PYTHON

Módulo 8 – Uso de módulos estándar de Python

USO DE MÓDULOS ESTÁNDAR DE PYTHON

Built-in functions

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - https://docs.python.org/3/library/functions.html
  - abs() → Devuelve el valor absoluto de un número
  - all() → Devuelve True si todos los elementos de un *iterable* (listas, tuplas, diccionarios y conjuntos) son True o está vacío.
  - any() → Devuelve True si un elemento de un *iterable* es True. Si está vacío devuelve False.
  - bin() → Convierte un número en base decimal a un string con la representación binaria.
  - bool() → Devuelve True o False como resultado del análisis del elemento pasado como parámetro.
  - callable()→Indica si un objeto posee método __call__. Esto implica que el objeto es *ejecutable*.

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - dict() → Convierte a diccionario
    - x = dict()
    - x = dict(a=1,b=2)
    - x = dict({'a':1,'b':2})
  - dir() → Sin parámetros, muestra la lista de nombres (variables, objetos, métodos,…) disponible en el ámbito actual. Pasando un objeto como parámetros, muestra la lista de nombres del mismo.
  - divmod()→Divide dos números y devuelve una tupla con el cociente entero y el resto.
    - >>> divmod(18,4)
    - (4, 2)

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - enumerate() → Convierte un iterable a enumerate
    - enumerate permite generar listados ordenados de elementos

```
>>> dias_semana = ("Lunes","Martes","Miércoles","Jueves","Viernes")
>>> for indice, valor in enumerate(dias_semana):
...     print(indice, valor)
...
0 Lunes
1 Martes
2 Miércoles
3 Jueves
4 Viernes
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - eval() → Permite evaluar una expresión. Devuelve el resultado de la evaluación.

```
>>> def saludar():
...     return "Hola"
...
>>> operacion = input("Introduce operacion:")
Introduce operacion:saludar()
>>> print(eval(operacion))
Hola
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - exec() → Permite ejecutar un bloque de código. NO devuelve el resultado de la evaluación.

```
>>> def saludar():
...     return "Hola"
...
>>> operacion = input("Introduce operacion:")
Introduce operacion:saludar()
>>> print(exec(operacion))
None
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- **Built-in functions:**
  - filter() → Permite aplicar una función de filtro a un iterable.

```
>>> dias = ("Lunes","Martes","Miércoles","Jueves","Viernes")

>>> def mas_de_5_caracteres(palabra):
...     resultado = len(palabra)>5
...     return resultado

>>> dias_largos = filter(mas_de_5_caracteres, dias)

>>> for d in dias_largos:
...     print(d)
...
Martes
Miércoles
Jueves
Viernes
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - float() → Convierte a número en coma flotante.
  - format() → Convierte un valor a una representación con formato.
  - getattr() → Obtiene el valor de un atributo de un objeto. El atributo no puede ser privado.
  - globals() → Obtiene todos los símbolos de ámbito global.
  - hasattr() →Permite saber si un objeto dispone de un determinado atributo. El atributo no puede ser privado.
  - help() → Proporciona información de una clase, función, variable (en función de su tipo) y objeto (en función de su clase)
  - hex()→Convierte un número a su representación hexadecimal como cadena de caracteres.

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - input() → Permite recoger una entrada de datos desde el teclado.
  - int()→Convierte a entero.
  - isinstance()→Permite determinar si un objeto es de una clase determinada.
    - `isinstance(o, Ordenador)`
  - iter()→Proporciona un iterador.
  - len()→Proporciona la longitud de un elemento (compatible).
  - list()→No es una función.
    - Ver: https://docs.python.org/3/library/stdtypes.html#list
  - locals() →Obtiene los símbolos de ámbito local

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - map() → Permite ejecutar una función sobre todos los elementos de un iterable sin utilizar bucles.
  - Ejemplo:

```
def suma_2(numero):
    return numero + 2

tupla = (1,2,3,4,5,6)
mapa = map(suma_2, tupla)
lista = list(mapa)
print(lista)
```

  - Resultado:
    - [3, 4, 5, 6, 7, 8]

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - max() → Proporciona el elemento mayor de un iterable o de una lista de elementos.
  - min() → Proporciona el elemento menor de un iterable o de una lista de elementos.
  - next()→En un iterador, proporciona el siguiente elemento.
  - oct()→Proporciona representado en un string el valor en base 8 (octal) del parámetro proporcionado
  - open() → Permite abrir un fichero.
  - ord()→Proporciona el valor numérico (ordinal) de la tabla ASCII del carácter proporcionado:
    - >>> print(ord("#"))
    - 35

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - pow() → Cálculo de potencias.
  - print() → Mostrar por la salida estándar.
  - repr()→Obtiene un string con la representación de un objeto.
  - round()→Redondea un número con decimales al entero más próximos.
  - set()→Proporciona un conjunto (set) a partir de un iterable.
    - >>> dias = ("Lunes","Martes","Miércoles","Jueves","Viernes")
    - >>> conjunto = set(dias)
    - >>> conjunto
    - {'Martes', 'Jueves', 'Lunes', 'Miércoles', 'Viernes'}

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - slice() → Proporciona un objeto *slice* que representa un conjunto de elementos de una secuencia (string, tupla, lista, rango o bytes).

```
>>> dias
('Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes')
>>> alternos = dias[slice(0,7,2)]
>>> alternos
('Lunes', 'Miércoles', 'Viernes')
>>> texto
'En un lugar de la mancha'
>>> letras = texto[slice(0,100,2)]
>>> letras
'E nlgrd amnh'
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - sorted() → Proporciona una lista ordenada a partir de un iterable

```
dias = ("Lunes","Martes","Miércoles","Jueves","Viernes")
ordenados = sorted(dias)
['Jueves', 'Lunes', 'Martes', 'Miércoles', 'Viernes']
```

  - Se puede proporcionar una función de ordenación:

```
def numero_letras_e(dia):
    numero = dia.count("e")
    return numero

dias = ("Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo")
ordenados = sorted(dias)
ordenados = sorted(dias, key=numero_letras_e)
print(ordenados)
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - str()→ Proporciona la versión string de un objeto.
  - sum()→ Suma un subconjunto de elementos de un iterable numérico. Admite un offset (un valor inicial)
  - tuple()→ No es una función. Ver:
    - https://docs.python.org/es/3/library/stdtypes.html#typesseq-tuple
  - type()→ Proporciona el tipo de una variable.

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in functions:
  - zip()→ Proporciona un iterador a partir de dos o más iterables.

```
dias = ("Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo")
planetas=("Mercurio","Venus","La Tierra","Marte","Jupiter","Saturno","Urano","Plutón")
numeros=[3,4,5,3,8,10,15,20]
todo_junto = zip(dias,planetas,numeros)
print(tuple(todo_junto))
```

```
(('Lunes', 'Mercurio', 3), ('Martes', 'Venus', 4), ('Miércoles', 'La Tierra', 5),
('Jueves', 'Marte', 3), ('Viernes', 'Jupiter', 8), ('Sábado', 'Saturno', 10), ('Domingo',
'Urano', 15))
```

USO DE MÓDULOS ESTÁNDAR DE PYTHON

Built-in constants

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in constants:
  - False
  - True
  - None
  - NotImplemented
  - Ellipsis
  - __debug__

USO DE MÓDULOS ESTÁNDAR DE PYTHON

- Built-in constants (módulo site):
  - Sólo en la consola interactiva.
  - quit(*code*)
  - exit(*code*)
  - copyright
  - credits
  - license