1. Variables y Tipos Básicos

```python
# Inicializando variables
x = 10  # variable de tipo entero
print(x)

cadena = "Hola Mundo"  # variable de tipo cadena
print(cadena)

# Asigna un mismo valor a tres variables
x = y = z = 10
print(x, y, z)

# La función type() permite conocer el tipo de una variable
print(type(x))
print(type(cadena))

# Se pueden cambiar los valores de las variables y el tipo se cambia automáticamente
x = "Hola Mundo"
cadena = 10
print(type(x))
print(type(cadena))

# Variables constantes (por convención en mayúsculas)
SEGUNDOS_POR_DIA = 60 * 60 * 24
PI = 3.14
print(f"Segundos por día: {SEGUNDOS_POR_DIA}")
```

```
10
Hola Mundo
10 10 10
<class 'int'>
<class 'str'>
<class 'str'>
<class 'int'>
Segundos por día: 86400
```

2. Manejo de Cadenas

```python
# Inicializando cadenas
cadena1 = 'Hola'
cadena2 = "Mundo"
print(cadena1)
print(cadena2)

# Concatenación de cadenas
concat_cadenas = cadena1 + " " + cadena2
print(concat_cadenas)

# Para concatenar un número y una cadena se debe usar la función str()
num_cadena = concat_cadenas + ' ' + str(3)
print(num_cadena)

# Uso de format() - método recomendado
num_cadena = "{} {} {}".format(cadena1, cadena2, 3)
print(num_cadena)

# Cambiando el orden con format()
num_cadena = "Cambiando el orden: {1} {2} {0}".format(cadena1, cadena2, 3)
print(num_cadena)
```

```
Hola
Mundo
Hola Mundo
Hola Mundo 3
Hola Mundo 3
Cambiando el orden: Mundo 3 Hola
```

3. Operadores

```python
# Operadores aritméticos
print(1 + 5)
print(6 - 3)
print(10 - 4)
print(100 / 50)
print(10 % 2)
print(((20 * 3) + (10 + 1)) / 10)
print(2 ** 2)

# Operadores booleanos
print(False and True)
print(True or False)
print(not True)

# Operadores de comparación
print(7 < 5)  # False
print(7 > 5)  # True
print((11 * 3) + 2 == 36 - 1)  # True
print((11 * 3) + 2 >= 36)  # False
print("curso" != "CuR80")  # True
```

```
6
3
6
2.0
0
7.1
4
False
True
False
False
True
True
False
True
```

4. Listas

```python
# Declaración de una lista simple
lista_diasDelMes = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

print(lista_diasDelMes)     # imprimir la lista completa
print(lista_diasDelMes[0])  # imprimir elemento 1
print(lista_diasDelMes[6])  # imprimir elemento 7
print(lista_diasDelMes[11]) # imprimir elemento 12

# Declaración de listas anidadas
lista_numeros = [['cero', 0], ['uno', 1, 'UNO'], ['dos', 2], ['tres', 3], ['cuatro', 4], ['X', 5]]

print(lista_numeros)        # imprimir lista completa
print(lista_numeros[0])     # imprime el elemento 0 de la lista
print(lista_numeros[1])     # imprime el elemento 1 de la lista
print(lista_numeros[2][0])  # imprime el primer elemento de la lista en la posición 2
print(lista_numeros[2][1])  # imprime el segundo elemento de la lista en la posición 2
print(lista_numeros[1][0])
print(lista_numeros[1][1])
print(lista_numeros[1][2])

# Se cambia el valor de uno de los elementos de la lista
lista_numeros[5][0] = "cinco"
print(lista_numeros[5])
```

```
[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
31
31
31
[['cero', 0], ['uno', 1, 'UNO'], ['dos', 2], ['tres', 3], ['cuatro', 4], ['X', 5]]
['cero', 0]
['uno', 1, 'UNO']
dos
2
uno
1
UNO
['cinco', 5]
```

5. Tuplas

```python
# Declaración de una tupla
tupla_diasDelMes = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
print(tupla_diasDelMes)     # Imprimir la tupla completa
print(tupla_diasDelMes[0])  # Imprimir elemento 1
print(tupla_diasDelMes[3])  # Imprimir elemento 4
print(tupla_diasDelMes[1])  # Imprimir elemento 2

# Declaración de tuplas anidadas
tupla_numeros = (('cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('X', 5))
print(tupla_numeros)        # imprimir tupla completa
print(tupla_numeros[0])     # imprime el elemento 0 de la tupla
print(tupla_numeros[1])     # imprime el elemento 1 de la tupla
print(tupla_numeros[2][0])  # imprime el primer elemento de la tupla en la posición 2
print(tupla_numeros[2][1])  # imprime el segundo elemento de la tupla en la posición 2
print(tupla_numeros[1][0])
print(tupla_numeros[1][1])
print(tupla_numeros[1][2])

# Probando la mutabilidad de las listas vs la no mutabilidad de las tuplas
print("valor actual {}".format(lista_diasDelMes[0]))
lista_diasDelMes[0] = 50
print("valor cambiado {}".format(lista_diasDelMes[0]))
```

```
(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
30
28
(('cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('X', 5))
('cero', 0)
('uno', 1, 'UNO')
dos
2
uno
1
UNO
valor actual 50
valor cambiado 50
```

6. Tuplas con Nombre

```python
from collections import namedtuple

# Se crea la tupla con nombre
planeta = namedtuple('planeta', ['nombre', 'numero'])

# Se crea el planeta 1 y se agregan a la tupla los valores correspondientes a los campos
planeta1 = planeta('Mercurio', 1)
print(planeta1)

# Se crea el planeta 2
planeta2 = planeta('Venus', 2)

# Se imprimen los valores de los campos
print(planeta1.nombre, planeta1.numero)
print(planeta2[0], planeta2[1])

print('Campos de la tupla: {}'.format(planeta1._fields))
```

```
planeta(nombre='Mercurio', numero=1)
Mercurio 1
Venus 2
Campos de la tupla: ('nombre', 'numero')
```

7. Diccionarios

```python
# Creando un diccionario
elementos = {'hidrogeno': 1, 'helio': 2, 'carbon': 6}

print(elementos)
print(elementos['hidrogeno'])

# Se pueden agregar elementos al diccionario
elementos['litio'] = 3
elementos['nitrogeno'] = 8
print(elementos)

# Creando un nuevo diccionario
elementos2 = {}
elementos2['H'] = {'name': 'Hydrogen', 'number': 1, 'weight': 1.00794}
elementos2['He'] = {'name': 'Helium', 'number': 2, 'weight': 4.002602}
print(elementos2)

# Imprimiendo los datos de un elemento del diccionario
print(elementos2['H'])
print(elementos2['H']['name'])
print(elementos2['H']['number'])
elementos2['H']['weight'] = 4.30  # Cambiando el valor de un elemento
print(elementos2['H']['weight'])

# Agregando elementos a una llave
elementos2['H'].update({'gas noble': True})
print(elementos2['H'])

# Mostrando todos los elementos del diccionario
print(elementos2.items())

# Mostrando todas las llaves del diccionario
print(elementos2.keys())
```

```
(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
30
28
(('cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('X', 5))
('cero', 0)
('uno', 1, 'UNO')
dos
2
uno
1
UNO
valor actual 50
valor cambiado 50
```

8. Funciones

```python
# Las funciones pueden recibir n número de parámetros
def imprime_nombre(nombre):
    print("hola " + nombre)

# Llamada a la función
imprime_nombre("Alan Juarez")

# Definiendo una función que regresa el cuadrado de un número
def cuadrado(x):
    return x ** 2

x = 5
print("El cuadrado de {} es {}".format(x, cuadrado(x)))

# Definiendo una función que regrese más de un valor
def varios(x):
    return x ** 2, x ** 3, x ** 4

# Los valores que regresa la función pueden ser guardado en variables separadas por ,
val1, val2, val3 = varios(2)
print("{} {} {}".format(val1, val2, val3))

# Función con un parámetro con un valor por defecto
def cuadrado_default(x=3):
    return x ** 2

print(cuadrado_default())  # Usa el valor por defecto
print(cuadrado_default(5)) # Usa el valor proporcionado

# Usando el operador '_' para ignorar valores
val4, _, val5 = varios(2)
print("{} {}".format(val4, val5))
```

```
hola Alan Juarez
El cuadrado de 5 es 25
4 8 16
9
25
4 16
```

9. Variables Globales

```
# Se crea una variable en el espacio global de nombres
vg = 'Global'

# Se crea una función que imprime la variable global
def function_v1():
    print(vg)

# Llamada a la función que imprime la variable global
function_v1()
print(vg)

# Se crea una variable local que tiene el mismo nombre que la variable global
def function_v2():
    vg = "Local"
    print(vg)

function_v2()
print(vg)  # Sigue imprimiendo "Global"

# Usando global para modificar la variable global
def function_v4():
    global vg
    print(vg)
    vg = "Local"
    print(vg)

function_v4()
print(vg)
```

```
Global
Global
Local
Global
Global
Local
Local
```

10. Estructuras de Control Selectivas

```python
# if simple
def obtenerMayor(param1, param2):
    if param1 < param2:
        print('{} es mayor que {}'.format(param2, param1))

obtenerMayor(5, 7)
obtenerMayor(7, 5)  # No imprime nada

# if-else
def obtenerMayorv2(param1, param2):
    if param1 < param2:
        return param2
    else:
        return param1

print("El mayor es {}".format(obtenerMayorv2(4, 20)))
print("El mayor es {}".format(obtenerMayorv2(11, 6)))

# Operador ternario en Python
def obtenerMayor_idiom(param1, param2):
    valor = param2 if (param1 < param2) else param1
    return valor

print("El mayor es {}".format(obtenerMayor_idiom(11, 6)))

# if-elif-else
def numeros(num):
    if num == 1:
        print("tu numero es 1")
    elif num == 2:
        print("el numero es 2")
    elif num == 3:
        print("el numero es 3")
    elif num == 4:
        print("el numero es 4")
    else:
        print("no hay opcion")

numeros(2)
numeros(5)

# Versión idiomática
def numeros_idiom(num):
    if num in (1, 2, 3, 4):
        print("tu numero es {}".format(num))
    else:
        print("{} no es una opcion".format(num))

numeros_idiom(2)
numeros_idiom(5)

# Estructura de control selectiva anidada
def obtenerMasGrande(a, b, c):
    if a > b:
        if a > c:
            return a
        else:
            return c
    else:
        if b > c:
            return b
        else:
            return c

print("El mas grande es {}".format(obtenerMasGrande(7, 13, 1)))
```

```
7 es mayor que 5
El mayor es 20
El mayor es 11
El mayor es 11
el numero es 2
no hay opcion
tu numero es 2
5 no es una opcion
El mas grande es 13
```

11. Estructuras de Control Repetitivas

```python
# Ciclo while - Ejemplo 1
def cuenta(limite):
    i = limite
    while True:
        print(i)
        i = i - 1
        if i == 0:
            break  # Rompiendo el ciclo

cuenta(5)

# Ciclo while - Ejemplo 2
def factorial(n):
    i = 2
    tmp = 1
    while i < n + 1:
        tmp = tmp * i
        i = i + 1
    return tmp

print(factorial(4))
print(factorial(6))

# Ciclo for - Iteración en listas
for x in [1, 2, 3, 4, 5]:
    print(x)

# La función range() sirve para generar una lista
for x in range(5):  # Equivalente a range(0,5)
    print(x)

# También se puede inicializar desde números negativos
for x in range(-5, 2):
    print(x)

for num in ["uno", "dos", "tres", "cuatro"]:
    print(num)

# Iteración en diccionarios
elementos = {'hidrogeno': 1, 'helio': 2, 'carbon': 6}

for llave, valor in elementos.items():
    print(llave, " = ", valor)

# Obteniendo sólo las llaves
for llave in elementos.keys():
    print(llave)

# Obteniendo sólo los valores
for valor in elementos.values():
    print(valor)

# Usando enumerate para obtener índice
for idx, x in enumerate(elementos):
    print("El índice es: {} y el elemento: {}".format(idx, x))

# Else en ciclos for
def cuenta_idiom(limite):
    for i in range(limite, 0, -1):
        print(i)
    else:  # Corresponde al for, NO al IF
        print("Cuenta finalizada")

cuenta_idiom(3)

# Else no se ejecuta si se rompe el ciclo
def cuenta_idiomv2(limite):
    for i in range(limite, 0, -1):
        print(i)
        if i == 3:
            break  # rompe el ciclo
    else:  # Corresponde al FOR, NO al IF
        print("Cuenta finalizada")

cuenta_idiomv2(5)
```

```
***  5
     4
     3
     2
     1
     24
     720
     1
     2
     3
     4
     5
     0
     1
     2
     3
     4
     -5
     -4
     -3
     -2
     -1
     0
     1
     uno
     dos
     tres
     cuatro
     hidrogeno  =  1
     helio  =  2
     carbon  =  6
     hidrogeno
     helio
     carbon
     1
     2
     6
     El índice es: 0 y el elemento: hidrogeno
     El índice es: 1 y el elemento: helio
     El índice es: 2 y el elemento: carbon
     3
     2
     1
     Cuenta finalizada
     5
     4
     3
```

12. Uso de Bibliotecas

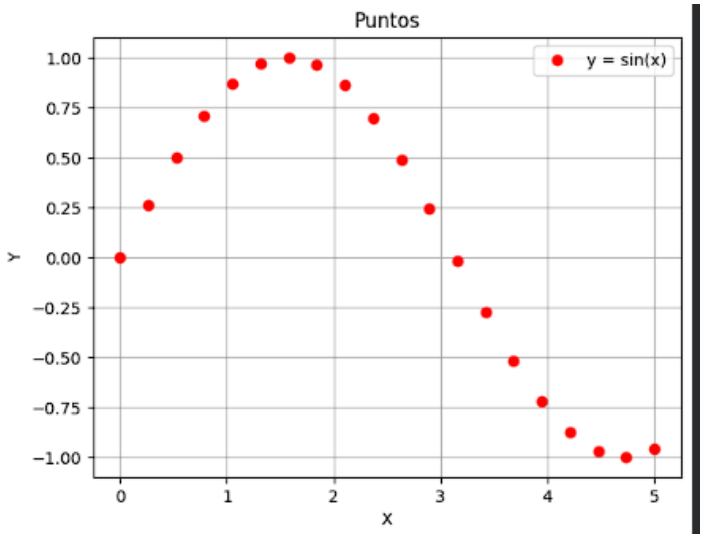
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']

```python
import math

# Una vez que la biblioteca está importada, se pueden conocer las funciones que éste contiene
print(dir(math))
# Para conocer cómo utilizar las funciones, se puede utilizar la función help
# help(math.log)
# Se puede definir un alias para llamar a las funciones
import math as ma

x = ma.cos(ma.pi)
print(x)

# Usando funciones específicas de math
print("Pi:", math.pi)
print("Raíz cuadrada de 16:", math.sqrt(16))
print("Factorial de 5:", math.factorial(5))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh
-1.0
Pi: 3.141592653589793
Raíz cuadrada de 16: 4.0
Factorial de 5: 120
```

13. GRAFICACION (si funciona lol)

```python
# Ejemplo de entrada de datos (ejecutar desde terminal)
# Importando las bibliotecas
import matplotlib.pyplot as plt
from numpy import linspace, sin

# Datos de entrada
x = linspace(0, 5, 20)  # Generando 20 puntos entre 0 y 5

fig, ax = plt.subplots(facecolor='w', edgecolor='k')

ax.plot(x, sin(x), marker="o", color="r", linestyle='None')

ax.grid(True)
ax.set_xlabel('X')      # Etiqueta del eje x
ax.set_ylabel('Y')      # Etiqueta del eje y
ax.grid(True)
ax.legend(["y = sin(x)"])

plt.title('Puntos')
plt.show()

fig.savefig("grafica.png")  # Guardando la gráfica
```



14. Ejemplo de Entrada de Datos

```
# Se pide el nombre al usuario
print("Hola, ¿cómo te llamas?")
# Se leen los datos introducidos por el usuario y se asignan a la variable nombre
nombre = input()
# Se escribe el nombre solicitado
print("Buen día {}".format(nombre))
```

```
Hola, ¿cómo te llamas?
Alan Yael
Buen día Alan Yael
```

```
print("--Calculadora--") #Opciones para el usuario
print("1- Sumar")
print("2- Restar")
print("3- Multiplicar")
print("4- Dividir")
print("5- Salir")

# Se solicita que el usuario especifique alguna operación
op = int(input('Opcion: '))
```

```
--Calculadora--
1- Sumar
2- Restar
3- Multiplicar
4- Dividir
5- Salir
Opcion: 3
```