A saber. Tema 4. Python

Los contenidos están agrupados según los apartados de la web w3schools, pero algunos de ellos se han visto en clase sin que vengan en esa web.

1. DEFAULT. Python HOME. Python INICIO

- Conocer la organización de w3schools: editor online, ejercicios, etc.

2. PYTHON INTRO. Introducción a Python

- Quién crea Python, cuándo y origen del nombre.
- Conocer cinco razones para aprender Python: https://blog.educacionit.com/2020/01/29/5-razones-para-aprender-a-programar-en-python/
- Conocer, al menos, tres empresas importantes que usan Python: http://sistemasgeniales.com/lenguajes-de-programacion/10-sitios-web-famosos-creados-con-python/

3. PYTHON GET STARTED. Comenzar con Python

Uso de Python en una terminal

- Comprobar si Python está instalado y la versión
- Uso del intérprete o shell:
  - Arrancar y salir.
  - Uso de la ayuda.
  - Ejecución de instrucciones
- Ejecución de programas .py
- Saber usar las istrucciones básicas de terminal en Linux para poder ejecutar ficheros .py

Uso de Python en un IDE

- Significado de IDE
- Thonny
  - Nombre de las dos partes
  - Uso de ambas: ejecutar, ejecutar paso a paso, ...

4. PYTHON SYNTAX. Sintaxis de Python

- Indentación, significado y utilidad.
- Forma de crear una variable, no hay instrucción para ello.
- Comentarios de una línea

5. PYTHON COMMENTS. Comentarios de Python

- Tres usos de los comentarios
- Comentario de una línea: #, también al final de una línea de código
- Comentario de más de una línea: “”” “””

6. PYTHON VARIABLES. Variables de Python

- Forma de crear variable, no hay comando, no se declaran como ningún tipo
- Concepto de casting, y forma de convertir a str, int, float

- Forma de conocer el tipo de una variable print(type(x))
- Comillas simples o dobles son válidas para declarar variables string
- Significado de Case-Sensitive

7. PYTHON VARIABLES NAMES. Nombres de variable

- Reglas para los nombres de variable
- Forma de hacer nombres de variable multipalabra y nombre de cada una:
  - Camel Case: myFriend
  - Pascal Case: MyFriend
  - Snake Case: my_friend

8. PYTHON VARIABLES MULTIPLE. Asignar múltiples valores

- Asignar valores diferentes a varias variables en una línea yo, tu, el= 5, 4, "José"
- Asignar un mismo valor a varias variables: x = y = z = "Orange"

9. PYTHON VARIABLES OUTPUT. Salida de variables

- Comando print
- Signo + para combinar: caso de dos strings y caso de dos números

10. PYTHON VARIABLES GLOBAL. Variables globales

11. PYTHON VARIABLES EXERCISES

Hay que haber hecho todos los ejercicios de variables.

12. PYTHON DATA TYPES

- Saber lo que son int, str, float y list
- Forma de obtener el tipo de dato: con la función type(). Dará un mensaje del tipo <class 'int'>

13. PYTHON NUMBERS

- Saber lo ya mencionado, lo que es int y float

14. PYTHON CASTING

- Nada nuevo, aparte de lo del apartado 6.

15. PYTHON STRINGS

- Definición de string y de qué están rodeadas: comillas sencillas o dobles
- Forma de mostrar un string en la pantalla
- Forma de asignar un string a una variable
- String multilínea. Con triple comilla sencilla o doble

- Los string funcionan como arrays. Forma de acceder a un carácter. El primer carácter tiene índice 0.
- Iterar en un string, por ejemplo con for para imprimir todos los caracteres
- Longitud de un string
- Saber si una palabra o un carácter está en un string
- También buscar que no está.

16. PYTHON STRINGS SLICING

- Significado de slicing: coger parte de un string
- Forma de coger valores referenciados el principio, y al final

17. PYTHON STRINGS MODIFY

- Concepto de **método:** es un programa o función
- Forma de usar en general un método, cómo se escribe para que modifique el string
- Saber que no se modifica el string original
- Mostrar todo en mayúsculas con *print(a.upper())*
- Todo en minúsculas *print(a.lower())*
- Eliminación de los espacios en blanco al principio y al final *print(a.strip())*
- Reemplazar un carácter con *print(a.replace("H", "J"))*
- Creación de una lista teniendo en cuenta un separador especificado *print(texto1.split(" "))* en este caso el string es texto1, y el separador sería el espacio en blanco.

18. PYTHON STRINGS CONCATENATE

- Significado de concatenar
- Signo +

19. PYTHON STRINGS FORMAT

- El método format() cuando queremos concatenar strings con valores numéricos
- Saber usar el método format() como en el ejemplo:

```
>>> dias=5
>>> dinero=250
>>> texto="He trabajado {} días y he ganado {} euros"
>>> texto.format(dias, dinero)
'He trabajado 5 días y he ganado 250 euros'
```

- Para mostrar en pantalla podemos usar el separador de coma

```
>>>uno="El número de personas es "
>>>dos=50
>>> print(uno, dos)
El número de personas es  50
```

- Podemos resolver el problema tando para imprimir como para crear una nueva variable, aparte del método format, con casting, es decir, convirtiendo los números en strings

```
>>> tres=uno+str(dos)
>>> tres
'El número de personas es 50'
```

20. PYTHON STRINGS ESCAPE

- Concepto de carácter de escape
- Forma de ponerlos en Python: barra invertida seguida por el carácter que queremos poner.
- Aprender a usar dos de ellos: \" para entrecomillar parte de un string, y \' para poner el apóstrofo cuando escribimos en inglés.

21. PYTHON STRINGS METHODS

- Saber que hay una colección de métodos que están en este apartado y que se pueden consultar si se busca alguna operación con strings.
- Estos métodos no cambian el string, dan nuevos valores.

22. PYTHON STRINGS EXERCISES

- Saber hacer los 8 ejercicios de strings

23. PYTHON BOOLEANS

- Valores que puede tomar este tipo de datos
- Ejemplo de uso, en condicionales.
- La función bool() para evaluar valores y variables: saber los resultados esperados según el tipo y valor de una variable (entero, string, lista,...).

24. PYTHON OPERATORS

- Definición de operador: realizan operaciones sobre valores y variables.
- Grupos de operadores en Python, saber usar los siguientes:
  - Operadores aritméticos: +, -, /, *, **, %, //
  - Operadores de asignación: =, +=, -=

- Operadores de comparación: ==, !=, <, >, <=, >=
- Operadores lógicos: and, or, not
- Operadores de pertenencia o membresía: x in y, x not in y

25. PYTHON LISTS

- Definición: almacenamiento de varios valores en una sola variable.
- Cuatro tipos de datos en Python que almacenan colecciones de datos: lista, tupla, diccionario y set.
- Forma de indicar una lista, corchetes
- Los valores: indexados (en orden), cambiables y pueden ser duplicados
- Añadir elemento
- Determinar la longitud
- Cualquier tipo de datos para los elementos de una lista, incluso otras listas.

26. PYTHON LISTS ACCESS

- Acceso a ítems: igual que hicimos en los strings
- Comprobar is un ítem existe: igual que en strings, *"apple" in thislist*

27. PYTHON LISTS CHANGE

- Cambiar un ítem, es sobreescribirlo
- Cambiar más de un ítem
- Cambiar uno por varios y varios por uno
- Insertar ítems

28. PYTHON LISTS ADD

- Añadir un ítem al final de la lista con el método append()
- Insertar un elemento en el index indicado, con el método insert()

29. PYTHON LISTS REMOVE

- Eliminar un elemento
- Método pop() para eliminar un ítem mediante su index
- Aparte del método pop() hay una **keyword** que elimina también un elemento de una lista: del lista3[2] para borrar el elemento 2 de la lista lista3.
- El comando del también puede borrar una lista completa, hacerla desaparecer, no sólo vaciarla.

- Hay un método clear() que vacía la lista, aunque no la borra.

30. PYTHON LISTS LOOP

- Iterar sobre los elementos de una lista con for. *for x in myList:*
- Iterar teniendo en cuenta el índice de cada elemento: *for i in range(len(thislist)):*

31. PYTHON LISTS COMPREHENSION

-

32. PYTHON LISTS SORT

- Ordenar de forma ascendente, método sort()
- Orden descendente se usa como argumento la palabra clave reverse=True en el método sort()
- sort() modifica una lista, no da salida. Es decir, no se puede poner en un print por ejemplo. Hay que hacer antes un sort a una lista, y después, imprimirla con el nuevo orden que ahora tiene.
- Saber si se ordenan antes números, palabras con mayúsculas, etc. por defecto.
- Invertir el orden de los elementos, método reverse()

33. PYTHON LISTS COPY

- No se puede copiar una lista simplemente escribiendo list2 = list1, ya que: list2 sólo será una referencia a list1, y los cambios realizados en list1 automáticamente, también serán realizados en list2.
- Hay que hacerlo así: *lista2=lista1.copy()*

34. PYTHON LISTS JOIN

- Hay varias formas de poner en una lista los elementos de otra, es decir, unirlas. La más sencilla es usar el signo + de concatenación.

35. PYTHON LISTS METHODS

- Es una colección de los métodos vistos. Hay que conocer todos excepto extend().

36. PYTHON LISTS EXERCISES

- Hacer los 8 ejercicios.

53. PYTHON DICTIONARIES

- Concepto de diccionario. Forma de escribirlo. myDict={‘nombre’:’José’, ‘edad’:27}
- Pares clave:valor
- Coleción ordenada, modificable y que no admite duplicados. Ordenada a partir de Python 3.7
- Forma de saber el número de elementos, con len(nombreDelDiccionario)
- Tipo de datos para los elementos: cualquiera, incluidas listas y otros diccionarios

54. PYTHON DICTIONARIES ACCESS

- Se puede acceder con el nombre de clave: myDict[‘edad’]
- También con el método get(): myDict.get(‘edad’)
- Comprobar si una clave existe en un diccionario: clave in myDict

55. PYTHON DICTIONARIES CHANGE

- Se cambia el valor de un ítem accediendo a la clave: myDict[‘nombre’]=’Juan’

56. PYTHON DICTIONARIES ADD

- Igual que cuando se cambia un ítem, pero en este caso, al no existir, se crea nuevo, al final del diccionario.

57. PYTHON DICTIONARIES REMOVE

- Método pop() para eliminar un ítem a partir de su clave: myDict.pop(‘edad’)

58. PYTHON DICTIONARIES LOOP

- Usar un for para obtener las claves. Saber obtener también los valores con el mismo for.

59. PYTHON DICTIONARIES COPY

- Ocurre como en las listas. No basta con igualar una variable a otra, pues se crea una referencia, no una copia. Hay que utilizar el método copy() como en listas.

60. PYTHON DICTIONARIES NESTED

- Concepto de diccionario anidado.

61. PYTHON DICTIONARIES METHODS

- Una colección de métodos, que ahí están por si hay que consultar.

62. PYTHON DICTIONARIES EXERCISES

- Cinco ejercicios en total

63. PYTHON CONDITIONS

Ya se han usado en los ejercicios hechos hasta ahora. No obstante hay que saber:

- Condiciones lógicas matemáticas soportadas
- Forma de escribir un if. Dos puntos, indentación.
- Elif
- Else
- And
- Or
- If anidados

64. PYTHON WHILE LOOPS

- Forma de escribir un buble while. Dos puntos, indentación,
- Break
- Continue
- Else

65. PYTHON FOR LOOPS

También usado ya en los ejercicios. Hay que saber:

- Saber escribir el bucle en su versión más básica.
- Instrucción break
- Instrucción continue
- La función range()
    - Desde los valores por defecto al caso más complejo
        - for x in range(2, 30, 3):
            - print(x)
- Instrucción else