prog_datasci_3_python_cast_entrega

March 11, 2022

1 Fundamentos de Programación

1.1 PAC 3 - Enunciado

En este Notebook se encontraréis el 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 final 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. Podréis sacar la máxima nota de la PAC 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.

Veréis que todas las actividades de la PEC 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 independiente**: 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 para la PEC

A continuación encontraréis los ejercicios que se deben completar en esta PEC y que forman parte de la evaluación de esta unidad.

1.2.1 Ejercicio 1

Considera la lista siguiente **(1.75 puntos)**:

[ ]:
```
lista=[25, 'Hello', False, 'Python', -4]
```

a) Tienes un fragmento de código. Modifícalo para que muestre los elementos que se indican a continuación. Razona tu respuesta NM **(0.5 puntos)**

Nota: Tienes que realizar los mínimos cambios posibles.

a1) Debe mostrar los siguientes elementos: Hello, False, Python, -4.

a2) Debe mostrar los siguientes elementos: 25, Hello, False.

[ ]:
```
# Código a modificar
for n in lista:
    if n == 'Hello':
        continue
    print(n)
```

[ ]:
```
# Respuesta
```

b)Escribe un código que, dada la lista que estamos trabajando, 1) detecte los elementos de tipo string de la lista, 2) los ordene al revés (es decir, Hello sería olleH y Python, nohtyP) y 3) los muestre por pantalla. NM **(0.25 puntos)**

[ ]:
```
# Respuesta
```

c) Escribe un código que, dado un rango de números de 1 a 25, guarde los números primos en una lista. Muestra después por pantalla los elementos de la lista.EG **(1 punto)**

Nota: Puedes consultar cómo detectar si un número es primero en este post de stack overflow.

[ ]:
```
# Respuesta
```

1.2.2 Ejercicio 2

a) Pon el código del ejercicio 1c) en forma de función. Debe cumplir los siguientes puntos:

- Como entrada (*input*), especifica el número máximo que se desea recorrer (por ejemplo, en el ejercicio anterior, el número máximo era 25).
- Como salida (*output*), debe devolver la lista de números primos.

NM **(0.5 puntos)**

```
[ ]: # Respuesta
```

b) Utiliza los casos de prueba detallados en la tabla siguiente para comprobar que tu código funciona correctamente. NM **(0.25 puntos)**

máximo	lista resultado
9	[2, 3, 5, 7]
15	[2, 3, 5, 7, 11, 13]
50	[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

1.2.3 Ejercicio 3

Utilizando el idiom de list comprehension, proporciona una expresión que devuelva:

a) Una lista de los números impares entre -10 y 10 (ambos incluidos). **(0.5 puntos)**

b) Una lista en valor absoluto de los números múltiples de 3 entre -30 y 5 (ambos incluidos).**(0.5 puntos)**

c) Una lista de todas las letras de la siguiente frase **A bird in hand is worth two in the bush**. La lista no debe tener elementos vacíos.**(0.5 puntos)**

d) Haz lo mismo que en el apartado c) pero con una función *lambda*. **Opcional**

(1.5 puntos) EG

Nota: Para realizar esta actividad necesitarás investigar qué son las *list comprehension* y qué sintaxis utilizan. Para ello, se recomienda, en primer lugar, que utilices un buscador para encontrar información genérica sobre esta construcción. Después, recomendamos que consultes stackoverflow (un sitio de preguntas-y-respuestas muy popular entre programadores) para ver algunos ejemplos de problemas que se pueden resolver con esta construcción:

https://stackoverflow.com/questions/12555443/squaring-all-elements-ina-list

https://stackoverflow.com/questions/57166908/using-list-comprehension-i-want-to-printodd-even-with-string-indicating-even/57167016

```
[ ]: # Respuesta
```

1.2.4 Ejercicio 4

a) Nos interesa mucho la geografía y queremos realizar un programa que, dado un país, nos dé información concreta. Los puntos que debemos considerar son los siguientes **(2.5 puntos)**:

- El código debe estar en forma de función, con el nombre del país como entrada (*input*). NM **(0.5 puntos)**

- Se debe crear un diccionario que, para cada país, contenga el nombre de su capital, la moneda, los habitantes por km^2 y el continente. Por ejemplo, por Francia, la información que tendríamos sería: París, Euro, 121 y Europa. NM **(0.5 puntos)**
- En el diccionario se deben incluir los países siguientes: Francia, España, Italia, Japón, India, China, USA, Canadá, México, Argentina, Perú, Ghana, Egipto, Nigeria, Australia y Nueva Zelanda.NM **(0.5 puntos)**
- Como salida (*output*), el programa debe mostrar un mensaje que contenga la información del país que solicitamos. Por ejemplo, en el caso de Francia:

La capital de Francia es París. La moneda utilizada es Euro. El país tiene 121 km^2 y se ubica en el continente Europa. NM **(0.5 puntos)** - Si se solicita información de un país que no está en el diccionario, el programa debe mostrar un mensaje avisando.NM **(0.5 puntos)**

[ ]:
```
# Respuesta
```

b) Utiliza los casos de prueba detallados en la tabla siguiente para comprobar que la función que has creado funciona correctamente. NM **(0.5 puntos)**

País	Mensaje
Italia	La capital de Italia es Roma. La moneda utilizada es Euro. El país tiene 196 habitantes por km2 y se sitúa en el continente Europa.
Japon	La capital de Japon es Tokio. La moneda utilizada es Yen. El país tiene 332 habitantes por km2 y se sitúa en el continente Asia.
Canada	La capital de Canada es Ottawa. La moneda utilizada es Dolar. El país tiene 4 habitantes por km2 y se sitúa en el continente America.
Argentina	La capital de Argentina es Buenos Aires. La moneda utilizada es Peso. El país tiene 17 habitantes por km2 y se sitúa en el continente America.
Alemania	[ERROR] El país Alemanya no está en la base de datos.
Australia	La capital de Australia es Canberra. La moneda utilizada es Dolar. El país tiene 3 habitantes por km2 y se sitúa en el continente Oceanía.

[ ]:
```
# Respuesta
```

1.2.5 Ejercicio 5

Queremos crear un programa que permita practicar las tablas de multiplicar. Para ello, tendremos en cuenta los siguientes puntos **(2 puntos)**: - Multiplicaremos 2 números enteros aleatorios entre 0 y 10. Para ello, puedes utilizar la función random.randint de la librería Numpy. EG **(0.5**

puntos) - El usuario introducirá por pantalla la respuesta a la multiplicación. EG **(0.5 puntos)** - El programa comparará la respuesta del usuario con la correcta. Si la respuesta es correcta, mostrará un mensaje felicitando al usuario. Si la respuesta es incorrecta, avisará al usuario y mostrará la respuesta correcta. NM **(0.5 puntos)** - El programa se pondrá en forma de función: calculadora_multiplicacion(). NM **(0.5 puntos)**

Nota: En este enllace encontrarás información sobre la función *input*, que permite al usuario introducir datos mediante el teclado. Ten en cuenta que la función input siempre devuelve strings, por tanto, debe transformarse el input del usuario a integer.

[ ]:
```
# Para importar la librería numpy, utilizaremos lo siguiente:
import numpy as np
```

[ ]:
```
# Respuesta
```

Opcional

Tal y como se ha diseñado el ejercicio (siguiendo los puntos del apartado anterior), si el input que introduce el usuario no es un número, por ejemplo, si pone por equivocación un string, todo el programa falla. Cuando se da un error como éste en el código, Python para la ejecución y devuelve una excepción, que nos indica que se ha encontrado un problema en el programa.

Que Python nos dé información al encontrarse una excepción es muy útil, pero normalmente no queremos que el programa se detenga, sino que queremos capturar la excepción y actuar en consecuencia. Para ello, Python tiene la sentencia try-except.

Dado un código, probamos (try) de ejecutarlo y, en caso de encontrar un error, lo capturamos y hacemos algo al respecto (excepto) en vez de detener el programa.

Busca información sobre la sentencia try-except y aplícala a su función para que, en caso de introducir por error un input que no es un número, el programa muestre un mensaje de error avisando al usuario.

[ ]:
```
# Respuesta
```

1.2.6 Ejercicio 6

Hemos encontrado un dataset con información de boxeadores profesionales. La primera columna contiene la cabecera, que define cada una de las columnas (Ranking, Surname, Name, Country, Weight y Sex) y, a continuación, tenemos la información de los boxeadores ordenados por su ranking. **(1 punto)**

a) Abre el archivo y responde a las siguientes preguntas: NM **(0.5 puntos)**

- ¿Cuántas mujeres boxeadoras hay?
- ¿Cuál es el porcentaje de mujeres respecto al total de boxeadores (hombres y mujeres)? Muestra el resultado con sólo 2 decimales.

[ ]:
```
# Respuesta
```

b) Guarda, en un nuevo archivo de texto, sólo la información de los boxeadores españoles (es decir, aquellas filas donde Country sea “ES”). NM **(0.5 puntos)**

```
[ ]: # Respuesta
```