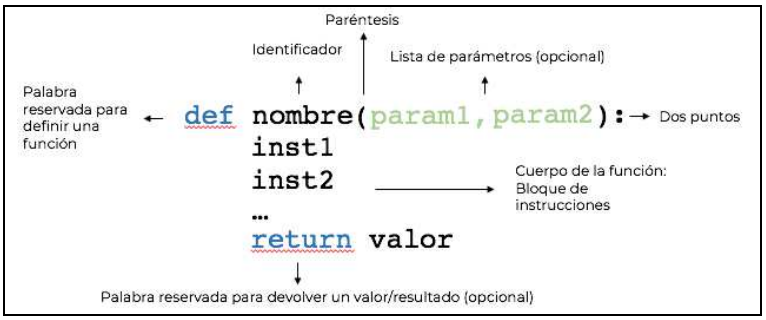
Funciones.

Una función no es más que un bloque de código (líneas de código) aislado (separado del programa) 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 más modulares y DRY para problemas complejos.

> *(**DRY**) "No te repitas". Es un principio del desarrollo de software destinado a reducir la repetición de patrones de software, reemplazándolos con abstracciones o utilizando la normalización de datos para evitar la redundancia.*
>
> *El 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".[3] Cuando el principio DRY se aplica con éxito, una modificación de cualquier elemento individual de 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 podrás usar en tus proyectos.
Una de las grandes ventajas de usar funciones en tu código es que reduce el número total de líneas de código del proyecto, por lo tanto más fácil de entender y corregir.

Comencemos.
Podremos definir una función, iniciando con la instrucción **def,** más un nombre de función descriptivo, para el cuál, 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:***




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

Y como se mencionó antes, desde nuestro programa podremos usarla, tantas veces como sea necesario, simplemente llamándola. Veamos un ejemplo completo: Un programa que tiene una función y la usa. También veremos lo que aparece en la pantalla.

MUY IMPORTANTE: **Por claridad,** a la definición de una función, la debe anteceder al menos una línea en blanco.

Código	Veremos en Pantalla
Func_001_Iniciando_001a_Hola_Mundo # -*- coding: utf-8 -*- """ Creado 24/04/2022 ***Primera Función - Hola Mundo.*** @author: Horacio """ print("--") print(" Escribiendo mi Primera Función.") print("--\n") ***# Definicion de una función*** def mi_primera_funcion( ): print("Hola Mundo") ***# Y ahora el programa*** print("Comienza el programa\n") mi_primera_funcion( ) ***#Llamamos a la función.*** ***# Termina Programa*** print("\n--") print("Programa Terminado.")	-- Escribiendo mi Primera Función. -- Comienza el programa Hola Mundo -- Programa Terminado.

Definición y Uso de Funciones con Python

En un programa, podremos escribir tantas funciones como queramos o necesitemos.

Código	Veremos en Pantalla

Func_001_Iniciando_001b_switch_dias_Semana
"""
Creado 24/04/2022
Función Muestra por Monitor día de la semana
@author: Horacio
"""
print("----------")
print(" Función Muestra por Monitor día de la semana")
print("----------\n")
def lunes( ):
print('Hoy es lunes')
def martes( ):
print('martes')
def miercoles( ):
print('miércoles')
def jueves( ):
print('jueves')
def viernes( ):
print('viernes')
def sabado( ):
print('sábado')
def domingo( ):
print('domingo')
def error( ):
print('error')
switch_semana = {
1: lunes,
2: martes,
3: miercoles,
4: jueves,
5: viernes,
6: sabado,
7: domingo
}
dia = int(input("Ingrese numero del día de la semana: "))
#tomamos la función asociada a la variable y la invocamos
switch_semana.get(dia, error)( )
# Termina Programa 	
print("\n----------")
print("Programa Terminado.")
	---------- 	
	Función Muestra por Monitor día de la semana
	---------- 	
	
	Ingrese numero del día de la semana: 1
	Hoy es lunes
	
	---------- 	
	Programa Terminado.

Una función, puede tener cualquier algoritmo, incluso podrá tener uno o muchos algoritmos, además puede utilizar cualquiera de las características vistas o que veremos a continuación. No obstante ello, una buena práctica del programador, es que una función debe ser pensada para, realizar una única acción, reutilizable y por lo tanto, tan genérica como sea posible.

Retorno de Valores

Las funciones pueden entregar o retornar (cuando se ejecutan) algún valor. Esto lo hace mediante el la sentencia return( ), que reforma en valor indicado.

Código	Veremos en Pantalla
Func_001_Iniciando_001c_Retorna_Valor # -*- coding: utf-8 -*- """ Creado 24/04/2022 ***Ejemplo de una Función retornando un valor*** @author: Horacio """ print("----------") print(" Función retornando un valor ") print("----------\n")	---------- Función retornando un valor ---------- Comienza el programa Ingresar un numero: 135

```
# Definición de una función
def Retorna_Valor( ):  #Nombre de la función
    Nro = int(input("Ingresar un numero: "))
    return Nro  #Retorna el numero leído

print("Comienza el programa\n")
print("\nEl valor que la función Retorna es: ", Retorna_Valor( ))

# Termina Programa
print("\n-----------------------------------------------")
print("Programa Terminado.")
```

```
El valor que la función Retorna es:  135

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

En una función, con el "***return***", podremos retornar tantos valores como queramos. Analiza este ejemplo.

Código

```
Func_001_Iniciando_001d_Retorna_Dos_Valores
 # -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Ejemplo de una Función retornando dos Valores
@author: Horacio
"""
print("-------------------------------------------------")
print(" Función retornando dos Valores")
print("    Formatos de Impresión")
print("-------------------------------------------------")

# Definición de una función
def Retorna_Datos():
    Nom = input("Ingresa tu Nombre: ")
    Ed = int(input("Ahora tu Edad: "))
    return (Nom, Ed)

print("Comienza el programa")

print("Datos Leídos (Formato 01): ", Retorna_Datos())
print("-------------------------------------------------")

print("Datos Leídos (Formato 02): Nombre %s, Edad: %i" % (Retorna_Datos()))
print("-------------------------------------------------")

Nombre, Edad = Retorna_Datos()
print("Datos Leídos (Formato 03): ", Nombre, Edad )
print("Datos Leídos (Formato 04): Nombre %s, Edad: %i" % (Nombre, Edad))

# Termina Programa
print("\n-----------------------------------------------")
print("Programa Terminado.")
```

Veremos en Pantalla

```
-------------------------------------------------
 Función retornando dos Valores
    Formatos de Impresión
-------------------------------------------------
Comienza el programa

Ingresa tu Nombre: Susana

Ahora tu Edad: 16
Datos Leídos (Formato 01):  ('Susana', 16)
-----------------------------------------------

Ingresa tu Nombre: Pedro

Ahora tu Edad: 17
Datos Leídos (Formato 02): Nombre Pedro, Edad: 17
-----------------------------------------------

Ingresa tu Nombre: Profe

Ahora tu Edad: 374
Datos Leídos (Formato 03):  Profe 374
Datos Leídos (Formato 04): Nombre Profe, Edad: 374

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

-o-

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 los hay, se indican entre los paréntesis, a modo de variables, a fin de poder utilizarlos como tales, dentro de la misma función.

Algo muy importante y que siempre debes recordar:
- Los parámetros que una función recibe, solo podrán ser usados por ella y dentro de ella (el código que este dentro de ella) y reciben el nombre de "variables de ámbito local" o simplemente "Variables locales" al igual que cualquier variable que se defina dentro de la función.
- Si la definición de una función incluye parámetros, cuando llames a la función, le debes proporcionar el mismo número de parámetros y el mismo orden que los espera.

Código

```
Func_002_Recibe_Parametros_001a_Recibe_Apellido_DNI
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Funcion con Parametros de entrada
@author: Horacio
"""
print("------------------------------------------------------------")
```

Veremos en Pantalla

```
------------------------------------------------------------
Escribir una programa que tenga una función que reciba
como. Parámetros Un Saludo y el nombre, y muestre por
pantalla.
------------------------------------------------------------

Comienza el programa
```

```
print("Escribir una programa que tenga una función que.")
print("reciba como Parámetros un Saludo y el nombre, y.")
print("lo muestre por pantalla.")
print("-------------------------------------------------------\n")

# Definición de una función
def Datos_Alumno(dni, nombre):
    Alumno = dni, nombre     #Los Guardo en una lista para después
    print("Los datos (directamente):  ",dni, nombre)
    print("Los datos (desde una lista): ",Alumno[0], Alumno[1])

# Y ahora el programa
print("Comienza el programa\n")

Nombre = "Fernando"
DNI = "32.143.912"

Datos_Alumno( 'Hola' , Nombre )

# Termina Programa
print("\n-------------------------------------------------------")
print("Programa Terminado.")
```

```
Los datos (directamente):   Hola Fernando
Los datos (desde una lista):  Hola Fernando

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

Recepción de Datos y Entrega de Resultados

Habiendo ya visto que las funciones **por un lado** pueden retornar uno o varios datos y **por otro lado** recibir datos/información (parámetros), es lógico que **una misma función** pueda recibir, procesar y retornar resultados . *Veamos un ejemplo*:

Func_003_Recibe_Parametros_001a_Suma_dos_Números

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Recibe dos números y retorna la suma
@author: Horacio
"""
print("-------------------------------------------------")
print(" Función Recibe dos números y retorna la suma")
print("-------------------------------------------------\n")

# Definición de una función
def Suma_Numeros( Nro_1, Nro_2 ):
    R = Nro_1 + Nro_2
    return R

print("Comienza el programa\n")

Dato_01 = int(input("Ingrese un número Entero: "))
Dato_02 = int(input("Ingrese Otro número Entero: "))
Resultado = Suma_Numeros(Dato_01, Dato_02)

print("\nLa suma es: ", Resultado)

# Termina Programa
print("\n-------------------------------------------------")
print("Programa Terminado.")
```

```
-------------------------------------------------
 Función Recibe dos números y retorna la suma
-------------------------------------------------

Comienza el programa

Ingrese un número Entero: 2
Ingrese Otro número Entero: 3

La suma es:  5

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

-o-

Func_003_Recibe_y_Retorna_001b_Dos_Funciones

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Función Recibe y Retornando un Resultado
@author: Horacio
"""
print("-------------------------------------------------")
print(" Función Recibe y Retornando un Resultado")
print("-------------------------------------------------\n")

# Definición de una función
def Retorna_Valor( ):
    Nro = int(input("Ingresar un número: "))
    return Nro

def Calcula_Cuadrado( Nro ):
```

```
-------------------------------------------------
 Función Recibe y Retornando un Resultado
-------------------------------------------------

Comienza el programa


Ingresar un número: 5

El resultado de la función es:  25

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

```
    Cuadrado = Nro**2
    return Cuadrado

print("Comienza el programa\n")
Numero = Retorna_Valor( )
Resultado = Calcula_Cuadrado(Numero)
print("\nEl resultado de la función es: ", Resultado)

# Termina Programa
print("\n-----------------------------------------------")
print("Programa Terminado.")
```

Parámetros por Omisión

Cuando definimos una función, también es posible, asignar valores por defecto a los parámetros que usemos. Esto significa, que la función podrá ser llamada con menos argumentos (parámetros) de los que ella espera. Analicemos un ejemplo donde usamos parámetros omitidos.

MUY IMPORTANTE:

- Al asignar parámetros por omisión, no debe dejarse espacios en blanco ni antes ni después del signo =.

- Los parámetros omitidos deben ser omitidos desde el último, en orden hacia adelante.

Func_004_Parametros_Omision_001a_Mensaje

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Ejemplos Función con Parámetros por Omisión
@author: Horacio
"""
print("-------------------------------------------------")
print("    Ejemplo de parámetro por omisión")
print("-------------------------------------------------\n")

# Definición de una función
def  Saludar(nombre='Profe.', mensaje='Bienvenido'):
    print( mensaje, nombre )

print("Comienza el programa\n")
Nombre = input("Ingresa tu Nombre: ")
print("\n-------------------------------------------------\n")
# Y ahora la función recibe dos parámetros
print("Entrego 2 parámetros")
Saludar( Nombre, 'Hola, como estas ')
print("\n-------------------------------------------------\n")
# Se activa el Parámetro omitido, solo entrego un Parámetro a la función
print("Entrego solo 1 parámetro")
Saludar( Nombre)
print("\n-------------------------------------------------\n")
# Se activa el Parámetro omitido, NO entrego Parámetros a la función
print("Sin Parámetros")
Saludar( )
# Termina Programa
print("\n-----------------------------------------------")
print("Programa Terminado.")
```

```
-------------------------------------------------
    Ejemplo de parámetro por omisión
-------------------------------------------------

Comienza el programa


Ingresa tu Nombre: Susana

-------------------------------------------------

Entrego 2 parámetros
Hola, como estas  Susana

-------------------------------------------------

Entrego solo 1 parámetro
Bienvenido Susana

-------------------------------------------------

Sin Parámetros
Bienvenido Profe.

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

Cantidad Desconocida (arbitrarios) de Parámetros

Como en algunos otros lenguajes de programación, en Python es posible que una función, espere recibir un número desconocido o arbitrario de argumentos (parámetros). Estos argumentos, llegarán a la función en forma de tupla y respetando siempre el mismo orden.

Para definir argumentos no esperados o arbitrarios en una función, se antecede al parámetro un asterisco (*):

MUY IMPORTANTE: Si una función espera recibir parámetros fijos y arbitrarios, **los arbitrarios siempre deben suceder a (continuación de) los fijos**.

Código	Veremos en Pantalla

Func_005_Parametros_Arbitrarios_001a_Datos_Inesperados
# -*- coding: utf-8 -*- 	
"""
Creado 24/04/2022
Ejemplo de parámetro Arbitrarios o Inesperados
@author: Horacio
"""
print("----------")
print(" Ejemplo de parámetro Arbitrarios o Inesperados ")
print("----------\n")
 	
# Definición de una función 	
def Mostrar_Parametros_Arbitrarios(parametro_Esperado, *arbitrarios):
  print(parametro_Esperado)
 	
  # Los parámetros arbitrarios se corren como tuplas
  for argumento in arbitrarios:
  print(argumento)
 	
print("Comienza el programa\n")
Mostrar_Parametros_Arbitrarios('Esperado',
  'Inesperado 1',
  'Inesperado 2',
  'Inesperado 3')
# Termina Programa 	
print("\n----------")
print("Programa Terminado.")
	---------- 	
	  Ejemplo de parámetro por omisión
	---------- 	
	
	Comienza el programa
	
	Esperado
	Inesperado 1
	Inesperado 2
	Inesperado 3
	
	---------- 	
	Programa Terminado.

Las Funciones También se pueden Asignar

Para Python, las funciones son objetos, por lo tanto, se las puede asignar a una variable para luego usar esa variable como una función. Esto es un manejo algo avanzado ara este momento del aprendisaje, pero es relativamente común encontrar este medo de programar, por lo tanto se incluye en este apunte. Por lo menos para que se conozca la existencia.

Código	Veremos en Pantalla

# -*- coding: utf-8 -*- 	
"""
Creado 24/04/2022
Funciones Objetos - Asignación de funciones a Variables
@author: Horacio
"""
print("----------")
print("Funciones Objetos - Asignación de funciones a Variables")
print("----------\n")
 	
# Definición de funciones 	
def Lee_Valor( ):
  Nro = int(input("Ingresar un número: "))
  return Nro
 	
def Calcula_Cuadrado(Nro):
  return Nro**2
 	
def Muestra_Valor(Nro):
  print("El Cuadrado es: ", Nro)
 	
print("---------- Comienza el programa ----------\n")
L = Lee_Valor ***# Asigno a la Variable L***
C = Calcula_Cuadrado ***# Asigno a la Variable C***
M = Muestra_Valor ***# Asigno a la Variable M***
 	
print("\nEjemplo 01 - Uso Nombre Originales")
Numero = Lee_Valor( )
Cuadrado = Calcula_Cuadrado(Numero)
Muestra_Valor(Cuadrado)
 	
print("\nEjemplo 02 - Uso Nombre Cambiados.")
Numero = L( )
Cuadrado = C(Numero)
M(Cuadrado)
 	
print("\nEjemplo 03 - llamado Críptico.")
	---------- 	
	Funciones Objetos - Asignación de funciones a Variables
	---------- 	
	
	---------- Comienza el programa ----------
	
	Ejemplo 01 - Uso Nombre Originales
	Ingresar un número: 2
	El Cuadrado es: 4
	
	Ejemplo 02 - Uso Nombre Cambiados.
	Ingresar un número: 3
	El Cuadrado es: 9
	
	Ejemplo 03 - llamado Críptico.
	Cada función Toma lo que la otra retorna.
	Ingresar un número: 4
	El Cuadrado es: 16
	
	Ejemplo 04 - Críptico - Nombres Cambiados.
	Cada función Toma lo que la otra retorna.
	Ingresar un número: 5
	El Cuadrado es: 25
	
	---------- 	
	Programa Terminado.

```
print("Cada función Toma lo que la otra retorna.")
Muestra_Valor( Calcula_Cuadrado( Lee_Valor( ) ) )

print("\nEjemplo 04 - Críptico - Nombres Cambiados.")
print("Cada función Toma lo que la otra retorna.")
M(C(L( )))

# Termina Programa
print("\n-----------------------------------------------")
print("Programa Terminado.")
```

Una Tema Fuera de Lugar, Pero Muy Fácil y Útil.

Y ahora veremos muy superficialmente, y solo por su gran utilidad, un tipo de dato que el usuario puede definir. Esta **clase** de datos la usaremos en muchas partes, incluyendo en la definición de listas (acá veremos un ejemplo) aunque las estudiaremos más detalladamente un poco más adelante.

Una **Clase**, como se mencionó anteriormente, es un **tipo de dato definido por el usuario** (aunque el lenguaje ya posee muchas), que nos permite crear instancias (objetos o variables de ese tipo). Las clases y los objetos son considerados los principales bloques de desarrollo para **Python**, el cual es un lenguaje de programación orientado a objetos.

Para crear una **clase,** usaremos la palabra reservada **class** seguido de un nombre, escrito en minúscula, a excepción de la primera letra de cada palabra, que se escribe en mayúscula y terminando la línea con dos puntos. Por ejemplo, declararemos una clase "*Producto*", que por ahora estará vacía, pero luego iremos completando:

Este pequeño apartado es para los que ya programaron en C, C++ o cualquiera de sus variantes. Sucede que Python no tiene una forma específica para definir una estructura "**struct**", ya que podremos trabajar directamente con objetos, los que tienen una forma más flexible y en algunos casos más poderosos que el C. este mismo ejemplo en C lo habríamos escrito así (si lo pruebas, asegúrate de cumplir con todos los requisitos del compilador de tu versión de C):

```
struct Producto{
   string Nombre;
   float Precio;
   string Descripcion:
   Producto(int nom="", float pre=0.0){
      strcpy( Nombre, Nom);
       Precio = pre;
    }
}

//luego en el programa
Producto a;
```

```
class Producto:
    pass
```

A continuaron, analizaremos un programa, en que ejemplo tras ejemplo, iremos agregando funcionalidades, permitiendo evolucionar en el conocimiento (usaremos prácticamente todas las cosas aprendidas anteriormente)

Comencemos por crear una clase de datos llamada Productos (es la que usamos de ejemplo unas líneas mas arriba y continua vacía). A continuación creamos y usaremos una función que llamaremos "***crear_Producto",*** como recomendación puedo agregar que prestes mucha atención, ya que usaremos prácticamente todo lo visto antes.

Programa: P_00_Especiales/struct_01.py

Ejemplos de una Clase Vacía

Código

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Objeto de una clase vacía
@author: Horacio
"""
print("------------------------------------------------------------")
print(" Creación de una Variable (Objeto) de una clase Vacía")
print("              Incorporación de campos")
print("         Proceso de carga y mostrar campos")
print("------------------------------------------------------------\n")

class Producto:  #Una clase de datos especial (Por ahora la declaramos vacía)
   pass

print("Preguntar al profe el funcionamiento de este proceso")

def crear_Producto(nombre=None, precio=0.0):  # Parámetros omitidos
   p = Producto( )  #Creación de una variable (objeto) de la clase Productos
   p.nombre = str(nombre)  #Crea en el objeto 1 variable y asignamos valor
   p.precio = float(precio) #crea en el objeto 1 variable y asignamos valor
```

Veremos en Pantalla

```
------------------------------------------------------------
Creación de una Variable (Objeto) de una clase Vacía
              Incorporación de campos
         Proceso de carga y mostrar campos
------------------------------------------------------------
Preguntar al profe el funcionamiento de este proceso

Producto: None , Precio: 0.0
None
0.0
==============================

Producto: Aceite , Precio: 9.5
Aceite
9.5
==============================


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

Programa: P_00_Especiales/struct_01.py

Ejemplos de una Clase Vacía

Código	Veremos en Pantalla

```
    p.descripcion = "Producto: " + p.nombre + " , Precio: " + str(p.precio)
    return p  # Retornamos el objeto (variable) creado y con datos.

a = crear_Producto( )
print( a.descripcion)
print( a.nombre)
print( a.precio)
print("================================")

b = crear_Producto("Aceite", 9.5)
print( b.descripcion)
print( b.nombre)
print( b.precio)
print("================================")

# Termina Programa
print("\n-------------------------------------------------------")
print("Programa Terminado.")
```

-o-

Programa: P_00_Especiales/struct_02.py

Ejemplos de una Clase Vacía

Código	Veremos en Pantalla

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Objeto de una clase vacía
@author: Horacio
"""
print("-------------------------------------------------------")
print(" ")
print("-------------------------------------------------------\n")

def Leer_Nombre( ):
    nombre = input("Nombre: ")
    return nombre

def Leer_Precio( ):
    precio = input("Precio: ")
    return precio


class Producto:
    pass

def crear_Producto(nombre=None, precio=0.0):
    p = Producto( )
    p.nombre = str(nombre)
    p.precio = float(precio)
#    p.descripcion = "Producto: %s, Precio: %f" % (p.nombre, p.precio)
#    p.descripcion = "Producto:", p.nombre, " , Precio: ", str(p.precio)
    p.descripcion = "Producto: " + p.nombre + " , Precio: " + str(p.precio)
    return p

a = crear_Producto( )
print( a.descripcion)
print( a.nombre)
print( a.precio)
print("================================")

b = crear_Producto("Aceite", 9.5)
print( b.descripcion)
print( b.nombre)
print( b.precio)
print("================================")

c = crear_Producto(Leer_Nombre( ), Leer_Precio( ))
print( c.descripcion)
```

```
-------------------------------------------------------

-------------------------------------------------------

Producto:None , Precio: 0.0
None
0.0
================================
Producto:Aceite , Precio: 9.5
Aceite
9.5
================================

Nombre: Primero

Precio: 1
Producto: Primero , Precio: 1.0
Primero
1.0
================================

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

Programa: P_00_Especiales/struct_02.py

Ejemplos de una Clase Vacía

Código	Veremos en Pantalla

```
print( c.nombre)
print( c.precio)
print("================================")

# Termina Programa
print("\n-------------------------------------------------------")
print("Programa Terminado.")
```

-o-

Programa: P_00_Especiales/struct_03.py

Ejemplos de una Clase Vacía

Código

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Objeto de una clase vacía
@author: Horacio
"""
print("-------------------------------------------------------------")
print(" ")
print("-------------------------------------------------------------\n")

def Leer_Nombre( ):
    nombre = input("Nombre: ")
    return nombre

def Leer_Precio( ):
    precio = input("Precio: ")
    return precio

class Producto:
    pass

def crear_Producto(nombre=None, precio=0.0):
    p = Producto( )
    p.nombre = str(nombre)
    p.precio = float(precio)
#    p.descripcion = "Producto: %s, Precio: %f" % (p.nombre, p.precio)
#    p.descripcion = "Producto:", p.nombre, " , Precio: ", str(p.precio)
    p.descripcion = "Producto: " + p.nombre + " , Precio: " + str(p.precio)
    return p

a = crear_Producto( )
print( a.descripcion)
print( a.nombre)
print( a.precio)
print("================================")

b = crear_Producto("Aceite", 9.5)
print( b.descripcion)
print( b.nombre)
print( b.precio)
print("================================")


c = crear_Producto(Leer_Nombre( ), Leer_Precio( ))
print( c.descripcion)
print( c.nombre)
print( c.precio)
print("================================")
print("================================")

print("Elementos de la lista")
Lista = []

for i in range(5):
```

Veremos en Pantalla

```
-------------------------------------------------------------

-------------------------------------------------------------

Producto: None , Precio: 0.0
None
0.0
================================
Producto: Aceite , Precio: 9.5
Aceite
9.5
================================

Nombre: Arbejas (Variable C)

Precio: 16.45
Producto: Arbejas (Variable C) , Precio: 16.45
Arbejas (Variable C)
16.45
================================
================================
Elementos de la lista
Nombre: Pollo (Lista Pos 0)

Precio: 1.1
================================
Nombre: Papas (Lista Pos 1)

Precio: 2.2
================================
Nombre: Pescado (Lista Pos 2)

Precio: 3.3
================================
Nombre: Remolacha (Lista Pos 3)

Precio: 4.1
================================
Nombre: Queso (Lista Pos 4)

Precio: 5.1
================================
Muestro Elementos

Producto: Pollo (Lista Pos 0) , Precio: 1.1
Pollo (Lista Pos 0)
1.1
================================
Producto: Papas (Lista Pos 1) , Precio: 2.2
Papas (Lista Pos 1)
2.2
================================
Producto: Pescado (Lista Pos 2) , Precio: 3.3
Pescado (Lista Pos 2)
```

Programa: P_00_Especiales/struct_03.py

Ejemplos de una Clase Vacía

Código	Veremos en Pantalla

```
    Lista.append(crear_Producto(Leer_Nombre( ), Leer_Precio( )))
    print("================================")

print("Muestro Elementos")
for i in range(5):
    print( Lista[i].descripcion)
    print( Lista[i].nombre)
    print( Lista[i].precio)
    print("================================")

# Termina Programa
print("\n------------------------------------------------------------")
print("Programa Terminado.")
```

```
3.3
================================
Producto: Remolacha (Lista Pos 3) , Precio: 4.1
Remolacha (Lista Pos 3)
4.1
================================
Producto: Queso (Lista Pos 4) , Precio: 5.1
Queso (Lista Pos 4)
5.1
================================
------------------------------------------------------------
Programa Terminado.
```

-o-

Programa: P_00_Especiales/struct_04.py

Ejemplos de una Clase Vacía

Código	Veremos en Pantalla

```
# -*- coding: utf-8 -*-
"""
Creado 24/04/2022
Objeto de una clase vacía
@author: Horacio
"""
print("------------------------------------------------------------")
print(" ")
print("------------------------------------------------------------\n")

def Leer_Nombre( ):
    nombre = input("Nombre: ")
    return nombre

def Leer_Precio( ):
    precio = input("Precio: ")
    return precio

class Producto:
    pass

def crear_Producto(nombre=None, precio=0.0):
    p = Producto( )

    p.LN = Leer_Nombre
    p.LP = Leer_Precio

    if nombre == None and not precio:
        p.nombre = p.LN( )
        p.precio = p.LP( )
    else:
        p.nombre = str(nombre)
        p.precio = float(precio)
#    p.descripcion = "Producto: %s, Precio: %f" % (p.nombre, p.precio)
#    p.descripcion = "Producto:", p.nombre, " , Precio: ", str(p.precio)
    p.descripcion = "Producto:" + p.nombre + " , Precio: " + str(p.precio)
    return p

a = crear_Producto( )
print( a.descripcion)
print( a.nombre)
print( a.precio)
print("================================")

b = crear_Producto("Aceite", 9.5)
print( b.descripcion)
print( b.nombre)
print( b.precio)
print("================================")
```

Programa: P_00_Especiales/struct_04.py

Ejemplos de una Clase Vacía

Código	Veremos en Pantalla

```
c = crear_Producto(Leer_Nombre( ), Leer_Precio( ))
print( c.descripcion)
print( c.nombre)
print( c.precio)
print("================================")
print("================================")

Lista = []

for i in range(5):
    Lista.append(crear_Producto( ))
    print("================================")

for i in range(5):
    print( Lista[i].descripcion)
    print( Lista[i].nombre)
    print( Lista[i].precio)
    print("================================")

# Termina Programa
print("\n-----------------------------------------------------------")
print("Programa Terminado.")
```

Quieres Repasar y Aprender un poco más sobre Funciones?

Te recomiendo visites este sitio:

https://www.freecodecamp.org/espanol/news/guia-de-funciones-de-python-con-ejemplos/

https://www.freecodecamp.org/espanol/news/ejemplos-de-funciones-de-python-como-declarar-y-llamar-con-parametros/