Python

Data Mining

Sesión 1

Ester Vidaña Vila

laSalle ENG
Universitat Ramon Llull

Python - operadores básicos

- *Suma: +*
- *Resta: -*
- *Multiplicación: **
- *División: /*
- *Cociente de la división: //*
- *Resto de la división: %*
- *Exponente: ***
- *Comentarios: #*
- *Los paréntesis se utilizan para dar prioridad de cálculo a los operandos deseados.*
- *Asignación de variables: =*
- *Strings: ‘ ‘ o “ “.*
- *Para printar por pantalla: función print().*

Python – tipos de datos

- ***Lists****:* utilizadas para guardar múltiples ítems en una única variable.
  - Se declaran entre [ ], cada elemento de la lista debe ir separado por una coma (,).
  - Se pueden anidar listas unas dentro de otras.
- ***Dictionaries***: se utilizan para guardar valores en pares clave:valor.
  - Un diccionario es una colección ordenada, modificable y que **NO** permite duplicados.
  - Se declaran entre { }, cada clave-valor se separa por dos puntos (:) y se separan los pares por comas (,): Ejemplo: {"clave1":"valor1", ”clave2”:”valor2”}.
- ***Tuples***: utilizadas para guardar múltiples ítems en una única variable.
  - Están ordenadas y su valor es **inalterable** (una vez declarada, no se puede modificar). Permiten tener elementos duplicados.
  - Se declaran entre ( ), y pueden accederse mediante índices.
- **Sets**: utilizadas para guardar múltiples ítems en una única variable.
  - No están ordenadas, y su valor es inalterable (una vez declarada, no se puede modificar). **NO** permiten tener elementos duplicados.
  - Se declaran mediante { } y **NO** pueden accederse a través de un índice.

Python – operadores de comparación

- *Mayor que: >*
- *Menor que: <*
- *Mayor o igual que: >=*
- *Menor o igual que: <=*
- *Igual: ==*
- *Diferente: !=*

Python – operadores booleanos

- *and*
- *or*

- *Trabajan con tipos datos booleanos tipo True or False.*

Python – flow control

- *For*
- *If*

- *Ejemplo:*
- Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
     print(x)
     if x == "banana":
          break
```

Python – flow control

- *While*
- *If*

- *Ejemplo:*
- Print a message once the condition is false:

```
i = 1
while i < 6:
     print(i)
     i += 1
else:
     print("i is no longer less than 6")
```

Python – methods in lists

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python – methods in dictionaries

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Python – methods in tuples

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python – methods in sets

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two or more sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with another set, or any other iterable

Python – methods in strings (i)

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable

Python – methods in strings (ii)

isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list

Python – methods in strings (iii)

rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

Python – list comprehension

- *Ofrece una sintaxis más corta para crear una nueva lista basada en valores de una lista existente.*
- *Ejemplo:*

```
fruits =["apple", "banana", "cherry", "kiwi", "mango"]
```

```
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)
```



```
newlist = [x for x in fruits if "a" in x]
```

Python – map

- **map()** es una función que devuelve un objeto *map* (un iterador) resultado de haber aplicado una función determinada sobre un conjunto de datos iterable (list, tuple, etc.).
- *Sintaxis:* map(fun, iter)
  Donde los parámetros son:
  - ***fun :*** *La función que se va a ejecutar para cada elemento iterable.*
  - ***iter :*** *El objeto iterable que va a mapearse (list, tuple, etc.)*

Python – filter

- **filter()** es una función que, dada una secuencia y una función, evalúa cada elemento de la secuencia para determinar si es cierta o no.
- *Sintaxis:* filter(fun, iter)
  Donde los parámetros son:
  - ***fun :*** *La función que se va a ejecutar para cada elemento iterable evaluando si éste es cierto o no.*
  - ***iter :*** *El objeto iterable que va a mapearse (list, tuple, etc.)*
  - Devuelve: un objeto iterable que ya ha sido filtrado por la función.