txt**" y cada registro contenga **Nombre y Edad** de los alumnos de un curso. Los datos deben ser leídos desde el teclado, e inmediatamente grabados en el archivo. La carga de datos (lectura) terminará cuando se ingrese un nombre nulo (Solo tecla **Enter**). Usar como separador de campo el caractér "/" (Barra diagonal o Slash)

Diagrama:	Diagramas/08_Archivos_001_TXT_Plano_Grabacion_000.png
Código:	Codigo/08_Archivos_001_TXT_Plano_Grabacion_000.txt

Hay casos en que el caractér "/" (Barra diagonal o Slash) leído o grabado en registros de tu archivo como separador de campo, es interpretado erróneamente, generando un problema en tu programa. En estos casos simplemente cámbialo por una coma "," (en el archivo o en tu registro). Pregunta a tu profesor como debes proceder. **:-)**

Repasa como separar los campos en una cadena (registro leído) hay más de una forma.

Nro	Enunciados con Lectura y Creación de archivos desde Python	Finaliza

74) Este Ejercicio tiene dos partes, resuélvelas como programas separados. Puedes usar como guía, el documento "**Resumen_Archivos.pdf**" que has impreso, o los apuntes de clase, cuando resolvimos un ejercicio similar.

a)- A continuación, te dejo un enunciado y el problema resuelto. Analízalo y escribe en tu carpeta que hace cada línea.

Hacer un programa (se usa **modo escritura 02**) que cree (escriba) el archivo "**Números_01.txt**", que contiene un único número en cada registro (renglón). Indicar que termina la carga de datos (números) dando un "**Enter**" sin haber ingresado un número.

Controlar que lo que se ingrese y grabe en el archivo sean números.

Proceso de Escritura. (Ejemplo con Modo 02)

```
Camino = "../Archivos/"
Nombre_txt = "Numeros_03.txt" # Nombre Físico del archivo de Texto
Archivo_Fisico = Camino + Nombre_txt # Datos completos del archivo

with open( Archivo_Fisico, 'w+') as Archivo_Logico:
   Numero = input("Ingrese un Numero (solo enter y terminará): ")
   while Numero: #Mientras se haya leido un Numero
      if Numero.isdigit(): # Verifico que se haya leído un numero
         Registro = Numero + "\n" # Armo el registro
         Archivo_Logico.write(Registro) # Grabo Registro
      else: print("Error, re ingrece Numero.")
      Numero = input("Ingrese un Numero (solo 'enter' y terminará): ")

Archivo_Logico.close()
# Aca puedes codificar otras acciones
```

Creación del Primer Archivo

Esto lo vimos en clase, pero igual te dejo el código para analizar. Ante cualquier duda consulta. A continuación el código completo para que lo bajes de la Nube.

Código:	Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_001_Registro_Un_Solo_Campo.txt

b)- Modifica el problema anterior, para que también controle que los números que se graban en el archivo, sean mayores que cero y menores o iguales que 100. En caso de ingresar un valor incorrecto, avisar del error, y leer nuevamente el número.

75) Este Ejercicio tiene cuatro partes, resuélvelas como programas separados e independientes. Puedes usar como guía, el documento "**Resumen_Archivos.pdf**" que has impreso, o los apuntes de clase, cuando resolvimos el ejercicio.

a)- Hacer un programa que escriba (Crea) el Archivo "**Nombres_01.txt**". Este archivo debe contener en cada registro un Nombre de persona. Cada nombre debe ser único e irrepetible.

b)- Hacer un nuevo programa que, lea registro a registro el archivo recién creado "**Nombres_01.txt**", muestre por monitor el nombre (contenido en el registro) y el usuario escriba (ingrese por teclado) el género correspondiente a dicho nombre (Palabra: Hombre o Mujer). Acto seguido, grabe en el nuevo archivo "**Nombres_02.txt**" estos dos campos:

Género	Nombre

Verificar que el único dato ingresado sea la palabras solicitas (Hombre o Mujer). Recuerda usar como separador de campo la coma.

c)- Hacer un nuevo programa que lea (registro a registro) el archivo "**Nombres_02.txt**" y automáticamente cambie en cada registro el campo género, las palabras "Hombre", "Mujer"; por un numero 1 o 2 según corresponda. Es decir, si en el campo género dice: "Hombre" lo cambias por 1 y si en el campo dice "Mujer" lo cambias por el numero 2. A cada registro que modifiques, lo deberás grabar en el archivo "**Nombres_03.txt**".

d)- Para finalizar hacer un programa que lea y liste por monitor, el contenido del archivo "**Nombres_03.txt**". Un registro por línea, y al finalizar informe cuantos registros contiene el archivo.

El formato que deberá tener el listado, puedes visualizarlo en el recuadro de la derecha:

```
Nombre: XXXXX
Género: ZZZZ
========================
Nombre: XXXXX
Género: ZZZZ
========================
Fin del listado
Se mostraron YYY registros
```

Nro	Enunciados con Lectura y Creación de archivos desde Python	Finaliza

76) Hacer un programa que sea capaz de crear (escribir) el archivo "**Alumnos_01.txt**" que contiene en cada registro, el nombre de un alumno y tres notas correspondientes a las evaluaciones del periodo de estudio:

Nombre	Nota_01	Nota_02	Nota_03

Información Importante: controlar que se cargue correctamente.

- Las notas de cada alumno, son números enteros comprendidos entre 1 y 10.

Al grabar en el archivo, usar como separador entre los campos la coma ",". Quizás estos ejemplos te sirvan, analízalos:

Ejemplo 01:	Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_000_Hace_Cadena_Char_01.txt
Ejemplo 02:	Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_000_Hace_Cadena_Char_02.txt
Ejemplo 03:	Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_000_Hace_Cadena_Char_03.txt

77) En un instituto de enseñanza, se dispone del archivo "**Alumnos_01.txt**" (creado anteriormente) que contiene en cada registro, el nombre de un alumno y tres notas correspondientes a las evaluaciones del periodo de estudio:

Nombre	Nota_01	Nota_02	Nota_03

Se pide que realices un programa **(Recuerda puedes usar listas)** que permita leer el archivo, calcular y mostrar la siguiente información:

a)	El promedio de cada uno de los alumnos. Que deberá mostrarse como número real (float) con 2 decimales. Recuerda, siempre debes evitar dividir por Cero.
b)	Informar el nombre y promedio del alumno que tiene el promedio individual mayor.
c)	El promedio del curso, correspondiente a cada una de las 3 evaluaciones (promedio de cada columna, correspondiente a una evaluación). Que deberá mostrarse como número real (float) con 2 decimales.
d)	Promedio general del curso (el promedio de TODAS las notas de TODOS los alumnos). Que deberá mostrarse como número real (float) con 2 decimales.

El Programa deberá usar las siguientes funciones:

-	CalculaPromedio( ) Recibe la suma de elementos y el número por el que deberá dividir
-	PromedioDeUnAlumno( ) Que debe Recibir 3 notas y retorna el promedio
-	MuestraNumeroConTextoExplicativo( )
	Esta última función deberá ser usada en los cuatro puntos solicitados (a, b, c, y d).

IMPORTANTE: Dispones del Archivo "**Alumnos_01.txt**", del programa que lo crea automáticamente con notas cargadas al Azahar, y algunos programas de ejemplo.

Puedes Bajarlo desde:	Codigo/08_Archivos_004_TXT_Alumnos_01.rar
Programa de consulta, crea archivo:	Codigo/08_Archivos_003_TXT_Plano_Z_Escritura_001_Registro_Varios_Campos.txt

Al grabar en el archivo, usar como separador entre los campos la coma ","

Nro	**Enunciados** - Continuamos con el tema	Finaliza

78) **Analizar el programa ejemplo:**

Analiza y explica este programa en tu carpeta, escribe detalladamente línea por línea lo que hace. Es importante que entiendas, por que usaremos esta lógica con frecuencia.

Programa:	Codigo/*08_Archivos_004_TXT_004_Alumnos_01_Grabo_Alumnos_02_con_Promedio.txt*
Que también esta contenido en paquete de programas, ver mas abajo.	

Cuyo enunciado dice: Tomando el mismo Archivo "**Alumnos_01.txt**" usado anteriormente, se pide crear el archivo "**Alumnos_02.txt**", que debe contener además del **nombre** y las **3 notas**, un nuevo campo "**Prom**", que contiene el promedio calculado con las 3 notas mencionadas.

Nombre	Nota_01	Nota_02	Nota_03	Prom

Puedes Bajar paquete de programas desde: Codigo/08_Archivos_004_TXT_Alumnos_01.rar

Nro	Enunciados - Continuamos con el tema	Finaliza

79)

Parte A: En esta primera parte, realiza el programa realice esta acción. En el cine "**VeoVeo**", durante el ingreso a la sala, los espectadores exhiben el comprobante de compra de su boleto, previamente abonado por Internet. En cada comprobante de pago, se visualizan claramente tres datos:

Número de Boleto	Sección a la que pertenece la butaca	Nombre de quien realizó la reserva

A los espectadores que ingresan a la sala se les controlan los comprobantes y manualmente se registran en el sistema informático, generando el archivo "**Boletos_01.txt**", que contiene los tres campos mencionados.

Datos Importantes: En cada registro del archivo, se usa como separador de campo a la coma ",". La sala dispone de 900 butacas numeradas desde el 1 al 900, divididas en tres zonas de 300 butacas cada una, identificadas como:
Zona A: Butacas numeradas desde el 1 al 300,
Zona B: Butacas numeradas desde el 301 al 600 y
Zona C: Butacas numeradas desde el 601 al 900.

Parte B: Usando la I.A. de tu elección, recrea el archivo mencionado en la primera parte. Recuerda adjuntar el **promp** en tu carpeta, a continuación del enunciado y resguarda el archivo para ser usado al momento de la corrección de tu programa. **Recuerda incluir nombres con Acentos en el archivo.**

Parte C: Una vez terminado el espectáculo, y usando el archivo recién escrito (escrito en la parte A o parte B) el sistema debe generar los siguientes informes:

Informe A	Informe B
- Listado con número de butaca, zona a la que pertenece (A, B o C) y nombre de quien realizó el Pago (de toda la Sala). - Listado con los números de Butaca disponibles en las Zonas A y B de la sala (que no Asistieron o no fueron vendidas)	- Listado con los número de butaca y nombre de quien realiza el pago, pero solamente de los espectadores de la zona A. - Listado con los números de Butaca disponibles en las Zonas B y C de la sala (que no Asistieron o no fueron vendidas)

OPCIONAL

Ejemplos. Para Quien Quiera Aprender, que Aprenda.

Borra una Carpeta Vacía	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_007_Borro_Carpeta_Vacia.txt
Borra Carpetas No Vacías	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_008_Borro_Carpeta_NO_VACIA.txt
Borra un tipo de Archivos	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_005_Borro_Tipo_Archivo.txt
Borra un Archivo Especifico	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_003_Borro_Archivo_Especifico.txt
Busca un Archivo en la PC	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_002_Busco_Archivo.txt
Cambia Nombre del Archivo	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_009_Cambia_Nombre_Archivo.txt
Cambiar Nombre de Archivo	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_012_Renombra_Un_Archivo.txt
Copia Todos los Archivos	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_011_Copia_Todos_los_Archivos.txt
Copia Un Archivo	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_010_Copia_Archivo.txt
Crea Nueva Carpeta	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_006_Crea_Carpeta_Vacia.txt
Lista Archivos de una Carpeta	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_004_Listado_Archivos_en_Carpeta.txt
Verifica Existencia Archivo 01	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_001_Verifica_Existencia_01.txt
Verifica Existencia Archivo 02	Codigo/08_Archivos_005_TXT_Plano_ESPECIAL_001_Verifica_Existencia_02.txt

80)

Aprende Esto.

Prompt, que Cosa es?

El Prompt es el texto que tú escribes para solicitar a una IA, exactamente lo que quieres que haga. El Prompt es el conjunto de palabras que componen la instrucción, que le da el contexto a la tarea que debe realizar la Inteligencia Artificial.

La calidad de la respuesta que te entregue la IA, dependerá de qué tan bueno sea tu prompt. Si el prompt es vago o poco claro, la respuesta será vaga o ambigua. Si el prompt es específico y creativo, la respuesta será genial.

¡Es la clave para desbloquear el potencial de la IA!

FUNCIONES CON PARÁMETROS OMITIDOS (Clase Teórica xx/xx/2026)

81) Clase Teórica

Los parámetros omitidos u opcionales, son aquellos que no se proporcionan en la llamada de una función. En Python, las funciones pueden tener parámetros opcionales que, deben tener un valor predeterminado. Si al usar la función, un valor no se proporciona para un parámetro opcional, la función utilizará el valor predeterminado. Esto significa que el parámetro se omite en la llamada y no es necesario especificarlo.

Recuerda que los parámetros, obligatoriamente, solo se pueden omitir de atrás para adelante. Es decir si solo falta uno, Python enmendará que falta el último. y si faltan dos parámetros, entenderá que faltan el ultimo y ante ultimo. etc. Ahora analiza este ejemplo:

Puedes Bajarlo desde: Codigo/09_Funciones_005_Parametros_Omitidos_001_Ej_01_Saludando.txt

PROGRAMA PYTHON: Saludando.

```
"""
Creado 24/04/2021
Saludando - Ejemplo 01
@author: Horacio
"""
def saludar(nombre, saludo="Hola"):
mensaje = saludo + ", " + nombre + "!"
print(mensaje)
print("------------------------------------------------------------")
print(" Saludando - Ejemplo 01")
print("------------------------------------------------------------")
# Ejemplo de uso:
saludar("Juan") # Uso el parámetro Omitido
saludar("María", "Buenos días") #Dato por parámetro Omitido
# Termina Programa
print("\n------------------------------------------------------")
print("Programa Terminado.")
```

Esto Veras En El Monitor

```
------------------------------------------------------
 Saludando - Ejemplo 01
------------------------------------------------------
Hola, Juan!
Buenos días, María!
------------------------------------------------------
Programa Terminado.
```

A continuación, el segundo ejemplo, en el que se usan valores numéricos. La función calcula el precio final con y sin descuento.

Puedes Bajarlo desde: Codigo/09_Funciones_005_Parametros_Omitidos_001_Ej_02_Precio_Descuento.txt

PROGRAMA PYTHON: Calculo descuento opcional

```
def Precio(total, descuento=0):
    return (total - (total * descuento / 100))
print("------------------------------------------------------------")
print(" Parámetros Omitido - Ejemplo 02")
print("------------------------------------------------------------")
# Inicia Programa
P = Precio(1000) # Usamos la función sin parámetro opcional
print("El precio final con descuento es:", P)
# Llamamos la función especificando el parámetro opcional
P = Precio(1000, 10) # Función con parámetro opcional
```

ESTO VERAS EN EL MONITOR

```
------------------------------------------------------
 Parámetro Omitido - Ejemplo 02
------------------------------------------------------

------------------------------------------------------
El precio final con descuento es: 1000.0
El precio final con descuento es: 900.0

------------------------------------------------------
Programa Terminado.
```

Clase Teórica

A Tener en Cuenta: Es importante que **NO CONFUNDIR** (cuando estas definiendo una función) los *Parámetros Omitidos* con los Parámetros o **Argumentos Posicionales Variables** (Tema que por ahora no estudiaremos).

Parámetros Omitidos: Como ya vimos, los parámetros Omitidos se usan (al definir la función) asignando un valor alternativo (por la dudas) si al llamar la función, se omite un valor (argumento).

```
def Suma( A, B = 0):
    return (A + B)
```

Parámetros o **Argumentos Posicionales Variables**: Cuando ves un asterisco simple "*" delante de un nombre de parámetro en la definición de una función, como en, significa que ese parámetro es una colección (conjunto) de argumentos que se recibirán, por ejemplo, una *tupla*. Podrías encontrar uno o dos asteriscos, que significarán cosas diferentes. Por ejemplo:

```
def sumar_elementos(nombre_operacion, *numeros):
    Aca escribes el cuerpo de la Función
```

Pero como ya se había mencionado un poco mas arriba, este tipo de parámetros no será estudiado por ahora

Nro	Enunciados con Funciones y Parámetros Omitidos	Finaliza
82)	**Parte A:** Realiza un programa que contenga una función a la que puedas entregarle un máximo de cinco parámetros, y te permita omitir (no entregarle) uno, dos o tres de esos	

Nro	Enunciados con Funciones y Parámetros Omitidos	Finaliza
	parámetros. Los parámetros serán números y la función siempre calcula la suma de los valores recibidos. **Parte B:** **(parte opcional, pero vale puntos)** Con la IA de tu preferencia, genera una imagen representativa de un parámetro omitido en una Función Python. Pon en funcionamiento tu imaginación. Pega la imagen y tu Prompt usado en tu carpeta y recuerda guardarlo en un Archivo txt, así lo puedes usar nuevamente cuando el profesor te lo solicite.	
83)	Crear y usar una Función a la que le entregues un nombre de persona y lo guarde en una lista existente, pero si llamas a la función sin entregarles los parámetros requeridos (La lista creada y el nombre que debe guardar en ella), la función descarte la lista que estabas usando y cree una lista nueva (vacía) para luego continuar usándola. Que pasa con la lista Vieja? Acá te dejo un ejemplo funcionando, pero usa números: **Bajar código:** Codigo/09_Funciones_005_Parametros_Omitidos_002_Ej_01_Agrega_o_Crea_Lista.txt	

Nro	Cosas Interesantes (Parte III)	Finaliza
	Python, versión instalada: Como saber (con programa Python) que versión tienes instalada en la PC que estás trabajando. Puedes bajar el programa de la nube y úsalo. Baja el código de acá: Codigo/90_Interesante_001_Muestra_Version_Python.txt	
pip	PIP, El Gestor de Paquetes Python: Si Todavía no lo has usado, acá te dejo la documentación necesaria para que puedas manejar las Librerías/Paquetes/Módulos en tu entorno. Baja el documento de la Nube: Más Información: PIP_Gestor_de_Paquetes.pdf	**pip**
	Librerías Parte 1 - Cuales están Instaladas en tu PC: (Solo para Investigadores). Rápido vistazo a las librerías instaladas en este equipo. Generar listado con nombre de librerías y versión. Precedencias y más. Bajar código desde acá: Codigo/90_Interesante_001_Librerias_01_Instaladas_A_Mustra_Nombre_y_Version.txt Más Información: Librerias_que_he_Instalado_en_mi_Sistema.pdf	

A continuación, cosas Importantes:

Antes de Finalizar esta guía de Trabajos Prácticos, debes recordar y tener bien claro que, de cada 10 alumnos....

01000011 01101001 01101110 01100011 01101111 00100000 01000101 01010011 00100000 01011001 00100000 01010011 01001001 01000101 01001101 01010000 01010010 01000101 00100000 01010011 01000101 01010010 01000001 00100000 01101100 01100001 00100000 01001101 01001001 01010100 01000001 01000100