prog_datasci_3_apython_entrega_es

March 18, 2020

1 Programación para *Data Science*

1.1 Unidad 3: Estructuras de control y funciones en Python

En este Notebook encontraréis un conjunto de **actividades evaluables** como PEC de la asignatura.

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.

1.2 Ejercicios y preguntas teóricas para la PEC

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

1.2.1 Ejercicio 1

Un número primo es aquél que solo es divisible por él mismo y por 1.

a) Escribe un código que compruebe si un número `x = 15` es solo divisible por 1 o por el mismo. Escribe este código usando un iterador (un `for` o un `while`) que barra todos los valores desde `2` a `x-1`. Crea una variable `divisible` que tenga por defecto valor `False` y asigne el valor `True` si a lo largo de la iteración encuentra un número natural divisible. Puedes usar el operador modulo `a % b` para saber si un numero `b` es divisible por `a`.

(1.5 puntos) NM

[ ]:
```
# Respuesta
```

b) Convierte tu código anterior en una función que compruebe si el número del argumento es primo o no, devolviendo True is es primo y False si no es primo. Comprueba tu función con los valores 492366587, 492366585, 48947 y 2,

(0.5 puntos) NM

[ ]:
```
# Respuesta
```

c) En el cálculo de la función anterior, una vez se ha encontrado un número que es divisible dentro del rango ya no tiene sentido comprobar el resto de números del rango. Por ejemplo si 10 ya es divisble entre 2, ya no hace falta probar de 3 en adelante pues ya sabemos que el número no es primo.

Modifica la función anterior de la siguiente forma: - Una vez se encuentra el divisor, la iteración se interrumpe para no probar el resto de enteros. - La función devuelve - **Si es primo**: True - **Si no es primo**, el primer divisor mayor que 1.

Puedes hacer uso del comando *break* dentro de un bucle para interrumpir este, puedes consultar más información sobre break en la documentación de python aquí.

Comprueba tu función con los valores 492366585, 492366587, 48947 y 2,

(1.5 puntos) NM

[ ]:
```
# Respuesta
```

1.2.2 Ejercicio 2

La Covid-19 es una enfermedad producida por la infección del virus SARS-CoV-2. La infección es transmisible de persona a persona y su contagiosidad depende de la cantidad del virus en las vías respiratorias. Si cada persona contagiada transmite la enfermedad a β contactos en promedio por periodo de tiempo t, es posible estimar la evolución del contagio con un modelo matemático sencillo.

Para $t = 1$día, las transmisiones en España se han estimado a partir de su histórico de las semanas de Febrero y Marzo del 2020 una $\beta = 0.35$ transmissiones por día por infectado.

Durante un periodo de tiempo (por ejempo un día d) la tasa de nuevos contagios se puede estimar como una proporción al número de contagiados del periodo anterior N:

$$\Delta N = N_1 - N = \beta \cdot N$$

(1)

Por tanto, podemos proyectar el número futuro de afectados como

$$N_1 = N + \beta \cdot N = (1 + \beta) \cdot N$$

(2)

En dos días:

$$N_2 = (1 + \beta) \cdot N_1 = (1 + \beta)^2 \cdot N$$

(3)

Y en general en D días tendremos

$$N_D = (1 + \beta)^D \cdot N$$

(4)

Asumiendo este sencillo modelo:

a) Implementa una función de dos parámetros (N: población infectada inicial, D: número de días), que devuelva el cálculo de afectados para D días siguiendo la ecuación (4). Suponiendo una población afectada de 4250 (población afectada en españa a día 13 de Marzo de 2020), usa la función para calcular la población estimada en 1, 2, 7 y 30 días.

(1.5 puntos) NM

[ ]:
```
# Respuesta
```

b) Sabiendo que los Servicios de Medicina Intensiva (SMI) disponen de 3363 camas para enfermos graves, y suponiendo que un 10% de los afectados por el covid-19 requerirán de SMI y una supervivencia del 2,5% (Exitus), escribe un código que calcule:

- El día en curso (Día)
- El total de afectados por el virus para cada día d (Afectados)
- El total de ingresados en SMI por el virus para cada día d (Críticos)
- El total de Exitus por el virus para cada día d (Exitus)
- Si los servicios de SMI no pueden aceptar los ingresados para cada día d (Estado: indicando Saturación/No Saturación)

Imprime en pantalla la información de cada día durante una simulación de tres semanas, suponiendo que no hay recuperaciones, con una población afectada inicial 4250 y una $\beta = 0.35$ constante.

(1 punto) NM

[ ]:
```
# Respuesta
```

c) Convierte el código anterior en una función que genere un archivo de texto con nombre `output.txt`, siguiendo este formato:

```
Dia, Afectados, Críticos, Exitus, Estado
0, 4250, 425,  106, No Saturación
1, 5737, 573,  143, No Saturación
2, 7745, 774,  193, No Saturación
...
```

Con los parámetros de entrada N, D, β, camas SMI.
(1 punto) NM

[ ]:
```
# Respuesta
```

1.2.3 Ejercicio 3

Dado el siguiente diccionario:

[ ]:
```
d = {"Alex":344334443, "Eva":5533443, "Cristina":443355, "Jonas":33223324}
```

Escribid una función que pregunte al usuario que introduzca el nombre de una persona y muestre por pantalla el nombre de la persona y su teléfono.

Tened en cuenta que:

- La función debe controlar que el valor introducido por el usuario es un nombre que existe en el diccionario. En caso contrario, mostrará un mensaje de error ("El nombre introducido no corresponde a ninguna persona") y devolverá el valor False.
- Debéis tener en cuenta que el nombre de las personas que nos pasan por parámetro puede ser en minúsculas, mayúsculas o una combinación de ambas, y que debemos encontrar el número de teléfono aunque la capitalización de la cadena entrada por el usuario no sea exactamente la misma que hemos almacenado en el diccionario.
- Suponed que no hay acentos en los nombres.

Nota 1: Para realizar la actividad, tendréis que capturar un texto que entrará el usuario. Consultad la documentación oficial de la función input para ver cómo hacerlo.

Nota 2: También tendréis que pensar cómo tratar el hecho de que el usuario pueda utilizar mayúsculas y minúsculas en la escritura del nombre en el diccionario. ąOs animamos a usar un buscador para intentar encontrar alguna alternativa para resolver este subproblema! ąRecordad citar las referencias que hayáis usado para resolverlo!

(1.5 puntos) EG

[ ]:
```
d = {"Alex":344334443, "Eva":5533443, "Cristina":443355, "Jonas":33223324}

# Respuesta
```

Referencias consultadas:
Incluir aquí las referencias

1.2.4 Ejercicio 4

Python dispone de un **idio** muy útil conocido como `list comprehension`. Utilizando este **idiom**, proporcionad una expresión que devuelva las listas siguientes.

Nota: Para realizar esta actividad necesitaréis investigar qué son las `list comprehension` y qué sintaxis utilizan. Para ello, se recomienda en primer lugar que utilicéis un buscador para encontrar información genérica sobre esta construcción. Después, os recomendamos que consultéis stackoverflow para ver algunos ejemplos de problemas que se pueden resolver con esta construcción.

stackoverflow es un sitio de preguntas-y-respuestas muy popular entre programadores. Veréis que para la gran mayoría de las dudas que tengáis, habrá alguien que ya les habrá tenido (y consultado) anteriormente! Así pues, más allá de preguntar vosotros mismos las dudas allí (nosotros ya tenemos el foro del aula para ello!), consultar esta web os permitirá ver qué soluciones proponen otros programadores a estas dudas. A menudo habrá más de una solución a un mismo problema, y podréis valorar cuál es la más adecuada para vuestro problema.

Para ver ejemplos de problemas que son adecuados para resolver con **list comprehensions**, os recomendamos leer las siguientes páginas: * https://stackoverflow.com/questions/12555443/squaring-all-elements-in-a-list * https://stackoverflow.com/questions/18551458/how-to-frame-two-for-loops-in-list-comprehension-python * https://stackoverflow.com/questions/24442091/list-comprehension-with-condition * https://stackoverflow.com/questions/41676212/i-want-to-return-only-the-odd-numbers-in-a-list * https://stackoverflow.com/questions/4260280/if-else-in-a-list-comprehension

(1 punto) EG

a) Una lista con los valores $4x^2$ donde x es cada uno de los números de la lista `list_1`:

[ ]:
```
list_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Respuesta
```

b) Una lista con los valores $x/(x+1)$ donde x es cada uno de los números de la lista `list_1`:

[ ]:
```
list_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Respuesta
```

c) Una lista con los valores $4x^2/(4x^2-1)$ donde x es cada uno de los números de la lista `list_1`:

[ ]:
```
list_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Respuesta
```

1.2.5 Ejercicio 5

Las funciones `lambda` son formas de expresar y definir funciones pequeñas sin necesidad de usar el constructor `def funcion():`.

Lee sobre las funciones lambda, por ejemplo aquí o aquí

Escribe una función f con argumento n, $f(n)$, que **devuelva una función** lambda, que esta a su vez devuelva n copias de una cadena de caracteres en su argumento:

(0.5 puntos) EI

[ ]:
```
# Respuesta

def f(n):
    # Definir de la función usando una método lambda
    return()
r = f(5)
r("hola")  # Donde deberíamos ver 5 copias del literal "Hola "
```

1.2.6 Ejercicio Opcional

Existe una expresión atribuida a John Wallis (1616) para la estimación del valor de π, consistente en:

$$\frac{\pi}{2} = \prod_{n=1}^{N} \left(\frac{4n^2}{4n^2 - 1}\right)$$

si N es suficientemente grande $N \to \infty$.

Escribe una función que, dado una aproximación N, calcule una estimación de π siguiendo la fórmula de Wallis.

Consideraciones:

- Investigad las funciones map, reduce
- También podéis usar una list comprehension
- Las funciones lambda os pueden ser útiles

EI

[ ]:
```
# Respuesta
```