prog_datasci_2_python_entrega

October 3, 2019

1 Programación para *Data Science*

1.1 Unidad 2: Breve introducción a la programación en Python

2 Introducción

En este Notebook encontraréis dos conjuntos de ejercicios: un primer conjunto de **ejercicios para practicar** y un segundo conjunto de **actividades evaluables** como PEC de la asignatura.

En cuanto al conjunto de ejercicios para practicar, éstos no puntúan para la PEC, pero os recomendamos que los intentéis resolver como parte del proceso de aprendizaje. Encontraréis ejemplos de posibles soluciones a los ejercicios al propio notebook, pero es importante que intentéis resolverlos vosotros antes de consultar las soluciones. Las soluciones os permitirán validar vuestras respuestas, así como ver alternativas de resolución de las actividades. También os animamos a preguntar cualquier duda que surja sobre la resolución de los **ejercicios para practicar** en el foro del aula.

En relación a las actividades evaluables, veréis que cada una de ellas tiene asociada una puntuación que indica el peso que tiene la actividad sobre la nota de la PEC. Adicionalmente, hay un ejercicio opcional, que no tiene puntuación dentro de la PEC, pero que se valora al final del semestre de cara a conceder las matrículas de honor y redondear las notas finales. Podéis sacar la máxima nota de la PEC sin necesidad de hacer este ejercicio! El objetivo de este ejercicio es que sirva como pequeño reto para los estudiantes que quieran profundizar en el contenido de la asignatura.

Además, veréis que todas las actividades tienen una etiqueta que indica los recursos necesarios para llevarla a cabo. Hay tres posibles etiquetas:

- NM **Sólo materiales**: las herramientas necesarias para realizar la actividad se pueden encontrar en los materiales de la asignatura.
- EG **Consulta externa guiada**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, pero el enunciado contiene indicaciones de dónde o cómo encontrar la información adicional necesaria para resolver la actividad.
- EI **Consulta externa independente**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, y el enunciado puede no incluir la descripción de dónde o cómo encontrar esta información adicional. Será necesario que el estudiante busque esta información utilizando los recursos que se han explicado en la asignatura.

Es importante notar que estas etiquetas no indican el nivel de dificultad del ejercicio, sino únicamente la necesidad de consulta de documentación externa para su resolución. Además, recordad que las **etiquetas son informativas**, pero podréis consultar referencias externas en cualquier momento (aunque no se indique explícitamente) o puede ser que podáis hacer una actividad sin consultar ningún tipo de documentación. Por ejemplo, para resolver una actividad que sólo requiera los materiales de la asignatura, puedéis consultar referencias externas si queréis, ya sea tanto para ayudaros en la resolución como para ampliar el conocimiento!

En cuanto a la consulta de documentación externa en la resolución de los ejercicios, recordad **citar siempre la bibliografía utilizada** para resolver cada actividad.

2.1 Ejercicios para practicar

Los siguientes 3 ejercicios no puntúan para la PEC, pero os recomendamos que los intentéis resolver antes de pasar a los ejercicios propios de la PAC. También encontraréis las soluciones a estos ejercicios al final del Notebook.

2.1.1 Ejercicio 1

Escribir un programa que defina una lista los valores con los nombres de los días de la semana. Haz que muestre el primero y el penúltimo día de la semana. NM

[1]:
```
# Respuesta
```

2.1.2 Ejercicio 2

Escribir una expresión en Python que muestre la siguiente salida por pantalla: Hello

Python (Tened en cuenta que hay un salto de línea entre las dos palabras y que la palabra "Python" está tabulada hacia la derecha) EG

Nota: En el notebook de teoría hemos visto cómo crear cadenas de caracteres. Para hacer la actividad, tendréis que investigar cómo incluir saltos de línea y tabulaciones en estas cadenas!

[ ]:
```
# Respuesta
```

2.1.3 Ejercicio 3

A partir de las siguientes listas

```
l_1 = [Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo]
l_2 = [1, 2, 3, 4, 5, 6, 7, 8]
```

Escribe una expresión que devuelva:

1. El último elemento de la primera lista
2. Una nueva lista con el primer elemento de cada una de las listas
3. Una nueva lista con los tres primeros elementos de la segunda lista
4. Una cadena de caracteres con el valor "Miércoles 8"

NM

[2]:
```
l_1 = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']
l_2 = [1, 2, 3, 4, 5, 6, 7, 8]

# Respuesta
```

2.2 Ejercicios y preguntas teóricas para la PAC

A continuación, los **ejercicios y preguntas teóricas que deberéis completar en esta PAC** y que forman parte de la evaluación de esta unidad.

En el siguiente enlace podréis encontrar la documentación oficial de *string functions* que puede ser de ayuda para completar los ejercicios: https://docs.python.org/2/library/string.html.

2.2.1 Pregunta 1

£Cuál es el valor final de a, b y c (atención a los tipos de número decimal y entero)? **(1 punto)** NM

a = 5 / 11
b = 6 / 2.
c, a = b // 2, b + 1

Respuesta

Escribid vuestra respuesta aquí.

2.2.2 Ejercicio 1

Proporciona una única expresión que asigne el valor `7.5` a la variable `a_number`, el valor `[42]` a la variable `a_list` y el valor `"hello world"` a la variable `a_string`. Es necesario que proporciones una única expresión que realice las tres asignaciones. **(1 punto)** NM

[1]:
```
# Respuesta
```

2.2.3 Ejercicio 2

Qué expresión en Python necesitamos para conseguir el *string* "Learning Python" utilizando sólo las variables `str1` `ystr2` definidas? **(1 punto)** NM

[33]:
```
str1 = "I love Python, it's great!"
str2 = "Learning"
```

[3]:
```
# Respuesta
```

2.2.4 Ejercicio 3

Escribir un programa que asigne tres valores reales aleatorios a tres variables con nombre `a`,`b` y `c`. Utiliza las variables definidas anteriormente para evaluar la siguiente expresión matemática:

$\frac{-b+\sqrt{b^2-4ac}}{2a}$

(1 punto) EG

Nota: En el notebook de teoría hemos visto como sumar, restar, multiplicar y dividir números reales. Para hacer la actividad, tendréis que investigar cómo calcular potencias y raíces cuadradas en Python! Necesitaréis utilizar la librería `Random` para la generación de números aleatorios.

[4]:
```
# Respuesta
```

2.2.5 Ejercicio 4

Escribir un programa que calcule el diámetro de una esfera de radio 5. **(1 punto)** NM

[5]:
```
# Respuesta
```

2.2.6 Ejercicio 5

Escribir un programa que defina una lista con los 10 primeros números primos. Mostrad por pantalla los tres últimos números de la lista. **(1 punto)** NM

[6]:
```
# Respuesta
```

2.2.7 Ejercicio 6

Escribir un programa en el que se definan los precios de una frutería utilizando un *diccionario*. Deberéis incluir, como mínimo, 5 tipos de fruta diferentes. Por cada fruta, deberá indicarse el precio por kilo.

Por ejemplo: Fresas de Huelva con precio 5.9, Fresas del Maresme con precio 10, Manzanas con precio 3.5, etc.

Muestra el precio por kilo de cualquiera de las frutas que has incluido en el diccionario, utilizando cualquiera de los métodos de acceso a elementos de un diccionario. **(1 punto)** NM

[7]:
```
# Respuesta
```

2.2.8 Ejercicio 7

Escribir un código que muestre por pantalla el precio de todas las frutas incluidas en el diccionario creado en el ejercicio anterior (accediendo al diccionario que acabas de definir). **(1 punto)** NM

El formato de la cadena de caracteres que muestra el precio debe ser el siguiente:

`Las frutas tienen un precio de: {precios} euros`

Siguiendo con el ejemplo del ejercicio anterior, habría que mostrar por pantalla:

`Las frutas tienen un precio de: [5.9, 10, 3.5] euros`

Nota: recordad que los diccionarios no tienen orden. Por lo tanto, otras ordenaciones de los precios también serían soluciones válidas, por ejemplo:

`Las frutas tienen un precio de: [10, 5.9, 3.5] euros`

[8]:
```
# Respuesta
```

2.2.9 Ejercicio 8

Ordenad la siguiente lista de cadenas de caracteres: **(1 punto)** EG

1. Invirtiendo el orden original

2. En orden alfabético inverso

Nota: Puedes consultar la documentación oficial de la función `sorted` para ver qué parámetros puedes utilizar para resolver la segunda parte de la actividad.

[9]:
```
st_chars = ["Benjamin Sisko", "Kira Nerys", "Odo", "Quark", "Jadzia Dax"]
```

[10]:
```
# Respuesta 8.1
```

[11]:
```
# Respuesta 8.2
```

2.2.10 Ejercicio 9

A partir de la siguiente lista,

`a_list = [42, 7.5," Answer to the Ultimate Question "," Dave ", 7.5]`

proporcionad expresiones que devuelvan:

1. El número de veces que aparece el elemento 7.5 en la lista.
2. La posición de la primera aparición del valor 7.5
3. La misma lista sin el último elemento

Nota: En el notebook de teoría hemos visto qué son las listas y algunas operaciones sobre ellas. Para hacer la actividad, necesitaréis investigar algunas operaciones adicionales que podemos realizar sobre listas. Para ello podéis consultar la documentación oficial de Python sobre listas (intro y more on lists).

(1 punto) EG

[12]:
```
a_list = [42, 7.5, "Answer to the Ultimate Question", "Dave", 7.5]

# Respuesta
```

2.2.11 Ejercicio Opcional

Escribir un programa que genere una lista de números aleatorios y calcule el producto de todos los números de la lista. Los números de la lista deberán cumplir la siguiente condición: deben ser números impares entre 0 y 100. Además, el programa tendrá que preguntar al usuario cuántos elementos debe tener la lista (el usuario introducirá el número de elementos usando el teclado) y mostrar un mensaje de error si el usuario pide generar una lista de más de 50 elementos. Haz que el programa muestre la lista generada y el producto de sus elementos. EI

[13]:
```
# Respuesta
```

2.3 Soluciones de Ejercicios para practicar

2.3.1 Ejercicio 1

La forma más fácil de definir un array en Python es utilizando [ ]:

[8]:
```
dias = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado',␣
↪'Domingo']
```

Para acceder a los elementos de un array utilizaremos el mismo operador [ ], pero especificaremos la posición o índice del elemento al que queremos acceder. Debemos tener en cuenta que en Python, el primer elemento de una lista tiene índice 0.

[10]:
```
# Primer día de la setmana
print (dias[0])

# Penúltimo día de la setmana
print (dias[-2])
```

```
Lunes
Sábado
```

Python nos permite utilizar un índice negativo que contaría desde el final del array hacia el principio.

2.3.2 Ejercicio 2

Hay diferentes maneras de conseguir el mismo resultado y después veremos algunas de ellas.

La primera opción, y quizás la más simple, sería utilizar las triples comillas dobles (""" """). En Python, las triples comillas dobles se interpretan de forma literal y cualquier carácter que incluya nuestra cadena será renderizado tal y como aparece en la cadena. Esto nos puede resultar útil si por ejemplo nuestra cadena incluye comillas simples y comillas dobles. Vemos el código:

[11]:
```
print("""Hello

    Python""")
```

```
Hello

    Python
```

La segunda opción incluye el uso de caracteres especiales como “ n” y “ t”. El primero genera un salto de línea en la salida por pantalla, mientras que el segundo genera un tabulado (TAB). Por lo tanto, nuestro código quedaría de la siguiente manera:

[13]:
```
print("Hello\n\n\tPython")
```

```
Hello

        Python
```

En este caso, podemos ver que la indentación de la palabra “Python” es superior que en el caso anterior. Esto es debido a que, por defecto, el carácter ĝenera 8 espacios en blanco para "simular" la tabulación.

[14]:
```
print("Hello\n\n    Python")
```

```
Hello

    Python
```

2.3.3 Ejercicio 3

[15]:
```
l_1 = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']
l_2 = [1, 2, 3, 4, 5, 6, 7, 8]
```

[16]:
```
# 1. El último elemento de la primera lista
print(l_1[-1])
```

```
Domingo
```

Utilizamos un índice negativo para indicar que queremos la última posición.

[17]:
```
# 2. Una nueva lista con el primer elemento de cada una de las listas
print([l_1[0], l_2[0]])
```

```
['Lunes', 1]
```

Creamos una nueva lista utilizando `[` y `]`. Dentro de la lista ponemos los elementos indicados, accediendo a partir de su índice.

[18]:
```
# 3. Una nueva lista con los tres primeros elementos de la segunda lista
print(l_2[0:3])

# o equivalentmente:
print(l_2[:3])
```

```
[1, 2, 3]
[1, 2, 3]
```

Indicamos que queremos los tres primeros elementos, especificando el rango `0: 3`. El cero se puede omitir, obteniendo la expresión `[: 3]` que también es válida para recuperar los mismos elementos.

[19]:
```
# 4. Una cadena de caracteres con el valor "Miércoles 8"
print(l_1[2] + " " + str(l_2[-1]))
```

```
Miércoles 8
```

Creamos la cadena concatenando el tercer elemento de la primera lista, un espacio, y el último elemento de la segunda lista.