------------------------------------------------------------------------------------------------------------ **PYTHON 3** ----------------------------------------------------------------------------------------------------------------------

Palabras reservadas: **and, as, assert, break, class, continue, def, del, elif,**

**else, except, exec, finally, for, from, global, if, import, in, is, lambda, not,**

**or, pass, print, raise, return, try, while, with, yield.**

**#** Comentarios en línea **"""** Comentarios en bloque (dobles o sencillas) **"""**

------------------------------------------------------------------ VARIABLES (Tipo número: entero, flotante, complejo) ---------------------------------------------------------------------------

**#Inicializando variables**

x = 10 **#variable de tipo número entero**

**#función para imprimir**

print( x )

**#Asigna un mismo valor a tres variables**

x = y = z = 10

print( x, y, z ) **# 10 10 10**

**#type() permite conocer el tipo de una variable**

print( type( x ) ) **# int**

print( type( 3.14146 ) ) **# float**

**#Cambio del tipo de variable**

x = 18.5

print( type( x ) ) **# float**

x = 3+5j **# ‘j’ sirve para indicar que es complejo**

print( type( x ) ) # **complex**

**#Constante, por convención, nombre en mayúsculas.**

SEGUNDOS\_POR\_DIA = 60 \* 60 \* 24

PI = 3.14

------------------------------------------------------------------------------------- CADENAS (Iterable) -----------------------------------------------------------------------------------------

**#Comillas dobles o simples para asignar e inicializar una cadena**

cadena1 = 'Hola '

cadena2 = "Mundo"

print( cadena1 ) **# Hola**

print( cadena2 ) **# Mundo**

**#Concatenación de cadenas**

concat\_cadenas = cadena1 + cadena2

print(concat\_cadenas) **# Hola Mundo**

**#Para concatenar número y cadena se usa “*str()*”**

num\_cadena = concat\_cadenas + ' '+ str( 3 ) **#**Cadena vacía para espacio

print( num\_cadena ) **# Hola Mundo 3**

**#El valor de la variable se va a imprimir en el lugar donde se encuentre ‘{}’ en la cadena. Además de utilizar la función ‘*format*’.**

num\_cadena ="{} {} {}".format( cadena1, cadena2, 3)

print( num\_cadena ) **# Hola Mundo 3**

**#Cuando se agrega un número dentro de {#}, el valor de la variable que se encuentra en esa posición dentro de la función format(), será impreso.**

num\_cadena = "Cambio: {1} {2} {0}".format(cadena1, cadena2, 3)

print(num\_cadena) **# Cambio: Mundo 3 Hola**

------------------------------------------------------------------------------- OPERADORES -----------------------------------------------------------------------------------------

**#Aritméticos: +, -, \*, /, \*\*, %**

print( 20 \* 3 + (9 + 1) / 10 - 2 ) **# 59**

print( 7 % 4 ) **# 3**

print( 5\*\*3 ) **# 125 #Exponente**

**#Asignación: +=, -=, \*=, /=, \*\*=, %=**

a = b = 7

a/=b **# Abreviación de y = y / x**

print (a) **# 1**

**#Comparación: >, <, >=, <=, ==, !=**

print (7 < 5) **# False**

print (7 > 5) **# True**

print ((11 \* 3)+2 == 36 - 1) **# True**

print ((11 \* 3)+2 >= 36) **# False**

print ("curso" != "CuRsO") **# True**

**#Booleanos: and, not, or**

print (7 > 5 and 3 != 2) **# True**

---------------------------------------------------------------------------------- LISTAS (Iterable. Mutable) -----------------------------------------------------------------------------------------

**#Declaración de una lista simple**

lista\_diasDeMeses = [31,28,31,30,31,30,31,31,30,31,30,31]

print (lista\_diasDeMeses) **# [31, 28, 31, 30, 31, 30, 31, 31...**

print (lista\_diasDeMeses[0]) **# 31**

print (lista\_diasDeMeses[6]) **# 31**

print (lista\_diasDeMeses[11]) **# 31**

------------------------------------------------------------------------- LISTAS ANIDADAS -----------------------------------------------------------------------------------------

ista\_numeros = [ [ 'cero', 0 ], [ 'uno', 1, 'UNO' ], [ 'X', 2 ], [ 'tres', 3 ] ]

print (lista\_numeros) **# [['cero', 0], ['uno', 1, 'UNO'], ['X', 2], ['tres', 3]]**

print (lista\_numeros[0]) **# ['cero', 0]**

print (lista\_numeros[1]) **# ['uno', 1, 'UNO']**

print (lista\_numeros[2][0]) **#X**

print (lista\_numeros[2][1]) **# 2**

**#Cambiando el valor de uno de los elementos**

lista\_numeros[2][0] = "Dos"

print (lista\_numeros[2]) **# ['Dos', 2]**

----------------------------------------------------------------------------- TUPLAS (Iterable. No mutable) --------------------------------------------------------------------------------------

**#Declaración de una tupla**

tupla\_diasDelMes=(31,28,31,30,31,30,31,31,30,31,30,31)

print (tupla\_diasDelMes) **# (31, 28, 31, 30, 31, 30, 31, 31...)**

print (tupla\_diasDelMes[0]) **# 31**

print (tupla\_diasDelMes[3]) **# 30**

print (tupla\_diasDelMes[1]) **# 28**

----------------------------------------------------------------------- TUPLAS ANIDADAS -----------------------------------------------------------------------------------------

**#Declaración de tuplas anidadas**

tupla\_numeros = (('cero', 0),('uno',1, 'UNO'), ('dos',2))

print (tupla\_numeros) **# (('cero', 0), ('uno', 1, 'UNO'), ('dos', 2))**

print (tupla\_numeros[0]) **# ('cero', 0)**

print (tupla\_numeros[1]) **# ('uno', 1, 'UNO')**

print (tupla\_numeros[2][0]) **# dos**

print (tupla\_numeros[2][1]) **# 2**

print (tupla\_numeros[1][0]) **# uno**

print (tupla\_numeros[1][1]) **# 1**

print (tupla\_numeros[1][2]) **# UNO**

--------------------------------------------------------------------- TUPLA CON NOMBRE (Tipo especial) ----------------------------------------------------------------------------------

from collections import namedtuple **# Se agrega librería**

**#Nombre de la tupla, y campos**

planeta = namedtuple('planeta', ['nombre', 'numero'])

**#planeta es la referencia a la tupla**

planeta1 = planeta('Mercurio', 1)

print(planeta1) **# planeta(nombre='Mercurio', numero=1)**

**#Se crea el planeta 2**

planeta2 = planeta('Venus', 2)

**#Se imprimen los valores de los campos. Referencia llama campo**

print(planeta1.nombre, planeta1.numero) **# Mercurio 1**

**#Se obtienen los valores por el orden de los campos**

print(planeta2[0], planeta2[1]) **# Venus 2**

print('Campos de la tupla: {}'.format(planeta1.\_fields)) **# Campos de la tupla: ('nombre', 'numero')**

------------------------------------------------------- CONJUNTO (Sin elementos duplicados. Colección desordenada) -------------------------------------------------------

basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}

print(basket) **# {'orange', 'pear', 'apple', 'banana'}**

print('orange' in basket) **# True**

print('raspberry' in basket) **# False**

---------------------------------------------------------------------------------- DICCIONARIOS (Llaves inmutables. Colección desordenada) -------------------------------------------------------------

**#Creando un diccionario**

elementos = { 'hidrogeno': 1, 'helio': 2, 'carbon': 6 }

**#Al momento de la impresión, pueden aparecer en diferente orden**

print (elementos) **# {'hidrogeno': 1, 'helio': 2, 'carbon': 6}**

print (elementos['hidrogeno']) **# 1**

**#Se pueden agregar elementos al diccionario**

elementos['litio'] = 3

elementos['nitrogeno'] = 8

print (elementos) **# {'litio': 3, 'hidrogeno': 1, 'helio': 2, 'carbon': 6, 'nitrogeno': 8}**

**#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) **# {'H': {'number': 1, 'weight': 1.00794, 'name': 'Hydrogen'}, 'He': {'number': 2, 'weight': 4.002602, 'name': 'Helium'}}**

**#Imprimiendo los datos de un elemento del diccionario**

print (elementos2['H']) **# {'number': 1, 'weight': 1.00794, 'name': 'Hydrogen'}**

print (elementos2['H']['name']) **# Hydrogen**

print (elementos2['H']['number']) **# 1**

elementos2['H']['weight'] = 4.30 **#Cambiando el valor de un elemento**

print (elementos2['H']['weight']) **# 4.3**

**#Agregando elementos a una llave**

elementos2['H'].update({'gas noble':True})

print (elementos2['H']) **# {'number': 1, 'gas noble': True, 'weight': 4.3, 'name': 'Hydrogen'}**

**#Muestra todos los elementos del diccionario**

print (elementos2.items()) **# dict\_items([('H', {'number': 1, 'gas noble': True, 'weight': 4.3, 'name': 'Hydrogen'}), ('He', {'number': 2, 'weight': 4.002602, 'name': 'Helium'})])**

**#Muestra todas las llaves del diccionario**

print (elementos2.keys()) **# dict\_keys(['H', 'He'])**

------------------------------------------------------- FUNCIONES (observar indexado al definir función) -----------------------------------------------------------------------

def imprime\_nombre(nombre):

print("hola " + nombre) **#Cadenas se pueden concatenar con el +**

**#Llamada a la función**

imprime\_nombre("JJ") **# hola JJ**

**#Definiendo una función que regresa el cuadrado de un número**

def cuadrado(x):

return x\*\*2

x = 5

print("Cuadrado de {} es {}".format(x, cuadrado(x))) **# Cua... de 5 es 25**

**#Definiendo una función que regrese más de un valor**

def varios(x):

return x\*\*2, x\*\*3, x\*\*4

**#Los valores devueltos pueden guardarse en variables separadas por ,**

val1, val2, val3 = varios(2)

print("{} {} {}".format(val1, val2, val3)) **# 4 8 16**

**#Función con un parámetro con un valor por defecto**

def cuadrado\_default(x=3):

return x\*\*2

**#Como la función tiene un valor por *default*, si se llama la función sin especificar el parámetro, se toma el que tiene por defecto**

cuadrado\_default() **# 9**

**#Si una función regresa más de una variable, puede usar el operando '\_', para no guardar un valor no deseado.**

**#La función regresa tres valores, sólo interesa el primero y el tercero**

val4, \_, val5 = varios(2)

print("{} y {}".format(val4, val5)) **# 4 y 16**

------------------------------------------------------------------------------ VARIABLES GLOBALES -----------------------------------------------------------------------------------------

vg = 'Global' **#crea una variable global**

def funcion\_v1(): **#Función que imprimirá la variable global**

print(vg)

funcion\_v1() **# Global #Desde función**

print(vg) **# Global**

def funcion\_v2(): **#Función que crea variable con mismo nombre**

vg = "Local"

print(vg)

funcion\_v2() **# Local**

print(vg) **# Global**

**#La siguiente función trata de imprimir la variable global pero existe una variable local con el mismo nombre. Habrá error de referencia.**

def funcion\_v3():

print(vg)

vg = "Local"

print(vg)

funcion\_v3() **#Genera un error**

def funcion\_v4(): **#Solución**

global vg **# Utilizar palabra reservada *global***

print(vg) **# Global**

vg = "Local"

print(vg) **# Local**

funcion\_v4() **# Global # Local**

print(vg) **# Local**

------------------------------------------------ CONSIDERACIONES PARA “ITERABLES” (cadenas, listas, tuplas) ---------------------------------------------

cadena = 'Hola, Mundo'

print( cadena[ 0 ] ) **# H # Imprime el primer caracter**

print( cadena[ 2:5 ] ) **# la, # (op rango) Imprime del 3ero hasta el 5to**

print( cadena[ 2: ] ) **# la, Mundo # Imprime a partir del 3ro**

print( cadena \* 2 ) **#** **... # (op repetición) Imprime la cadena dos veces**

print( cadena[ -1 ] ) **# o # Imprime el último caracter**

------------------------------------------------- FUNCIONES PARA LISTAS -----------------------------------------------------------------

lista = [1, 2, 3]

print( len( [1, 2, 3] ) ) **# 3 # “len” Tamaño de una lista**

lista = lista + [3, 4, 5] **# Se agranda la lista con otros 3 elementos**

print( lista ) **# [1, 2, 3, 3, 4, 5]**

print( 4 in lista ) **# True # Operador ‘in’, indica membresía**

print( 1 not in lista ) **# False**

print( max( lista ) ) **# 5 # “max” obtiene el máximo en la lista**

print( min( lista ) ) **# 1 # “min” obtiene el mínimo en la lista**

print( lista.count( 3 ) ) **# 2 # “count” Obtiene el número de ocurrencias**

otraLista = [1, 'Gato', 5.6]

otraLista.append( "Perro" ) **# “append” agrega el elemento al final**

print( otraLista ) **# [1, 'Gato', 5.6, "Perro" ]**

otraLista.insert( 0, "Perico" ) **# “insert” agrega elemento según índice**

print( otraLista ) **# [ "Perico", 1, 'Gato', 5.6, "Perro" ]**

otraLista.pop() **# “pop” elimina el elemento que está al final de la lista**

otraLista.pop( 2 ) **# “pop()” elimina el elemento según índice**

print( otraLista) **# [ "Perico", 1, 5.6 ]**

otraLista.remove( 5.6 ) **# “remove()” elimina el elemento**

print( otraLista) **# [ "Perico", 1 ]**

print( otraLista.index( 'Perico' ) ) **# 0 # “index” Índice de 1er ocurrencia**

**# Para usar lista como cola**

from collections import deque **# Se agrega librería**

cola = deque(["Eric", "John", "Michael"]) **# Usar “deque”**

cola.append("Terry") **# Se agrega al final de la lista**

print(cola) **# deque(['Eric', 'John', 'Michael', 'Terry'])**

cola.popleft() **# ‘Eric’ # “popleft” elimina con política FIFO**

cola.popleft() **# ‘John’**

print(cola) **# deque(['Michael', 'Terry'])**

------------------------------------------------- LIBRERÍAS -----------------------------------------------------------------

import math

print( math.sqrt( 2 ) ) **#1.4142**

import random

print( random.randrange(10, 15) ) **#Aleatorio de 10 a 15**