Curso de Especialización de «Desarrollo de Aplicaciones en Lenguaje Python»

Avanza 2024/25

Módulo «Estructuras de control en Python»

Unidad de trabajo 3.- Tareas

Profesor: Ismael Reyes Rodríguez

Tabla de contenido

Instrucciones previas

Esta práctica está compuesta de varios ejercicios. Una vez lo finalices, debes generar un PDF con un nombre que te identifique: **apellido1_apellido2_nombre_tarea3_EC.pdf**. Como vemos, el nombre del trabajo identifica al alumno, la tarea y el módulo (EC). Esta forma de nombrar a los documentos facilita su gestión de cara a corregirlos.

Cuando corrija vuestro trabajo, sobre vuestro PDF os realizaré las indicaciones y aclaraciones que considere oportunas y os lo entregaré indicando la nota del mismo.

Las directrices que seguiré para puntuar un trabajo las podéis encontrar en los documentos "Rúbrica.- Elaboración y entrega de tareas.pdf" y "Rúbrica.- Desarrollo de código.pdf", en la pestaña de "Inicio". En resumen, la idea es que las **tareas** entregadas sean **completas y claras** y el **código** que realicéis sea **correcto** y se interprete fácilmente.

Como comenté en el documento de "Conceptos previos y recomendaciones.pdf" del **Tema 1**, para probar algunos ejercicios utilizaremos el paquete "**pypas**". Este paquete os permitirá a vosotros y a mi comprobar que el código desarrollado es correcto por lo que, en la entrega de estos ejercicios, será necesario que no solo mostréis el código desarrollado, sino que también mostréis el resultado de la validación del código con *pypas*[1].

En cualquier caso, el **código desarrollado** es esencial de cara a evaluar la tarea por lo que, si no aparece en modo texto en el PDF (de modo que pueda hacer pruebas fácilmente), **comprimid** todos los programas en Python realizados en la tarea y **adjuntad** el fichero junto con el PDF a la tarea.

Para la resolución de los ejercicios podréis utilizar el IDE que os resulte más cómodo, el que mejor conozcáis.

[1] Recordad que se debe descargar cada ejercicio con *pypas get <ejercicio>*. De esta forma se obtiene un PDF con la descripción y un fichero ***main.py*** que es sobre el que se trabajará. Aún así, en este documento ya se incluye la descripción del ejercicio.

Ejercicio 1.- Varios ejercicios con pypas (10 puntos en total)

Los siguientes ejercicios se deben descargar con pypas y se deben entregar junto con la validación pertinente (ver el documento "**Conceptos previos y recomendaciones.pdf**", apartado 6: "*Prueba de los ejercicios con pypas*".

Ejercicio 1.1.- Isograma (1,25 puntos)

Descargar con pypas: **isogram**

Determina si una cadena de texto dada es un isograma, es decir, si no se repite ninguna letra.

Notas:

- No utilices la función *count*().
- Los guiones medios no cuentan como carácter repetido.
- Una letra minúscula es igual a una letra mayúsculas (a efectos de carácter repetido).

Ejemplo:

- **'downstream' sí** es un isograma.
- **'circus' no** es un isograma (se repite la 'c').

Ejercicio 1.2.- Número de palabras (0.75 puntos)

Descargar con pypas: **num-words**

Dada una cadena de texto, determina el número de palabras que contiene.

Notas:

- Se asume que todas las palabras vienen separadas por un espacio.

Ejemplo:

- Entrada: 'All you need is love'
- Salida: 5 (palabras).

Ejercicio 1.3.- Reorganizando fechas (1.5 puntos)

Descargar con pypas: **fix-date**

Transforma una fecha de entrada en otra de salida, pero modificando su formato.

Notas:

- Formato de fecha de entrada → <mes>/<día>/<año-con-2-dígitos>
- Formato de fecha de salida → <día-2dig>-<mes-2dig>-<año-4dig>
- Otro parámetro indicará el año base que debemos "sumar" al año de la fecha de entrada.
- Las fechas son cadenas de caracteres.

Ejemplo:

- Fecha de entrada: '12/31/23'
- Año base: 2000
- Salida: '31-12-2023'

Ejercicio 1.4.- Producto escalar (0.5 puntos)

Descargar con pypas: **dot-product**

Dados dos vectores (como listas), utilice la función ***zip*()** para calcular su producto escalar.

Notas:

- Si los vectores no tienen la misma dimensión (longitud) habrá que devolver *None*.

Ejemplo:

- Vectores:

  u = (4, 3, 8, 1)

  v = (9, 2, 7, 3)

- Cálculo y salida: u · v = 4 · 9 + 3 · 2 + 8 · 7 + 1 · 3 = 101

Ejercicio 1.5.- Contando letras (1 punto)

Descargar con pypas: **count-letters**

Cuenta el número de veces que aparece cada letra dentro de una cadena de texto dada.

Notas:

- Utiliza un diccionario para resolver el ejercicio.
- No puedes utilizar la función "built-in" *count*()

Ejemplo:

- Entrada: 'zoom'
- Salida: {'z': 1, 'o': 2, 'm': 1}

Ejercicio 1.6.- Tabla ASCII (1,5 puntos)

Descargar con pypas: **ascii-table**

A partir de un código de inicio y de un código de fin, muestra la Tabla ASCII correspondiente a los códigos de caracteres en el intervalo dado.

Notas:

- Organiza cada fila con 5 caracteres.
- Rellena con ceros los códigos de menos de 3 dígitos.
- Deja 3 espacios entre carácter y carácter.
- Los códigos de inicio y de fin también hay que incluirlos en la salida.

Ejemplo:

Si el código de inicio es 33 y el de final es 47, la salida debería ser la siguiente:

```
033 !   034 "   035 #   036 $   037 %
038 &   039 '   040 (   041 )   042 *
043 +   044 ,   045 -   046 .   047 /
```

Ejercicio 1.7.- Producto cartesiano (0,75 puntos)

Descargar con pypas: **cartesian**

A partir de dos cadenas de texto, computa el producto cartesiano letra a letra, dando como resultado un nuevo *string* completo.

Ejemplo:

- text1 = 'x1'
- text2 = 'y2'
- Producto cartesiano → 'xyx21y12'

Ejercicio 1.8.- Producto acumulado en cuadrados (0,5 puntos)

Descargar con pypas: **cumprod-sq**

Dado un número entero, calcula el producto acumulado de cada valor al cuadrado hasta llegar a dicho número.

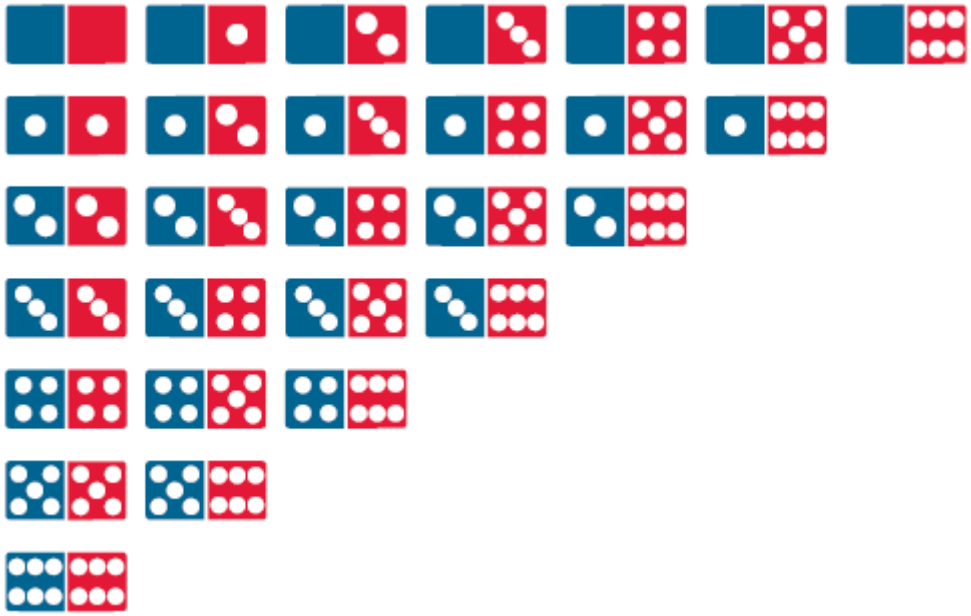
Ejemplo:

- Si n = 4 el resultado saldría del siguiente cálculo: $1^2 \cdot 2^2 \cdot 3^2 \cdot 4^2 = 576$

Ejercicio 1.9.- Dichas del domino (0,5 puntos)

Descargar con pypas: **domino**

Muestra por pantalla las fichas del dominó siguiendo esta estructura:



La salida debería ser la siguiente:

0|0 0|1 0|2 0|3 0|4 0|5 0|6

1|1 1|2 1|3 1|4 1|5 1|6

2|2 2|3 2|4 2|5 2|6

3|3 3|4 3|5 3|6

4|4 4|5 4|6

5|5 5|6

6|6

∗ La ficha “en blanco” se representa por un 0.

Ejercicio 1.10.- Máximo común divisor (0,75 puntos)

Descargar con pypas: **gcd**

Encuentra el máximo común divisor entre dos números. El *mcd* se define como el mayor número entero que divide a ambos números (con resto 0).

Notas:

- No es necesario utilizar ningún algoritmo existente.
- Basta con probar divisores.

Ejemplo:

- mcd(44, 12) = 4

Ejercicio 1.11.- Mitad fuera (0,5 puntos)

Descargar con pypas: **half-out**

Dado un conjunto de números enteros A, genera otro conjunto B con los valores de A divididos por 2.

Notas:

- Si el resultado de la división es un valor que ya existe **en A** no se debe incluir en B.
- Utiliza la división entera.

Ejemplo:

- Entrada: {50, 100, 4, 6, 12}
- Salida: {25, 2, 3}

Ejercicio 1.12.- Eliminando duplicados (0,5 puntos)

Descargar con pypas: **remove-dups**

Dada una lista de números, genera otra lista donde se eliminen los valores duplicados.

Notas:

- Se debe respetar el orden de los valores tal y como aparecen en la lista de entrada.

Ejemplo:

- Entrada: [4, 3, 7, 3, 3, 1, 4, 6]
- Salida: [4, 3, 7, 1, 6]