*18/09/2020*

**Clase 4**

1. Lenguajes compilados e interpretados
   1. Compilados 🡪 más rápidos, en binario
   2. Interpretados 🡪 más intuitivos 🡪 Python es uno de ello
2. Escribir en la terminal 🡪 jupyter notebook (o jupyter-notebook) para abrir el navegador web con Jupyter
3. localhost:8888 🡪 puerto local
   1. Para correr en servidores, se puede determinar un puerto concreto y abrir un túnel para poder trabajar en dicho servidor pero en remoto desde, por ejemplo, mi ordenador personal
   2. Cuando se gestionan grandes archivos (20GB, por ejemplo), se suele operar directamente en servidores, ya que los ordenadores personales no los gestionan bien por falta de recursos
      1. Como regla general, si el archivo que se está gestionando ocupa más que la memoria RAM del dispositivo, suele dar problemas en la manipulación
4. Tipos de cluster:
   1. On premises 🡪 tú te autogestionas todo
   2. On the Cloud 🡪 en la nube
5. %pylab inline 🡪 línea que configura que los plots salgan por defecto por pantalla
6. Cada tipo de objeto (string, integer, etc.) tiene sus propias “built-in functions”
7. isinstance(nombre\_variable, (int, dict)) 🡪 muy útil para control de flujo con “if” 🡪 si “a” es un entero, haz X, sino, haz Y…
8. dir(variable) 🡪 muestra todos los atributos posibles de la variable
   1. dir() a secas se utiliza para mostrar todas las variables utilizadas
      1. Creadas por el usuario 🡪 normales
      2. Creadas por Python (internas) 🡪 precedidas de “\_\_”
9. Importante saber cuándo se hacen asignaciones por valor y por referencia!! Por defecto, Python hace asignaciones por referencia !!

a=[1,2,3]

b=a

a.append(4)

a

>>> [1, 2, 3, 4]

b

>>> [1, 2, 3, 4]

1. Para hacer asignaciones por valor 🡪 c=a.copy()
2. id(a), id(b) 🡪 identifica la posición de memoria de cada variable, comprobando que en ese caso sería el mismo!

Paso por valor y referencia

Dependiendo del tipo de dato que enviemos a la función, podemos diferenciar dos comportamientos:

Paso por valor: Se crea una copia local de la variable dentro de la función.

Paso por referencia: Se maneja directamente la variable, los cambios realizados dentro de la función le afectarán también fuera.

Tradicionalmente:

**Los tipos simples se pasan por valor: Enteros, flotantes, cadenas, lógicos...**

**Los tipos compuestos se pasan por referencia: Listas, diccionarios, conjuntos...**

1. Control de flujo en bucles
   1. continue 🡪 hace que salte al siguiente valor omitiendo el resto de instrucciones de ese step del bucle
   2. break 🡪 hace que se interrumpa el bucle totalmente
2. while 🡪 cuidado con lanzar bucles infinitos
   1. Recomendable combinarlos con if + break
3. Funciones:
   1. Importante meter los argumentos con el tipo esperado, ya que de lo contrario Python intenta ejecutarlo sin identificar que no era lo que esperaba encontrar
   2. Valores locales VS globales:
      1. global nombre\_variable 🡪 Como norma general, tratar de huir de las variables globales 🡪 se mete dentro de la función, y hace que se modifique el valor de la variable fuera de la función
   3. Variables mudas 🡪 son las variables que se usan dentro de las funciones 🡪 importante usar nombres intuitivos para saber su contenido
   4. pass 🡪 se utiliza cuando quieres que una función no devuelva nada
4. Módulos:
   1. Lo normal es importar módulos que ya existen, pero si se quiere crear uno propio, se dejaría el .py en el directorio de lectura, y funcionaría igual
   2. Para que no se pisen, utilizamos la siguiente nomenclatura 🡪 import math as m
5. Importante distinguir métodos inplace de los que no lo son!
6. find vs index vs count
   1. find e index son parecidos 🡪 se diferencian en que index falla cuando no encuentra el string buscado
7. Importante usar el slicng para listas y cadenas