Programación

Introducción a la programación
(Python II)

Programación

Python II

Programación

Editores on line

Editores on line:

- https://www.online-python.com/
- https://www.w3schools.com/python/trypython.asp?filename=demo_compiler
- https://pytwiddle.com

Programación

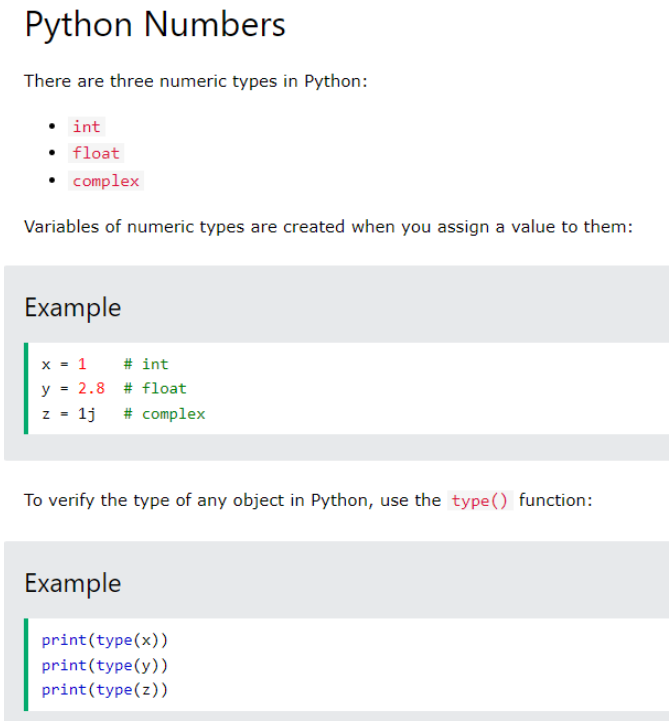
Entornos de desarrollo (IDE)

Entornos de desarrollo (IDE):

- https://www.python.org/
- https://www.jetbrains.com/es-es/pycharm/
- https://code.visualstudio.com/
- https://www.spyder-ide.org/
- https://jupyter.org/

Tipos de datos

- Tipos de datos básicos en python:
  - Numéricos
    - int
    - float
    - complex
  - Booleano
    - Bool
  - Cadenas de caracteres
    - str



Python Numbers

There are three numeric types in Python:

- `int`
- `float`
- `complex`

Variables of numeric types are created when you assign a value to them:

Example

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

To verify the type of any object in Python, use the `type()` function:

Example

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

https://docs.python.org/es/3/library/stdtypes.html

https://blog.hubspot.es/website/tipos-de-datos-python

https://ellibrodepython.com/numeros-complejos

Tipos de datos

- Tipos de datos en python:
  - Cadenas de caracteres
  - Secuencias
    - Listas
    - Tuplas
    - Rangos
  - Iteradores
  - Diccionarios
  - Ficheros

https://docs.python.org/es/3/library/stdtypes.html

https://blog.hubspot.es/website/tipos-de-datos-python

Programación

Índice

- Cadenas de caracteres (String)
- Listas.
- Tuplas.
- Conjuntos.
- Diccionarios.
- Funciones.
- Ficheros.

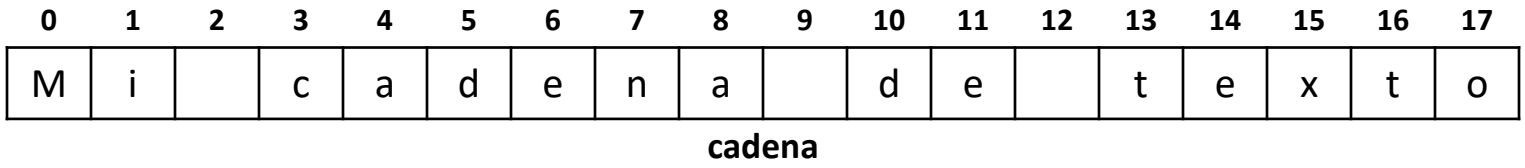
Cadenas de caracteres (String)

Programación

Cadenas de caracteres (Definición)

- Una **cadena de caracteres** es una secuencia de caracteres. Se puede acceder a ellos mediante un índice entero.

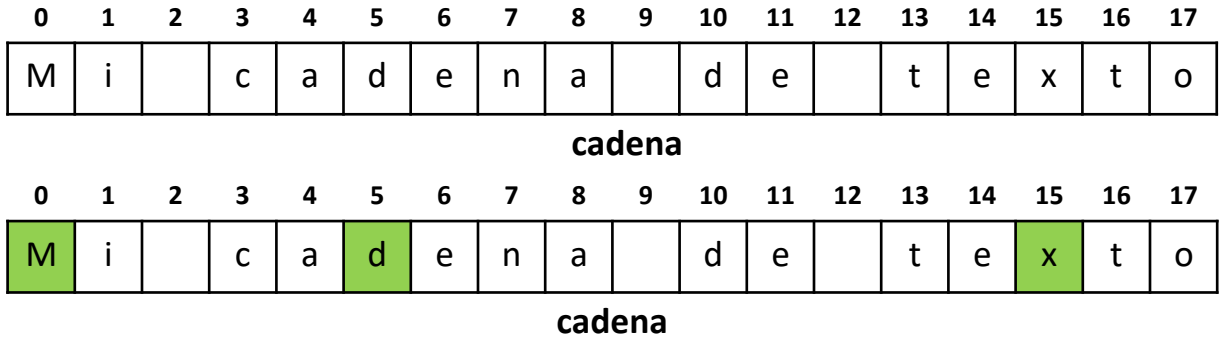
cadena = "Mi cadena de texto"



Programación

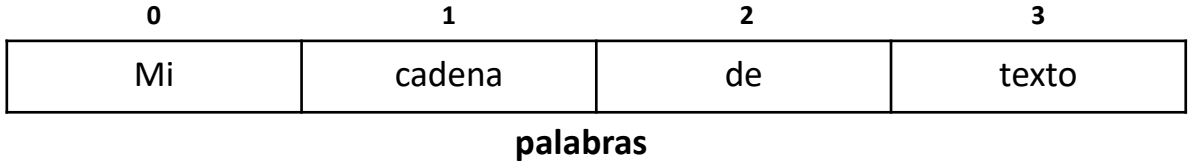
String (I)

```
cadena = "Mi cadena de texto"
print(cadena)

print( cadena[0] )
print( cadena[5] )
print( cadena[15] )

print( len ( cadena ))
```



18

```
palabras=str.split(cadena)
print(palabras)
```

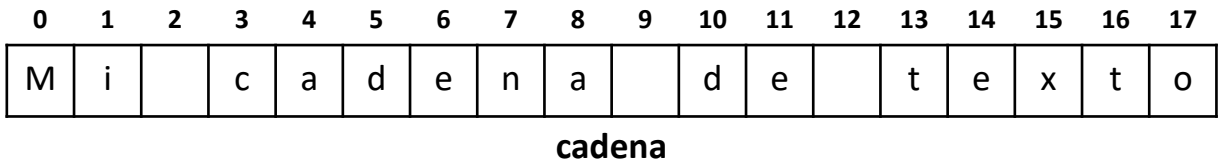


Programación

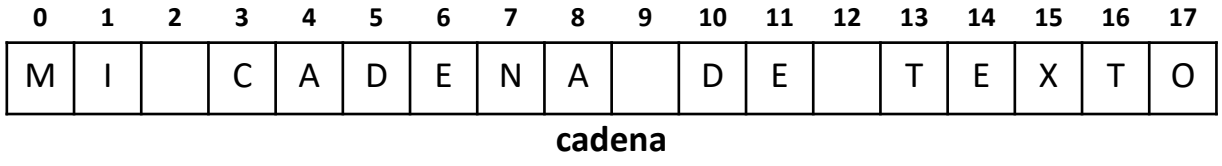
String (II)

```
print("Mi cadena de texto")
print(str.upper("Mi cadena de texto"))
print(str.lower("Mi cadena de texto"))
```

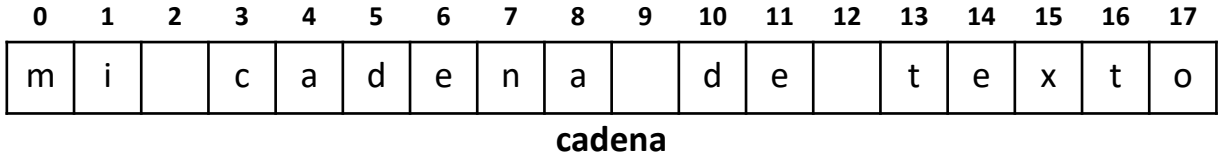
```
cadena = "Mi cadena de texto"
print(cadena)
```




```
cadena = str.upper(cadena)
print(cadena)
```




```
cadena=str.lower(cadena)
print(cadena)
```



Programación

Ejercicio

Ejercicio:
Escribe un programa que pida una cadena de texto al usuario y muestre esa cadena en mayúsculas y minúsculas.

Ejercicio:
Escribe un programa que pida una cadena de texto al usuario y calcule la longitud de la cadena y el número de palabras que la forman.

Ejercicio:
Escribe un programa que pida una cadena de texto al usuario y muestre esa cadena en orden inverso.

Listas

Programación

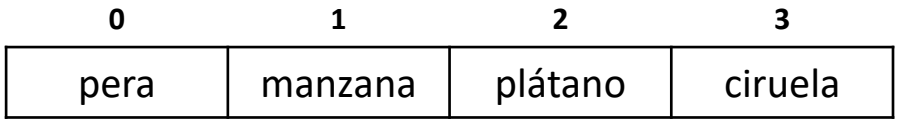
Listas (Definición)

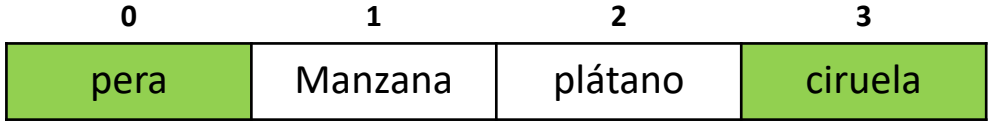
- Una **lista** es una secuencia de valores. En una lista, los valores pueden ser de cualquier tipo y se accede a ellos mediante un índice entero.
- Los elementos de la lista están ordenados, son modificables y permiten valores duplicados.

Programación

Listas (I)

```
#Creando una lista
#frutas = []
frutas = ["pera", "manzana", "plátano", "ciruela"]

#Imprimiendo la lista
print(frutas)

#Acceder a los elementos por índice
print (frutas[0])
print (frutas[3])

#Determinar la cantidad de elementos en la lista
print ( len (frutas) )
```



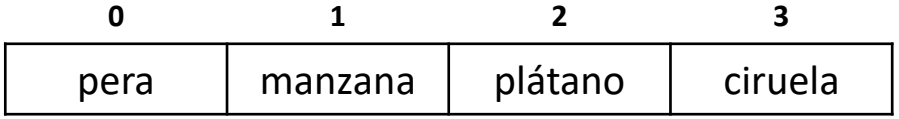


4

Programación

Listas (II)

```
#Creando una lista vacía
frutas = []

#Añadir elementos a la lista
frutas.append ("pera")
frutas.append ("manzana")
frutas.append ("plátano")
frutas.append ("ciruela ")

#Imprimir la lista
print(frutas)

#Determinar la cantidad de elementos en la lista
print ( len (frutas) )
```



4

Programación

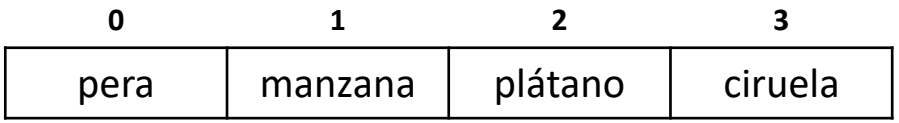
Listas (III)

```
#Creando una lista
frutas = ["pera", "manzana", "plátano", "ciruela"]
print(frutas)
#Añadir elementos a la lista
frutas.append("piña")
print(frutas)
#Eliminar elementos a la lista
frutas.remove ("ciruela")
print(frutas)
frutas.remove (frutas[0])
print(frutas)
#Modificando valores de la lista
frutas[2]="melocotón"
print(frutas)
#Determinar la cantidad de elementos en la lista
print(len(frutas))
```

Programación

Listas (IV)

```
#Creando una lista
#frutas =[]
frutas = ["pera", "manzana", "plátano", "ciruela"]
print(frutas)

#Acceder a los elementos por índice
print (frutas[0])
print (frutas[3])

#Utilizando slices
print (frutas[0:2])
print (frutas[1:3])
print (frutas[:3])
print (frutas[-4:-2])

print ("*" * 50)
print (frutas * 3)
```



https://www.luisllamas.es/python-slices/

Programación

Listas (V)

```
#Creando una lista
frutas = ["pera", "manzana", "plátano", "ciruela"]
print(frutas)

#Recorriendo la lista
for fruta in frutas:
        print(fruta)

#Ordenando la lista
frutas.sort()
print(frutas)

#Ordenando la lista (orden descendente)
frutas.sort(reverse=True)
print(frutas)
```

0	1	2	3
pera	manzana	plátano	ciruela

0	1	2	3
ciruela	manzana	pera	plátano

0	1	2	3
plátano	pera	manzana	ciruela

Programación

Listas (VI)

```
#Crear una lista
#numeros = []
numeros = [1, 5, 7, 3, 2, 10]
print(numeros)
```

0	1	2	3	4	5
1	5	7	3	2	10

```
#Añadir elementos a la lista
numeros.append(9)
print(numeros)
```

0	1	2	3	4	5	6
1	5	7	3	2	10	9

```
#Ordenar la lista
#sorted(numeros)
numeros.sort()
print(numeros)
```

0	1	2	3	4	5	6
1	2	3	5	7	9	10

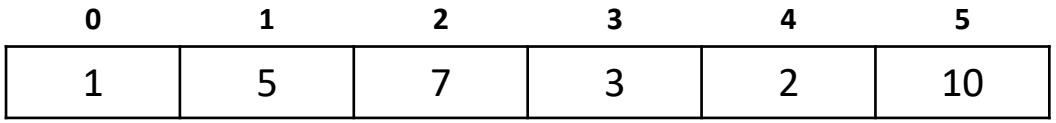
```
#Ordenar la lista en orden descendente
#sorted(numeros, reverse=True)
numeros.sort(reverse=True)
print(numeros)
```

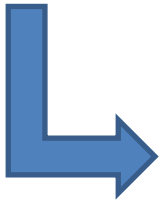
0	1	2	3	4	5	6
10	9	7	5	3	2	1

Programación

Listas (VII)

```
#Crear una lista
numeros = [1, 5, 7, 3, 2, 10]
print(numeros)

suma = 0
#Sumar los elementos de la lista
for numero in numeros:
        suma = suma + numero

print("La suma total de los elementos de la lista es igual a: " + str(suma))
```





28

Programación

Listas (VIII)

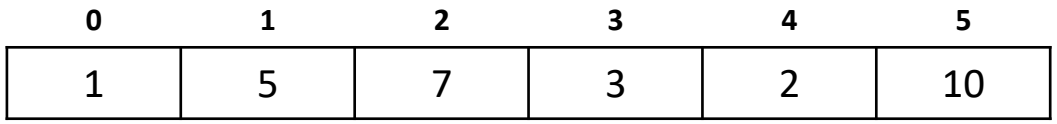
Otros métodos para utilizar con listas::

- **clear()**: elimina todos los elementos de la lista.
- **copy()**: arroja una copia de la lista.
- **count()**: arroja el número de elementos con el valor indicado.
- **extend()**: añade los elementos de una lista (o cualquier iterador) al final de la lista actual.
- **insert()**: añade un elemento en la posición que se indica.
- **pop()**: elimina el elemento de la posición que se indica.
- **reverse()**: invierte el orden de la lista.

Programación

Listas (IX)

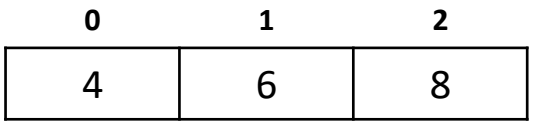
```
#Crear una lista
numeros = [1, 5, 7, 3, 2, 10]
print(numeros)
```




```
for numero in numeros:
        print (numero)
```

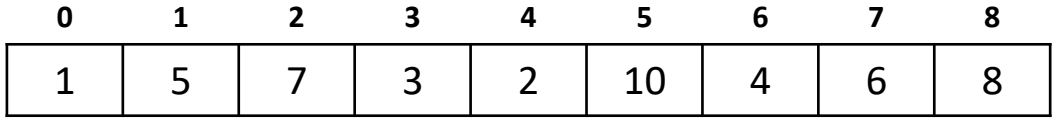
Las listas se recorren utilizando un bucle for

```
numeros2 = [4, 6, 8]
print(numeros2)
```




```
print(numeros + numeros2)
```

```
numeros3 = numeros + numeros2
print(numeros3)
```



Programación

Listas (X)

```
#Crear una lista
numeros = [1, 5, 7, 3, 2, 10]
print(numeros)
```

0	1	2	3	4	5
1	5	7	3	2	10

print (7 in numeros) → True
print (5 in numeros) → True
print (9 in numeros) → False
print (2 in numeros) → True
print (8 in numeros) → False

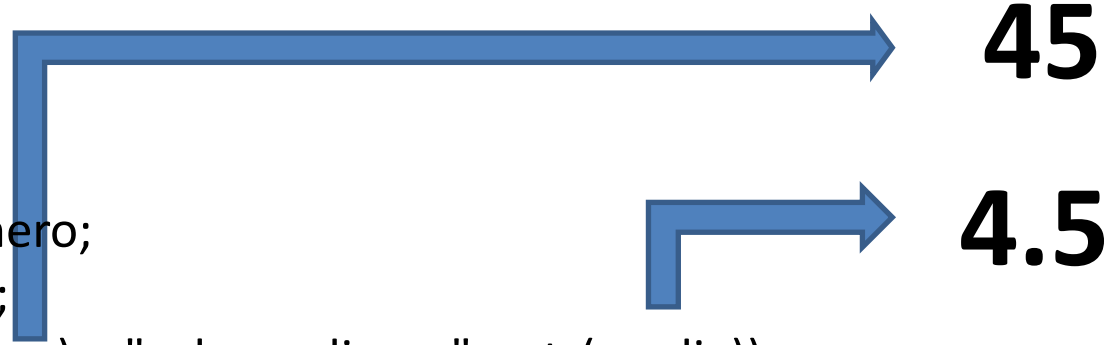
print (9 not in numeros) → True
print (2 not in numeros) → False
print (8 not in numeros) → True

Programación

Listas (XI)

```
#Crear una lista vacía
miLista = [];
#Añadir elementos a la listatos
for i in range (10):
        numero= int(input  ("Introduzca un número entero: "));
        miLista.append(numero);

#Aquí tendríamos una lista con los 10 números enteros

#Calcular la suma y la media
suma =0;
for numero in miLista:
        suma =suma + numero;
media=suma/len (numeros);
print ("La suma es " + str(suma) + " y la media es " + str(media))
```

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



Programación

Ejercicios

Ejercicio:
Crea una lista con al menos seis lenguajes de programación.
Ejercicio:
Crea una lista de números enteros y recórrela indicando si el número correspondiente es par o impar.
Ejercicio:
Recorre la lista anterior cambiando los números impares por el doble de su valor.
Ejercicio:
Crea una lista de números enteros y recórrela cambiando cada valor por su opuesto.

Tuplas

Programación

Tuplas (Definición)

- Una **tupla** es una secuencia de valores. En una tupla, los valores pueden ser de cualquier tipo y se accede a ellos mediante un índice entero. La diferencia con las listas es que las tuplas son inmutables.
- Los elementos de la tupla están ordenados, no son modificables y permiten valores duplicados.
- Una tupla es una colección ordenada e inmutable.

Programación

Tuplas (I)

```
#Creando una tupla
frutas = ("pera", "manzana", "plátano", "ciruela")
print(frutas)

#Acceder a los elementos por índice
print (frutas[0])
print (frutas[3])

#Determinar la cantidad de elementos en la tupla
print(len(frutas))
```

Las tuplas son inmutables (no se pueden modificar)

Programación

Tuplas (II)

```
#Creando una tupla
frutas = ["pera", "manzana", "plátano", "ciruela"]
print(frutas)

#Acceder a los elementos por índice
print (frutas[0])
print (frutas[3])

#Utilizando slices
print (frutas[0:2])
print (frutas[1:3])
print (frutas[:3])
print (frutas[-4:-2])

#repitiendo elementos
print ("*" * 50)
print (frutas * 3)
```

https://www.luisllamas.es/python-slices/

Programación

Tuplas (III)

```
#Creando una tupla
frutas = ("pera", "manzana", "plátano", "ciruela“)
print(frutas)

#Recorriendo la tupla
for fruta in frutas:
        print(fruta)
```

Programación

Tuplas (IV)

```
#Creando una tupla
numeros = (1, 5, 7, 3, 2, 10)
print(numeros)
```

0	1	2	3	4	5
1	5	7	3	2	10

```
#Recorriendo la tupla
for numero in numeros:
        print (numero)
```

Las tuplas se recorren utilizando un bucle for

```
numeros2 = (4, 6, 8)
print(numeros2)
```

0	1	2
4	6	8

```
print(numeros + numeros2)
```

```
numeros3 = numeros + numeros2
print (numeros3)
```

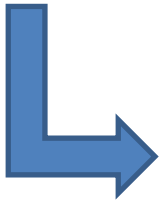
0	1	2	3	4	5	6	7	8
1	5	7	3	2	10	4	6	8

Programación

Tuplas (V)

```
#Crear una tupla
numeros = (1, 5, 7, 3, 2, 10)
print(numeros)

suma = 0
#Sumar los elementos de la tupla
for numero in numeros:
        suma = suma + numero

print("La suma total de los elementos de la lista es igual a: " + str(suma))
```

0	1	2	3	4	5
1	5	7	3	2	10

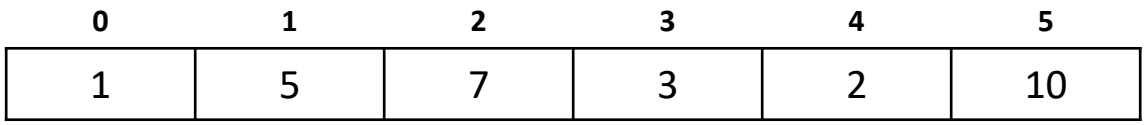


28

Programación

Tuplas (VI)

```
#Crear una tupla
numeros = (1, 5, 7, 3, 2, 10)
print(numeros)
```




```
print (7 in numeros)        → True
print (5 in numeros)        → True
print (9 in numeros)        → False
print (2 in numeros)        → True
print (8 in numeros)        → False

print (9 not in numeros)    → True
print (2 not in numeros)    → False
print (8 not in numeros)    → True
```

Programación

Ejercicios

Ejercicio:
Crea una tupla con al menos seis lenguajes de programación.
Ejercicio:
Crea una tupla de números enteros y recórrela indicando si el número correspondiente es par o impar.
Ejercicio:
Recorre la tupla anterior indicando si el número correspondiente es positivo o negativo.
Ejercicio:
Crea una tupla de números enteros y a continuación pide un número al usuario y comprueba si ese número está o no en la tupla.

Conjuntos

Programación

Conjuntos (Definición)

- Un **conjunto** es una colección de elementos.
- Los elementos del conjunto no son modificables y no se permiten valores duplicados.
- En los conjuntos no se pueden modificar los elementos, pero se pueden eliminar o añadir elementos nuevos.

Programación

Conjuntos (I)

```
#Creando un conjunto
#frutas = {}
frutas = {"pera", "manzana", "plátano", "ciruela"}

#Imprimiendo el conjunto
print(frutas)

#Añadiendo elementos al conjunto
frutas.add ("naranja")
print(frutas)

#Quitando elementos del conjunto
frutas.remove ("manzana")
print(frutas)
```

Programación

Conjuntos (II)

```
#Creando un conjunto
#frutas = {}
frutas = {"pera", "manzana", "plátano", "ciruela"}

#Imprimiendo el conjunto
print(frutas)

#Recorriendo el conjunto
for fruta in frutas:
        print(fruta)

#Determinar la cantidad de elementos en el conjunto
print ( len (frutas) )
```

Los conjuntos se recorren utilizando un bucle for

4

Diccionarios

Programación

Diccionarios (Definición)

- Un **diccionario** es una colección de valores a los que se accede mediante una clave.
- Los elementos del diccionario son modificables y no se permiten duplicados de claves.
- Los elementos del diccionario se presentan en pares clave:valor y se puede hacer referencia a ellos utilizando el nombre de la clave.

Programación

Diccionarios (I)

```
#Crear un diccionario
persona1={"nombre":"Pepe Noveo Mecaigo","edad":17,"curso":"1º Bachillerato"}
print(persona1)
```

	nombre	Edad	curso
persona1	Pepe Noveo Mecaigo	17	1º Bachillerato

```
#Modificar valores
persona1["curso"]="2º Bachillerato"
print(persona1)
```

	Nombre	Edad	curso
persona1	Pepe Noveo Mecaigo	17	2º Bachillerato

```
#Vaciar el diccionario
persona1.clear()
print(persona1)
```

persona1={} (Diccionario vacío)

https://www.datacamp.com/es/tutorial/python-dictionary-append

https://ellibrodepython.com/diccionarios-en-python

Programación

Diccionarios (II)

```
persona1={"nombre":"Pepe Mellamo Nosecomomellamo","edad":17,"curso":"1º Bachillerato"}
persona2={"nombre":"Juan Sin Miedo Valiente","edad":18,"curso":"1º Bachillerato"}

print(persona1)
print(persona1["nombre"])
print(persona1["edad"])
print(persona1["curso"])

print()
print(persona2)
print(persona2["nombre"])
print(persona2["edad"])
print(persona2["curso"])
```

nombre	Edad	curso
Pepe Mellamo Nosecomomellamo	17	1º Bachillerato

persona1

nombre	Edad	curso
Juan Sin Miedo valiente	18	1º Bachillerato

persona2

https://ellibrodepython.com/diccionarios-en-python

Programación

Diccionarios (III)

```
#Crear un diccionario
persona1={"nombre":"Juan Sin Miedo Valiente","edad":17,"curso":"1º Bachillerato"}

#Imprimir diccionario
print(persona1)
```

	nombre	Edad	curso
persona1	Pepe Mellamo Nosecomomellamo	17	1º Bachillerato

```
#Imprimir valores
print(persona1["nombre"])
print(persona1["edad"])
print(persona1["curso"])

#Actualizar valores
persona1["nombre"] = "Juan Con Miedo Cobarde“
print(persona1)
```

	nombre	Edad	curso
persona1	Juan Con Miedo Cobarde	17	1º Bachillerato

https://ellibrodepython.com/diccionarios-en-python

Programación

Diccionarios (IV)

```
persona1={"nombre":"Pepe Mellamo Nosecomomellamo","edad":17,"curso":"1º Bachillerato"}
print(persona1)
```

	nombre	Edad	curso
persona1	Pepe Mellamo Nosecomomellamo	17	1º Bachillerato

```
#Imprimir las claves (key) del diccionario
print()
for clave in persona1:
        print(clave)

#Imprimir los valores (value)
print()
for clave in persona1:
        print(persona1[clave])
```

Los diccionarios se recorren utilizando un bucle for

https://ellibrodepython.com/diccionarios-en-python

Programación

Diccionarios (IV)

```
persona1={"nombre":"Pepe Mellamo
Nosecomomellamo","edad":17,"curso":"1º Bachillerato"}

#Imprimir la clave (key) y el valor (value) a la vez
for clave, valor in persona1.items():
        print(clave, valor)
```

Los diccionarios se recorren utilizando un bucle for

https://ellibrodepython.com/diccionarios-en-python

Programación

Ejercicios

Ejercicio:
Crea un diccionario para representar la información de un alumno en una asignatura. Las claves serán id, nombre y nota. Puedes asignar los valores que consideres convenientes.

Ejercicio:
Crea una lista de alumnos (los alumnos tendrán una estructura de diccionario semejante a la del ejercicio anterior). Puedes asignar los valores que consideres convenientes. La lista ha de tener al menos cinco alumnos.

Ejercicio:
Recorre la lista de alumnos anterior y muestra por pantalla si un alumno está aprobado o suspenso.

Funciones

Programación

Funciones(I)

```
#Se define la función mostrarMensaje
def mostrarMensaje ():
        print("Este es el mensaje a mostrar")

#Llamar a la función
mostrarMensaje()
```

Programación

Funciones(II)

```
#Se define la función mostrarMensaje
def mostrarMensaje (mensaje):
        print(mensaje)

#Llamar a la función
mostrarMensaje("Este es el mensaje a mostrar")
```

Programación

Funciones(III)

```
#Se define la función suma
def suma(a, b):
        print(f'La suma de {a} y {b} es {a+b}')

#Pedir datos
a = int(input('Introduzca el primer número: '))
b = int(input('Introduzca el segundo número: '))

#Procesar e imprimir el resultado
suma(a,b)
```

Programación

Funciones(IV)

```
#Se define la función suma
def suma(a, b):
        return (a + b)

#Pedir datos
a = int(input('Introduzca el primer número: '))
b = int(input('Introduzca el segundo número: '))

#Procesar e imprimir el resultado
print(f'La suma de {a} y {b} es {suma(a, b)}')
```

Programación

Funciones(V)

```
#Importar módulos
import random

#Se define la función lanzarDado
def lanzarDado():
        numero = random.randint(1,6)
        return numero

#Llamar a la función
print(lanzarDado())
```

Programación

Funciones(VI)

```
#Importar módulos
import random

#Se define la función numeroAleatorio
def numeroAleatorio(minimo,maximo):
        numero = random.randint(minimo,maximo)
        return numero

#Llamar a la función
print(numeroAleatorio(0,100))
```

Programación

Funciones(VII)

```
#Se define la función mostrarMensaje
def mostrarMensaje ():
        cadena= "Este es el mensaje a mostrar "
        return cadena

#Llamar a la función
cadena = mostrarMensaje()
print (cadena)
```

Programación

Funciones(VIII)

```
#Se define la función resta
def resta (a, b):
        return (a - b)

#Llamar a la función pasando como argumentos valores constantes
print (resta (5, 1))
a = 5
b = 1
#Llamar a la función pasando los argumentos en orden
print (resta (a, b))
#Llamar a la función pasando los argumentos identificados por el nombre
print (resta (a = 5, b = 1))
print (resta (b = 1, a = 5))
```

Programación

Ejercicio

Ejercicio:
Escribe una función que imprima el producto de dos números que se pasan como argumentos.
Ejercicio:
Escribe una función que devuelva el producto de dos números que se pasan como argumentos.
Ejercicio:
Escribe una función que devuelva el mayor de dos números que se pasan como argumentos.
Ejercicio:
Escribe una función que simule el lanzamiento de una moneda.

Excepciones

Programación

Excepciones(I)

```
print ("Introduzca un número entero: ")
try:
        numero = input()
        numero = int(numero)
        resultado = 3 * numero
        print("El resultado es igual a " + str(resultado))
except:
        print("El valor introducido no es un número entero.")
```

Programación

Excepciones(II)

```
try:
        b = int(input("Introduce un número entero: "))
        a=3/b
        print(a)
except ZeroDivisionError:
        print("No se puede dividir por cero")
except ValueError:
        print("El número introducido no es un entero")
except:
        print("Error desconocido")
```

Ejercicio

Ejercicio:
Escribe un programa que imprima la suma de dos números que se pasan como argumentos. Se debe controlar mediante excepciones que los valores introducidos por el usuario no sean valores numéricos.

Ejercicio:
Escribe un programa que imprima el cociente de dos números que se pasan como argumentos. Se debe controlar mediante excepciones que los valores introducidos por el usuario no sean valores numéricos y que el divisor sea un cero.

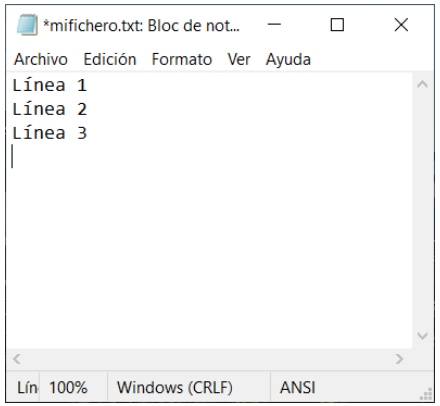
Ficheros de texto

Programación

Leer ficheros de texto

```
fichero=open("mifichero.txt", 'r')
lineas = fichero.readlines()
print(lineas)
fichero.close()
```

Crea una lista con todas las líneas del fichero.




```
fichero=open("mifichero.txt", 'r')
lineas = fichero.readlines()
for linea in lineas:
    print(linea)
fichero.close()
```

Bucle for para recorrer la lista con las líneas.

```
fichero=open("mifichero.txt", 'r')
print(fichero.read())
fichero.close()
```

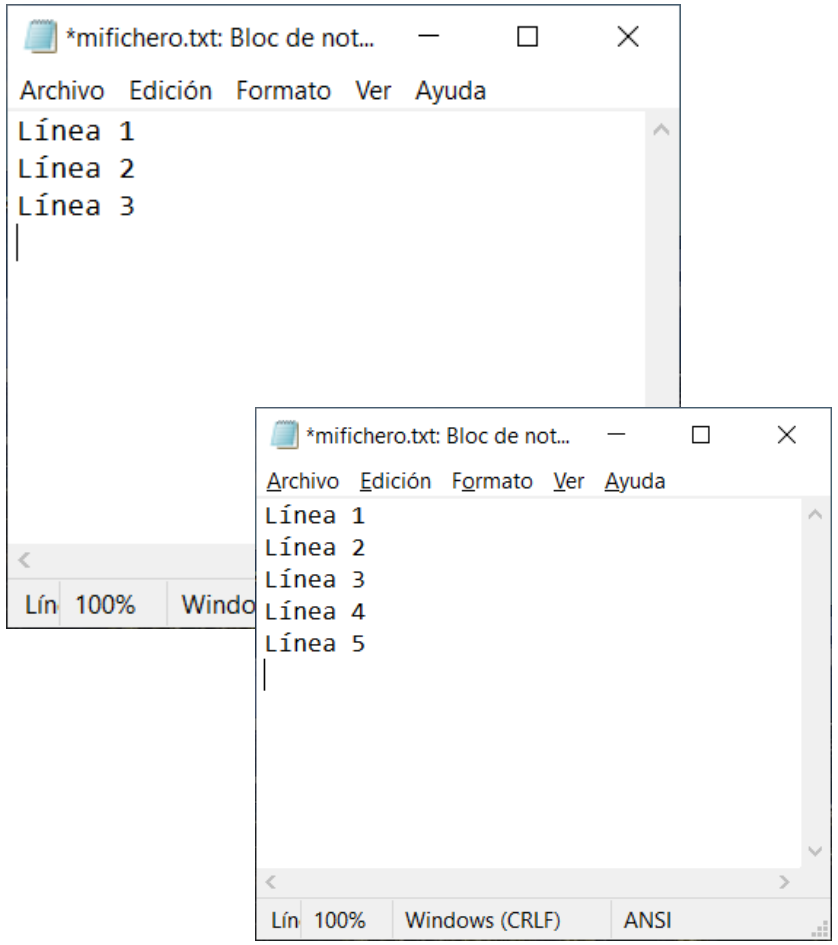
Lee todo el fichero de golpe.

Programación

Escribir ficheros de texto

```
fichero=open("mifichero.txt", 'w')
fichero.write("Línea 1\n")
fichero.write("Línea 2\n")
fichero.write("Línea 3\n")
fichero.close()
```

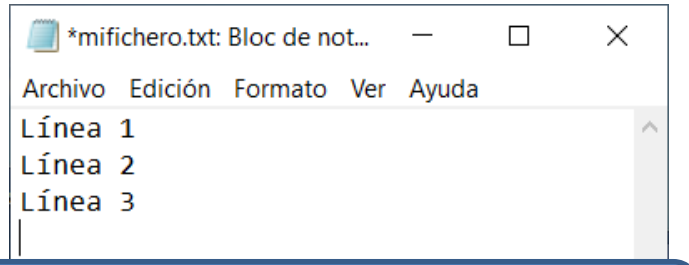
```
fichero=open("mifichero.txt", 'a')
fichero.write("Línea 4\n")
fichero.write("Línea 5\n")
fichero.close()
```



Programación

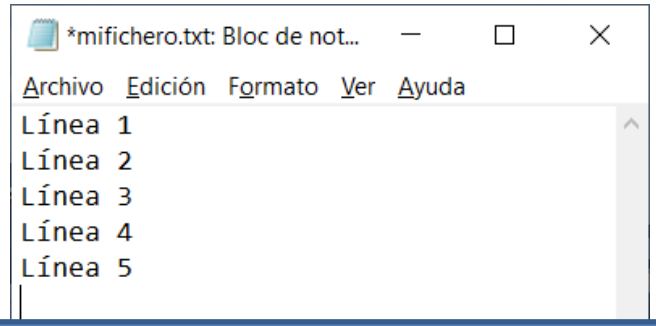
Escribir ficheros de texto

```
fichero=open("mifichero.txt", 'w')
fichero.write("Línea 1\n")
fichero.write("Línea 2\n")
fichero.write("Línea 3\n")
fichero.close()
```



Si el archivo no existe, lo crea. Si existe, lo sobreescribe.

```
fichero=open("mifichero.txt", 'a')
fichero.write("Línea 4\n")
fichero.write("Línea 5\n")
fichero.close()
```



Las líneas se añaden al final del archivo.

Fin

Programación

Introducción a la programación (Python II)