Creado por:

Isabel Maniega

Tuplas

In [1]:
```
a = (1, 3, 4, 8) # 0, 1, 2, 3...
a
```

Out[1]:
```
(1, 3, 4, 8)
```

In [2]:
```
a[2]
```

Out[2]:
```
4
```

In [3]:
```
a[-1]
```

Out[3]:
```
8
```

In [4]:
```
# apendizar valors en tuplas
a.append(8)
a
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[4], line 2
      1 # apendizar valors en tuplas
----> 2 a.append(8)
      3 a

AttributeError: 'tuple' object has no attribute 'append'
```

In [5]:
```
# Modificar valores:
a[0] = 10
a
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 2
      1 # Modificar valores:
----> 2 a[0] = 10
      3 a

TypeError: 'tuple' object does not support item assignment
```

In [6]:
```
# Poner siempre una coma al final para asegurarnos que es una tupla,
# NO un entero (int)

b = (10,)
print(type(b))
```

```
<class 'tuple'>
```

In [7]:
```
# concatenación de tuplas:
c = a + b
```

In [8]:
```
c
```

Out[8]:
```
(1, 3, 4, 8, 10)
```

- Las tuplas NO se pueden apendizar ni modificar
- Las tuplas se pueden concatenar

Listas

In [9]:
```
# izq > derch: 0, 1, 2,...
# derch > izq: -1, -2, -3, ...
D = [1, 8, 7, 7]
D
```

Out[9]:
```
[1, 8, 7, 7]
```

In [10]:
```
type(D)
```

Out[10]:
```
list
```

In [11]:
```
F = list((c))
F
```

Out[11]:
```
[1, 3, 4, 8, 10]
```

In [12]:
```
F.append(400)
F
```

Out[12]:
```
[1, 3, 4, 8, 10, 400]
```

In [13]:
```
F[0] = 74
F
```

Out[13]:
```
[74, 3, 4, 8, 10, 400]
```

In [14]:
```
G = [1, 2,]
G
```

Out[14]:
```
[1, 2]
```

In [15]:
```
R = F + G
```

In [16]:
```
R
```

Out[16]:
```
[74, 3, 4, 8, 10, 400, 1, 2]
```

In [17]:
```
sum(R)
```

Out[17]: 502

In [18]:
```
max(R), min(R)
```

Out[18]: (400, 1)

Función sort

In [19]:
```
# Ordena los valores de una lista en orden ascendente
listado = [25, 48, 20, 10, 8, 47, 3, 400]
listado.sort()
listado
```

Out[19]: [3, 8, 10, 20, 25, 47, 48, 400]

In [20]:
```
# Ordena los valores de una lista en orden descendente, reverse=True
listado = [25, 48, 20, 10, 8, 47, 3, 400]
listado.sort(reverse=True)
listado
```

Out[20]: [400, 48, 47, 25, 20, 10, 8, 3]

In [21]:
```
# Ordena los valores de una lista en orden ascendente, reverse=False
listado = [25, 48, 20, 10, 8, 47, 3, 400]
listado.sort(reverse=False)
listado
```

Out[21]: [3, 8, 10, 20, 25, 47, 48, 400]

Slicing

In [22]:
```
# Nos permite filtrar valores:

listado = [25, 48, 20, 10, 8, 47, 3, 400]
# Filtramos los valores desde la posicion 0 hasta la posicion 5 NO inclui
listado[0:5]
```

Out[22]: [25, 48, 20, 10, 8]

In [23]:
```
# Filtramos los valores desde la posicion 0 hasta la posicion 5 NO inclui
listado[:5]
```

Out[23]: [25, 48, 20, 10, 8]

In [24]:
```
# Filtramos los valores desde la posicion 2 hasta la posicion 7 NO inclui
listado[2:7]
```

Out[24]: [20, 10, 8, 47, 3]

In [25]:
```
# Filtramos los valores desde la posicion 2 hasta el final:
listado[2:]
```

Out[25]: [20, 10, 8, 47, 3, 400]

In [26]:
```
# Filtramos los valores desde la posicion 0 de 1 en 1:
print(listado)
```

```
In [26]: listado[::1]
```

```
[25, 48, 20, 10, 8, 47, 3, 400]
```

```
Out[26]: [25, 48, 20, 10, 8, 47, 3, 400]
```

```
In [27]: # Filtramos los valores desde la posicion 0 de 2 en 2:
print(listado)
listado[::2]
```

```
[25, 48, 20, 10, 8, 47, 3, 400]
```

```
Out[27]: [25, 20, 8, 3]
```

```
In [28]: # leemos el listado en orden inverso (descendiente):
print(listado)
listado[::-1]
```

```
[25, 48, 20, 10, 8, 47, 3, 400]
```

```
Out[28]: [400, 3, 47, 8, 10, 20, 48, 25]
```

```
In [70]: # Leemos todos los valores:
print(listado)
listado[:]
```

```
[74, 3, 4, 8, 10, 400, 1, 2]
```

```
Out[70]: [74, 3, 4, 8, 10, 400, 1, 2]
```

- Las listas se pueden apendizar y modificar
- Las listas se pueden concatenar

List comprehension

```
In [2]: listado = [74, 3, 4, 8, 10, 400, 1, 2]
listado
```

```
Out[2]: [74, 3, 4, 8, 10, 400, 1, 2]
```

```
In [3]: listado_a = []

for valor in listado:
    if valor >= 8:
        listado_a.append(valor)

listado_a
```

```
Out[3]: [74, 8, 10, 400]
```

```
In [4]: listado_b = [valor for valor in listado if valor >= 8]
listado_b
```

```
Out[4]: [74, 8, 10, 400]
```

```
In [5]: # Es un generador que la variable sólo existe una vez
listado_b = (valor for valor in listado if valor >= 8)
listado_b
```

Out[5]:
```
<generator object <genexpr> at 0x73354314d4d0>
```

In [7]:
```
# Al consumir el generador este desaparece:
for i in listado_b:
    print(i)
```

Bucles

In [34]:
```
# break: romper la iteración.

lista = [5, 8, 9, 7, 6, 4]

for i in lista:
    if i == 7:
        break
    else:
        print(i)
```

```
5
8
9
```

In [35]:
```
# continue: continua la iteración.

lista = [5, 8, 9, 7, 6, 4]

for i in lista:
    if i == 7:
        continue
    else:
        print(i)
```

```
5
8
9
6
4
```

In [36]:
```
# pass: continua la iteración.

lista = [5, 8, 9, 7, 6, 4]

for i in lista:
    if i == 7:
        pass
    else:
        print(i)
```

```
5
8
9
6
4
```

Prints

In [37]:
```
print('Hola', end='-')
print('Mundo')
```

```
Hola-Mundo
```

In [38]:
```
print(1, 2, 3, sep='*')
```

```
1*2*3
```

In [39]:
```
x = 25
print(f'El valor de x es {x}')
print('El valor de x es', x)
print('El valor de x es ' + str(x))
print('El valor de x es %s' %(x))
```

```
El valor de x es 25
El valor de x es 25
El valor de x es 25
El valor de x es 25
```

Asignación y copias en variables

In [40]:
```
x = [1, 5, 7, 6]
y = x
y[0] = 10

print(x) # [10, 5, 7, 6]
print(y) # [10, 5, 7, 6]
print(x == y)
print(x is y)
```

```
[10, 5, 7, 6]
[10, 5, 7, 6]
True
True
```

In [41]:
```
x = [1, 5, 7, 6]
y = x[:]
y[0] = 10

print(x) # [1, 5, 7, 6]
print(y) # [10, 5, 7, 6]
```

```
[1, 5, 7, 6]
[10, 5, 7, 6]
```

In [42]:
```
x = [1, 5, 7, 6]
y = x[:]

print(x) # [1, 5, 7, 6]
print(y) # [1, 5, 7, 6]
print(x == y)
print(x is y)
```

```
[1, 5, 7, 6]
[1, 5, 7, 6]
True
False
```

Strings

Concatenación

In [43]:
```
strings = 'Hola ' + 'Mundo'
strings
```

Out[43]:
```
'Hola Mundo'
```

In [44]:
```
strings = 'Hola'+ ' ' + 'Mundo'
strings
```

Out[44]:
```
'Hola Mundo'
```

Multiplicación

In [45]:
```
# Cuando se multiplica por un número un strings,
# este se repite tantas veces como nos indique el número

repeticion = 2 * 'a'
repeticion
```

Out[45]:
```
'aa'
```

In [46]:
```
repeticion = 10 * 'a'
repeticion
```

Out[46]:
```
'aaaaaaaaaa'
```

join()

In [47]:
```
strings = "Hola Mundo"
strings = '-'.join(strings)
strings
```

Out[47]:
```
'H-o-l-a- -M-u-n-d-o'
```

Signo de escape `\`

In [48]:
```
# \ signo de escape + n (newline)
print('Hola \nMundo')
```

```
Hola
Mundo
```

In [49]:
```
print('Hola')
print('\\')
print('Mundo')
```

```
Hola
\
Mundo
```

In [50]:
```
print('Hola')
print('\\\\')
```

```
print('Mundo')
```

```
Hola
\\
Mundo
```

In [51]:
```
print('Hola')
print('\')
print('Mundo')
```

```
  Cell In[51], line 2
    print('\')
          ^
SyntaxError: unterminated string literal (detected at line 2)
```

Args vs Kwargs

In [52]:
```
# *args: son todos los valores numericos, devueltos como una tupla
# **kwargs: son todos los valores asignados como variables devueltos como

def new_func(x=1, y=2, *args, **kwargs):
    z = x + y
    return z, args, kwargs

new_func(2, 2, 2, 3, 4, singleton=25, decorador='Hola')
```

Out[52]:
```
(4, (2, 3, 4), {'singleton': 25, 'decorador': 'Hola'})
```

Función lambda

In [53]:
```
def funcion_suma(x, y):
    return x + y
```

In [54]:
```
funcion_suma(3, 2)
```

Out[54]:
```
5
```

In [55]:
```
# LAMBDA ES LA FUNCIÓN ANÓNIMA:

(lambda x, y: x + y)(3, 2)
```

Out[55]:
```
5
```

In [56]:
```
funcion_suma = lambda x, y: x + y
funcion_suma(3, 2)
```

Out[56]:
```
5
```

In [57]:
```
funcion_suma(7, 8)
```

Out[57]:
```
15
```

Importar modulos

A) Importar el modulo:

`import math`

In [58]:
```
import math

math.sin(math.pi/2)
```

Out[58]: 1.0

modulo + '.' + entidad

B) Importar el modulo + entidad:

`from math import sin`

- from para importar el modulo
- import importar la entidad

In [59]:
```
from math import sin, pi

sin(pi/2)
```

Out[59]: 1.0

C) Importar el modulo y todas las entidades:

`from math import *`

- from para importar el modulo
- import importar TODO

In [60]:
```
from math import *

sin(pi/2)
```

Out[60]: 1.0

In [61]:
```
dir(math)
```

Out[61]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'comb',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'dist',
 'e',
 'erf',
 'erfc',
 'exp',
 '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',
 'tan',
 'tanh',
```

```
 'tau',
 'trunc',
 'ulp']
```

Importar con un alias

```
import pandas as pd
```

- as: alias para abreviar en el código

In [62]:
```
import pandas as pd

df = pd.DataFrame([1, 2, 4, 8], columns=['Numeros'])
df
```

```
/tmp/ipykernel_107144/3242169067.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major rele
ase of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and b
etter interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/
54466

  import pandas as pd
```

Out[62]:

	Numeros
0	1
1	2
2	4
3	8

Peticiones por teclado

In [63]:
```
input('Dime como te encuentras: ')
```

Out[63]:
```
'Bien, gracias'
```

In [64]:
```
# La variable generada es un string:
texto = input('Dime como te encuentras: ')
texto
```

Out[64]:
```
'Bien, gracias'
```

Convertir ese string en un número

In [65]:
```
numero = int(input('Dime tu número entero favorito: '))
numero
```

Out[65]:
```
2
```

In [66]:
```
print('Mi numero favorito es:', numero)
```

```
Mi numero favorito es: 2
```

Convertir ese string en un decimal

In [67]:
```
decimal = float(input('Dime tu número decimal favorito: '))
decimal
```

Out[67]:
```
1.6
```

In [68]:
```
print('Mi numero favorito decimal es:', decimal)
```

```
Mi numero favorito decimal es: 1.6
```

range()

Nos sirve para generar datos desde un valor de inicio hasta un valor de fin, podemos crear un rango de valores de 1 en 1 o con otro tipo de salto.

In [9]:
```
# range(inicio, fin + 1, step)
rango = range(0, 10, 1)
rango
```

Out[9]:
```
range(0, 10)
```

In [10]:
```
list(rango)
```

Out[10]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [11]:
```
for i in rango:
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [13]:
```
# range(inicio, fin + 1, step)
# el parametro step es opcional, si no se pone es por defecto=1
rango = list(range(1, 11))
rango
```

Out[13]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [14]:
```
# range(inicio, fin + 1, step)
# si no se indica el primer valor cogerá el 0 por defecto
rango = list(range(11))
rango
```

Out[14]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [16]:
```
# range(inicio, fin + 1, step)
# si no se indica el primer valor cogerá el 0 por defecto
rango = list(range(0, 13, 2))
rango
```

Out[16]:
```
[0, 2, 4, 6, 8, 10, 12]
```

random()

In [20]:
```
# valor pseudoaleatorio entre 0 y 1:

import random

print(random.random())
```

```
0.6112915336276639
```

In [23]:
```
# valor pseudoaleatorio entre 0 y 1:

import random

random.seed(0)

print(random.random())
```

```
0.8444218515250481
```

Set

Nos permite a partir de una lista dada obtener los valores únicos:

In [1]:
```
lista = [10, 40, 4, 5, 4, 10, 20, 15, 6, 5, 5, 40, 25]
```

In [2]:
```
# Es un conjunto de datos a nivel matemático, NO CONFUNDIR CON UN DICCION

lista = set(lista)
lista
```

Out[2]:
```
{4, 5, 6, 10, 15, 20, 25, 40}
```

In [3]:
```
# En un set no se permite apendizar valores
lista.append(50)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[3], line 2
      1 # En un set no se permite apendizar valores
----> 2 lista.append(50)

AttributeError: 'set' object has no attribute 'append'
```

In [4]:
```
# No se puedo modificar:
```

```
lista[1] = 100
lista
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 3
      1 # No se puedo modificar:
----> 3 lista[1] = 100
      4 lista

TypeError: 'set' object does not support item assignment
```

In [5]:
```
lista = list(lista)
lista
```

Out[5]:
```
[4, 5, 6, 40, 10, 15, 20, 25]
```

In [6]:
```
# En un set no se permite apendizar valores
lista.append(50)
lista
```

Out[6]:
```
[4, 5, 6, 40, 10, 15, 20, 25, 50]
```

In [7]:
```
# No se puedo modificar:

lista[1] = 100
lista
```

Out[7]:
```
[4, 100, 6, 40, 10, 15, 20, 25, 50]
```

Diccionarios

In [1]:
```
# {'clave': 'valor'}
diccionario = {'clave1': 10, 'clave2': 20, 'clave3': 40}
diccionario
```

Out[1]:
```
{'clave1': 10, 'clave2': 20, 'clave3': 40}
```

In [2]:
```
diccionario['clave1']
```

Out[2]:
```
10
```

In [3]:
```
diccionario['clave3']
```

Out[3]:
```
40
```

In [4]:
```
# Modificar un valor de un clave:

diccionario['clave3'] = 50
diccionario
```

Out[4]:
```
{'clave1': 10, 'clave2': 20, 'clave3': 50}
```

Claves

In [5]:
```
diccionario.keys()
```

Out[5]:
```
dict_keys(['clave1', 'clave2', 'clave3'])
```

In [6]:
```
type(diccionario.keys())
```

Out[6]:
```
dict_keys
```

In [7]:
```
claves = [key for key in diccionario.keys()]
claves
```

Out[7]:
```
['clave1', 'clave2', 'clave3']
```

Valores

In [8]:
```
diccionario.values()
```

Out[8]:
```
dict_values([10, 20, 50])
```

In [9]:
```
type(diccionario.values())
```

Out[9]:
```
dict_values
```

In [10]:
```
valores = [value for value in diccionario.values()]
valores
```

Out[10]:
```
[10, 20, 50]
```

Items

In [11]:
```
diccionario.items()
```

Out[11]:
```
dict_items([('clave1', 10), ('clave2', 20), ('clave3', 50)])
```

In [12]:
```
type(diccionario.items())
```

Out[12]:
```
dict_items
```

In [13]:
```
for key, value in diccionario.items():
    print('clave:', key, 'value:', value)
```

```
clave: clave1 value: 10
clave: clave2 value: 20
clave: clave3 value: 50
```

Agregar nueva clave

In [14]:
```
# Añadir un valor a una clave:

diccionario['clave4'] = 100
diccionario
```

Out[14]:
```
{'clave1': 10, 'clave2': 20, 'clave3': 50, 'clave4': 100}
```

Eliminar nueva clave

In [15]:
```
del diccionario['clave3']
diccionario
```

Out[15]:
```
{'clave1': 10, 'clave2': 20, 'clave4': 100}
```

Vaciar listas, diccionarios,...

In [16]:
```
diccionario.clear()
diccionario
```

Out[16]:
```
{}
```

Ejercicio de certificación

In [17]:
```
datos = [14, {}, [1, 2], (25), 2.8, {"1": 3}, (7,), "team", {4, 8, 20}, [

for dato in datos:
    print(type(dato))
```

```
<class 'int'>
<class 'dict'>
<class 'list'>
<class 'int'>
<class 'float'>
<class 'dict'>
<class 'tuple'>
<class 'str'>
<class 'set'>
<class 'list'>
```

Creado por:

Isabel Maniega