

Tutorial básico de Python 1

Ascii

https://elcodigoascii.com.ar/#google_vignette

<https://www.ibm.com/docs/es/aix/7.1.0?topic=adapters-ascii-decimal-hexadecimal-octal-binary-conversion-table>

Variables

Tipos de datos:

Text Type: **str**

Numeric Types: **int, float, complex**

Sequence Types: **list, tuple, range**

Mapping Type: **dict**

Set Types: **set, frozenset**

Boolean Type: **bool**

Binary Types: **bytes, bytearray, memoryview**

None Type: **NoneType**

x = str(3) **Cadena**

y = int(3) **Número entero**

z = float(3) **Número de coma flotante, número real con decimales**

g= 2+3j **Número complejo. Complex**

x = ["apple", "banana", "cherry"] **lista** **list1 = ["abc", 34, True, 40, "male"]**

y = ["apple", "banana", "cherry"]

y.append("orange")

print(y) **→ ["apple", "banana", "cherry", "orange"]**

x = ("apple", "banana", "cherry") **tuple, en español tupla o tipo de dato.**

x = "apple", "banana", "cherry" **tuple, en español tupla o tipo de dato.**

estetuple = ("apple", "banana", "cherry")

y = ("orange",)

estetuple += y

print(estetuple)

→ ("apple", "banana", "cherry", "orange")

x = range(10) **rango o range**

print(x) **→ range(0, 10)**

print(list(x)) **→ [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

x = {"name" : "John", "age" : 36} **dict, es la abreviatura de diccionario.**

x = {"apple", "banana", "cherry"} **set, estructura de datos que permite almacenar una colección de elementos únicos y desordenados**

x = frozenset({"apple", "banana", "cherry"}) **frozenset es un tipo de conjunto inmutable. Funciona igual que un conjunto normal (set), pero no se puede modificar después de crearlo**

x = True **bool o booleano, "True" o "False"**

x = b"Hello" **byte o un literal de bytes**

x = b"Hello"

print(x) **→ b'Hello'**

print(type(x)) **→ <class 'bytes'>**

x = bytearray(5) **bytearray o array de bytes**

x = bytearray(5)

print(x) **→ bytearray(b'\x00\x00\x00\x00\x00')**

print(type(x)) **→ <class 'bytearray'>**

Tutorial básico de Python 2

```
x = memoryview(bytes(5))  Un memoryview en Python es un objeto incorporado que te
                           permite acceder a los datos internos de otro objeto (como bytes, bytearray o
                           array.array) sin necesidad de copiar los datos en la memoria
                           print(x)           →           <memory at 0x01368FA0>
                           print(type(x))      →           <class 'memoryview'>
x = None  None. es un valor especial que representa la ausencia de un valor o un
          dato vacío
          print(x)           →           None
          print(type(x))      →           <class 'NoneType'>
```

Se puede averiguar el tipo de dato que es con la siguiente función:

```
print(type(x))
print(type(y))
print(type(z))
```

Otro ejemplo

```
x = 20.5
#display x:
print(x)
#display the data type of x:
print(type(x))
```

Nombres de variables:

```
myvar = "Aba"
my_var = "Aba"
_my_var = "Aba"
myVar = "Aba"
MYVAR = "Aba"
myvar2 = "Aba"
```

Otras formas de declarar las variables

Forma simplificada de asignar valores

```
x, y, z = "Orange", "Banana", "Cherry"
x = y = z = "Orange"
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
```

variables globales. Se pueden incluir en una función. Y valer para todos.

```
global x
x = "fantastic"
```

Conversión entre tipo de datos.

```
x=1
print (type(x))      <class 'int'>
x = float(x)
print (type(x))      <class 'float'>
thislista = list(("gato", "perro", "zorro"))
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
```

Variable tipo diccionario dict

Tutorial básico de Python 3

```
coche1 = {
    "marca": "Opel",
    "modelo": "Astra",
    "año": 2014
}
x = coche1.get("marca")
y = coche1.values()
z = coche1.items()
d = coche1.copy()
e = dict(coche1)
coche1["color"] = "Gris metalizado"
print(coche1["marca"])      → Opel
print(coche1["modelo"])    → Astra
print(coche1["año"])       → 2014
print(x)                   → Opel
print(coche1.keys())       → dict_keys(['marca', 'modelo', 'año', 'color'])
print(y)                   → dict_values(['Opel', 'Astra', 2014, 'Gris metalizado'])
print(z)                   → dict_items([('marca', 'Opel'), ('modelo', 'Astra'), ('año',
2014), ('color', 'Gris metalizado')])
print(d)                   → {'marca': 'Opel', 'modelo': 'Astra', 'año': 2014, 'color':
'Gris metalizado'}

print(e)                   → {'marca': 'Opel', 'modelo': 'Astra', 'año': 2014, 'color':
'Gris metalizado'}
```

diccionarios anidados dict nest

```
mifamilia = {
    "niño1" : {
        "nombre" : "Juan",
        "edad" : 2004
    },
    "niño2" : {
        "nombre" : "Luisa",
        "edad" : 2007
    },
    "niño3" : {
        "nombre" : "Laura",
        "edad" : 2011
    }
}
print (mifamilia)          {'niño1': {'nombre': 'Juan', 'edad': 2004}, 'niño2': {'nombre':
'Luisa', 'edad': 2007}, 'niño3': {'nombre': 'Laura', 'edad': 2011}}
```

Funciones estándar

“print” imprimir en pantalla.

Dos formas de imprimir

```
print ("Esto es una prueba")
print ('Esto es una prueba')
```

Para que no haya un salto de línea entre dos “print”

```
print("Hello World!", end=" ")
print("I will print on the same line.")
```

Otra forma de mostrar variables

```
x = "Python "
y = "es "
z = "guay"
print(x + y + z)
```

“for” sirve para hacer procesos repetitivos, por ejemplo impresión de datos

```
frutas = ["platano", "manzana", "higo"]
for x in frutas:
    print(x)
```

```
for x in "otorrinolaringólogo":
    print(x)
```

```
frutas = ["platano", "manzana", "higo"]
for x in frutas:
    if x=="higo":
        break
    print(x)
```

```
frutas = ["platano", "manzana", "higo"]
for x in frutas:
    if x=="higo":
        continue # Se salta la rutina hasta el siguiente
    print(x)
```

```
for x in range(6):
    print(x)          # Imprime del 0 al 5
```

```
for x in range(2, 6):
    print(x)          # Imprime del 2 al 5
```

```
for x in range(2, 30, 3):
    print(x)          # Imprime del 2 al 29 de 3 en 3.
```

```
for x in range(6):
    if x == 3: break
    print(x)
else:
    print("Finally finished!")
```

Tutorial básico de Python 5

Si el bucle se interrumpe, el bloque **else** no se ejecuta.

```
color = ["rojo", "grande", "sabroso"]
frutas = ["manzana", "platano", "cereza"]
for x in color:
    for y in frutas:
        print(x, y)
```

```
for x in [0, 1, 2]:
    pass
```

having an empty for loop like this, would raise an error without the pass statement

“len” sirve para averiguar el tamaño o número de elementos de una cadena o una lista...

```
palabra = "caracola"
print(len(palabra))                → 8
lista = ["cara", "nariz", "oreja", "ojo", "ceja"]
print(len(lista))                  → 5
```

“if” establece una condición

```
estatupe = ("apple", "banana", "cherry")
if "apple" in estatupe:
    print("Yes, 'apple' is in the fruits tuple")
```

“input” entrada de datos.

El dato de salida es un **'str'**

Un pequeño algoritmo para saber si es o no es un número:

```
a=input("Pon un número ")
print(a.isdigit())
print(type(a))
if a.isdigit() == False:
    print("No es un numero")
else: print("Es un numero")
```

Funciones creadas

Primero se define la función:

```
def funcionmia ():  
    x="maravilloso"  
    print ("Python es ",x)
```

Para ejecutar la función se pone:

```
funcionmia ()
```

Ejemplo de dos valores para x

```
x = "awesome"  
def myfunc():  
    x = "fantastic"  
    print("Python is " + x)
```

En el primer código usa la variable local, en la segunda la global.

```
myfunc()  
print("Python is " + x)
```

Librerías

Hay que usar el comando “**import**” y el nombre de la librería al principio del código Python

Ejemplo

```
import random
import sys
```

Tipos de librerías

random: generador de números y datos aleatorios

Ejemplo 1

```
import random
D = random.choice(("Juan","Pepe","Pedro"))
print (D)
```

Ejemplo 2

```
import random
print(random.randrange(1, 10)) # Número aleatorio entre 1 y 10
```

sys: datos del sistema

Ejemplo

```
import sys
D = sys.version
print (D) # 3.12.3 (main, Apr 10 2024, 05:33:47) [GCC 13.2.0]
```

math: módulo matemático

Algunas de las funciones que encontrarás en el módulo matemático:

sin() Seno de , que debe estar expresado en radianes.

cos() Coseno de , que debe estar expresado en radianes.

tan() Tangente de , que debe estar expresado en radianes.

exp() El número elevado a .

ceil() Redondeo hacia arriba de (en inglés, «ceiling» significa techo).

floor() Redondeo hacia abajo de (en inglés, «floor» significa suelo).

log() Logaritmo natural (en base) de .

log10() Logaritmo decimal (en base 10) de .

sqrt() Raíz cuadrada de (del inglés «square root»).

round() redondeo un número con decimales a número entero.

pow(x,y) potencia x^y

El valor de x de: sin(x), cos(x), tg(x), debe estar en radianes

Pi radianes son 180 grados.

```
import math
from math import sin
from math import cos
from math import tan
from math import pi
a = sin (math.pi)
b = cos (0)
c = tan(0.5)
d = math.sin (math.pi)
print (a,b,c,d)
```

platform: módulo sobre el sistema operativo

```
import platform
x = platform.system()
print(x) → Windows
```

```
import platform
x = dir(platform)
print(x)
```

datetime : módulo sobre fecha, hora....

```
import datetime
x = datetime.datetime.now()
print(x) → 2026-09-10 20:30:35.731218
```

```
import datetime
x = datetime.datetime.now()
print(x.year) → 2026
print(x.strftime("%A")) → Thursday
```

```
import datetime
x = datetime.datetime(2020, 5, 17)
print(x) → 2020-05-17 00:00:00
```

```
import datetime
x = datetime.datetime(2018, 6, 1)
print(x.strftime("%B")) → June
```

módulos: es crear tu propio módulo

Ejemplo:

guarda este código en `abrahammod.py`

```
persona1 = {
    "name": "Abraham",
    "edad": 52,
    "country": "España"
}
```

Importa el modulo llamado `abrahammod.py`

```
import abrahammod
a = abrahammod.person1["edad"]
print(a)
```

Otra forma de hacerlo

```
import abrahammod as suf
a = suf.person1["edad"]
print(a)
```

Webs de los que se ha extraído parte de la información.

<https://www.w3schools.com/python/default.asp>