ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación Python

José Javier Galán Hernández

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación Python

8. LISTAS

8. LISTAS

Secuencia ordenada de valores a los que se puede acceder por índice o posición.

Declaración:

Separados por comas y entre corchetes:

In [1]:
```
lista = ["Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"]
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo']
```

Usando la función list(), ojo al doble paréntesis

In [3]:
```
lista = list(("Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"))
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo']
```

Declaración lista vacía

In [6]:
```
1 listaVacia = []
2 print(listaVacia)
3
```

```
[]
```

8. LISTAS

Accedemos a un elemento de la lista mediante su índice, la posición del primer elemento es 0.

In [10]:
```
lista = ["Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"]
print(lista[5])
```

```
Sabado
```

In [11]:
```
lista[5] = "Sábado"
print("He modificado ", lista[5], " de la lista ",lista)
```

```
He modificado  Sábado  de la lista  ['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']
```

8. LISTAS

Podemos **añadir** elementos a la lista:

In [12]:
```
lista.append("Dia extra")#Indicamos que añada un valor mas en la lista creada
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo', 'Dia extra']
```

In [13]:
```
lista[len(lista):] = ["Otro dia"]#Asignamos un valor en la posicion longitud, al empezar en 0, sera como añadir
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo', 'Dia extra', 'Otro dia']
```

In [14]:
```
lista.extend(["Tercer valor añadido"])
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo', 'Dia extra', 'Otro dia', 'Tercer valor añadido']
```

Eliminar un elemento:

In [17]:
```
del lista[9]
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo', 'Dia extra', 'Otro dia']
```

In [22]:
```
del lista[len(lista)-1]
print(lista)
```

```
['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo', 'Dia extra']
```

8. LISTAS

Errores fuera de rango:

```
In [15]: lista[15] = ["15"]#Error
         print(lista)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-15-c4158152d14a> in <module>
----> 1 lista[15] = ["15"]#Error
      2 print(lista)

IndexError: list assignment index out of range
```

```
In [16]: print(lista[15])
```

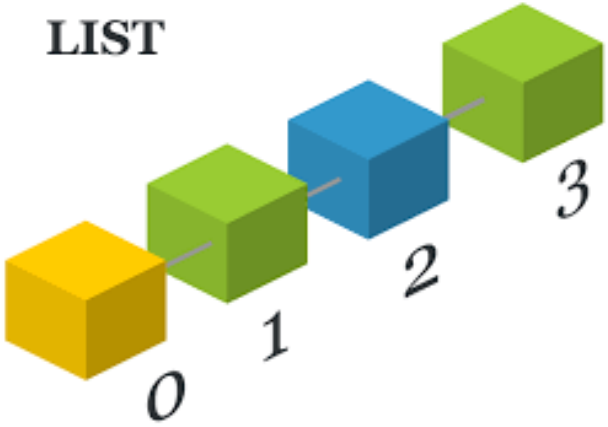
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-16-111230de3482> in <module>
----> 1 print(lista[15])

IndexError: list index out of range
```

8. LISTAS

Las listas pueden incluir elementos de **todo tipo**, inclusos **listas dentro de listas**.

```
In [24]: superLista = [["1","hola",1],[["as"],[2],[5],["df"],[324]]]#Lista con 2 listas de valores de distinto tipo
         print(superLista[1][2])
```

```
[5]
```



8. LISTAS

Se puede hacer referencia a la lista en **sentido inverso** haciendo uso de números negativos.
Se puede hacer referencia a una sección (slicing, rebanado) de la lista indicando:
la posición del valor inicial : numero anterior al segundo indicado

In [25]:
```
listaNumeros = [1,2,3,4,5,6,7,8,9,10]
print(listaNumeros)
```

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

In [26]:
```
print(listaNumeros[-1])
```

```
10
```

In [27]:
```
print(listaNumeros[-5])
```

```
6
```

In [28]:
```
print(listaNumeros[1:5])
```

```
[2, 3, 4, 5]
```

In [31]:
```
nuevaLista = (listaNumeros[1:5])
print(nuevaLista)
```

```
[2, 3, 4, 5]
```

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación Python

9. TUPLAS

9. TUPLAS

Parecidas a las listas, las tuplas no pueden modificarse.
No se escriben corchete [] (para tuplas se pueden usar parentesis ())
Se separan los elementos mediante comas ,.

In [34]:
```
tuplaNumeros = 1,2,3,4,5,6,7,8,9,10
print(tuplaNumeros)
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

In [35]:
```
tuplaNumeros[1]=23
print(tuplaNumeros)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-84bbf407ff29> in <module>
----> 1 tuplaNumeros[1]=23
      2 print(tuplaNumeros)

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

Tupla de 1 elemento

In [36]:
```
tuplaNumeros = (1,)#tupla de un elemento se indica con coma , al final
print(tuplaNumeros)
```

```
(1,)
```

9. TUPLAS

Tupla vacía.

In [37]:
```
tuplaVacia = ()
print(tuplaVacia)
```

```
()
```

Acceso a tupla, similar a listas pero no se permite modificar

In [38]:
```
tuplaNumeros = (1,2,3,4,5,6,7,8,9,10)
print(tuplaNumeros[2])
print(tuplaNumeros[-2])
print(tuplaNumeros[2:4])
```

```
3
9
(3, 4)
```

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación Python

10. DICCIONARIOS

10. DICCIONARIOS

Son conjuntos de valores emparejados.

Los valores de cada par se separan mediante dos puntos (:), de otros pares por coma (,) y el conjunto por llaves ({})

Para acceder a un valor se accede por índice pero el índice puede no ser un numero.

```
In [56]: diccionario = {"nombre":"pepe","altura":"190","peso":"90","dni":"09040284P","direccion":"Madrid"}
         print(diccionario)

         {'nombre': 'pepe', 'altura': '190', 'peso': '90', 'dni': '09040284P', 'direccion': 'Madrid'}

In [57]: print(diccionario["nombre"])

         pepe

In [58]: print(diccionario["pepe"])#por el segundo elemento no se busca

         ---------------------------------------------------------------------------
         KeyError                                  Traceback (most recent call last)
         <ipython-input-58-18c3e9c3cf42> in <module>
         ----> 1 print(diccionario["pepe"])

         KeyError: 'pepe'
```

10. DICCIONARIOS

Podemos añadir

In [62]:
```
diccionario.append("calle":"calle la goma")#Append no se usa con diccionario
print(lista)
```

```
  File "<ipython-input-62-055d1c49f786>", line 1
    diccionario.append("calle":"calle la goma")#Indicamos que añada un valor mas en la lista creada
                               ^
SyntaxError: invalid syntax
```

In [65]:
```
diccionario.update({"calle":"calle la goma"})#Si no existe lo crea
print(diccionario)
```

```
{'nombre': 'Juan', 'altura': '190', 'peso': '90', 'dni': '09040284P', 'direccion': 'Madrid', 'calle': 'calle la goma'}
```

In [66]:
```
diccionario.update({"calle":"calle la goma2"})#si existe modifica su valor
print(diccionario)
```

```
{'nombre': 'Juan', 'altura': '190', 'peso': '90', 'dni': '09040284P', 'direccion': 'Madrid', 'calle': 'calle la goma2'}
```

y eliminar parejas de valores.

In [67]:
```
del diccionario["calle"]
print(diccionario)
```

```
{'nombre': 'Juan', 'altura': '190', 'peso': '90', 'dni': '09040284P', 'direccion': 'Madrid'}
```

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación Python

11. FOR

11. FOR

Se utiliza para recorrer objetos iteradores como listas o diccionarios.

Sintaxis:

for elemento **in** listaElementos**:**
    sentencias

Ejemplo para diccionarios:

In [68]:
```
diccionario = {"nombre":"pepe","altura":"190","peso":"90","dni":"09040284P","direccion":"Madrid"}
for palabra in diccionario:
    print(palabra)
```

```
nombre
altura
peso
dni
direccion
```

In [1]:
```
diccionario = {"nombre":"pepe","altura":"190","peso":"90","dni":"09040284P","direccion":"Madrid"}
for palabra in diccionario:
    print(diccionario[palabra])
```

```
pepe
190
90
09040284P
Madrid
```

11. FOR

Ejemplo para listas:

In [2]:
```
diccionario =  ["uno","dos","tres","cuatro","cinco","seis","siete","ocho"]
for palabra in diccionario:
    print(palabra)
```

```
uno
dos
tres
cuatro
cinco
seis
siete
ocho
```

Ejemplo para cadenas:

In [3]:
```
frase = "En un lugar de la mancha..."
for letra in frase[:]:
    print(letra)
```

```
E
n

u
n

l
u
g
a
r

d
e

l
a

m
a
n
c
h
a
.
.
.
```

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID

Programación Python

12. FUNCIONES

12. FUNCIONES

Bloque de código que se puede usar varias veces desde distintos puntos del programa.

Sintaxis:

def nombreFuncion**():**
    sentencias

In [4]:
```
def Menu():
    print("1. Opcion1")
    print("2. Opcion2")
    print("3. Opcion3")
    print("4. Opcion4")
```

In [5]:
```
Menu()
```

```
1. Opcion1
2. Opcion2
3. Opcion3
4. Opcion4
```

Una función que no devuelve nada se llama subrutina (procedimiento)

12. FUNCIONES

Subrutina que recibe parámetros

```
In [7]: def Suma(numero1, numero2):
            print(numero1+numero2)
```

```
In [8]: Suma(2,234)
```

```
236
```

Función que recibe parámetros y devuelve un parámetro

```
In [9]: def Suma(numero1, numero2):
            return(numero1+numero2)
```

```
In [10]: resultado=Suma(234,342)
         print(resultado)
```

```
576
```

Se puede devolver mas de un parámetro separándolos por coma. Ejemplo return “hola”, “adiós”

12. FUNCIONES

```
def Menu():
    print("1.Opción 1")
    print("2.Opción 2")
    print("3.Opción 3")
    print("4.Salir")

def Valida(minimo, maximo):
    opcion=0
    while opcion<minimo or opcion>maximo:
        opcion=int(input("Escribe una opción entre 1 y 4: "))
    return opcion
```

```
Menu()
opcion=Valida(1,4)
```

En este ejemplo creamos 2 funciones, la primera muestra un menú de opciones y la segunda chequea/valida que la opción introducida sea valida.
Después hacemos uso de ambas.

…ahora…comencemos!

¡A programar!

Listas y Diccionarios

1. Crea un programa que pida al usuario un número de palabras a insertar (mínimo 7).Después, solicita al usuario que introduzca esas palabras y guárdalas en una lista. Muestra la lista completa por pantalla.
2. Pide al usuario un número entre 1 y 7 y una palabra. El número introducido indica la posición en la lista anterior cuyo valor será sustituido por la nueva palabra. Muestra la lista actualizada.
3. Pide al usuario la posición de un elemento en la lista anterior. Elimina de la lista la palabra que se encuentra en esa posición y muestra la lista resultante.
4. Crea un **diccionario** con 10 palabras en inglés y su traducción al español.
   Permite que el usuario realice **3 consultas** para traducir palabras del inglés al español.
5. Añade **4 palabras más** al diccionario anterior y muestra el diccionario completo actualizado.

For, funciones.

1. Solicita al usuario **10 palabras**, guárdalas en una **lista** y muéstralas **en orden inverso** (de la última a la primera).
2. Implementa un programa que simule una calculadora usando funciones:
   sumar(a, b)
   restar(a, b)
   multiplicar(a, b)
   dividir(a, b) (no obligatorio controla la división por cero).
   El programa debe pedir dos números y permitir elegir la operación para mostrar el resultado.
3. Solicita **10 números**, guárdalos en una **lista** y muéstralos recorriendo la lista con un **bucle for**.
4. Ordena la lista del ejercicio anterior de menor a mayor y muéstrala por pantalla.

Avanzado: Juego del ahorcado.

ntic master
revolucionamos la comunicación

UNIVERSIDAD COMPLUTENSE DE MADRID