



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

Avanza 2024/25

Módulo «Estructuras de control en Python»

Unidad de trabajo 2.- Tareas

Profesor: Ismael Reyes Rodríguez

Tabla de contenido

Instrucciones previas.....	3
Ejercicio 1.- Varios ejercicios con pypas (7 puntos en total)	4
Ejercicio 1.1.- Atajos de teclado (1 punto)	4
Ejercicio 1.2.- Piedra, papel o tijera (1 punto)	5
Ejercicio 1.3.- Alfabéticamente (0.75 puntos).....	5
Ejercicio 1.4.- Todo son caritas (0.75 puntos)	6
Ejercicio 1.5.- Año bisiesto (0.75 puntos)	7
Ejercicio 1.6.- Marvel Akinator (1.5 puntos)	8
Ejercicio 1.7.- Operación simple (1.25 puntos)	8
Ejercicio 2.- Asignación condicional en una única línea (0.5 puntos)	9
Ejercicio 3.- Aplicación de un descuento (0.5 puntos)	9
Ejercicio 4.- Calificación con letras (0.5 puntos)	9
Ejercicio 5.- De número a texto (0.5 puntos)	10
Ejercicio 6.- Cálculo de IMC (0.5 puntos).....	10
Ejercicio 7.- Estaciones del año (0.5 puntos)	10

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_tarea1_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 (7 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.- Atajos de teclado (1 punto)

Este ejercicio ya se hizo en el **Tema 1**, indicándose que no era necesario utilizar sentencias “*match-case*”, ya que aún no se habían visto suficientes conocimientos teóricos de la misma. En este caso, se trata de realizar el mismo ejercicio, pero **sin utilizar sentencias «if»**. Si en el Tema 1 ya se resolvió el ejercicio con “*match-case*”, resolver este ejercicio con «if» anidados.

Nota: Aunque en este módulo no se profundiza sobre el uso de listas en Python, sería bueno que practicarais la creación de listas en base a los 3 parámetros introducidos y el uso de “*match-case*”.

Descargar con pypas: **shortcuts**

Combinando las teclas CTRL y ALT podemos conseguir muchos *shortcuts* (atajos de teclado). En el caso concreto de sistemas Linux tenemos aquí algunas de ellas:

CTRL+ALT+T	Open terminal
CTRL+ALT+L	Lock screen
CTRL+ALT+D	Show desktop
ALT+F2	Run console
CTRL+Q	Close window
CTRL+ALT+DEL	Log out

Se pide hacer un programa que reciba 3 valores de entrada (las 3 teclas pulsadas) y que obtenga la acción a llevar a cabo.

Notas:

- Cualquier combinación de teclas que no esté registrada conlleva la acción ‘*Undefined*’.
- Cuando no haya “tecla” vendrá como valor de entrada la cadena vacía.

Ejercicio 1.2.- Piedra, papel o tijera (1 punto)**Descargar con pypas: rps**

Implementa el juego de Piedra-Papel-Tijera (en inglés, rock-paper-scissors).

Notas:

- La entrada siempre vendrá en formato *string* con valores: 'rock', 'paper' o 'scissors'.
- La entrada puede estar en mayúsculas, minúsculas o mezcla de ambas.
- La salida deberá ser un valor numérico entero:
 - 1 si gana la primera jugadora.
 - 2 si gana la segunda jugadora.
 - 0 si hay empate.

Ejemplo:

- Jugadora 1 → 'rock'
- Jugadora 2 → 'SCISSORS'
- Salida → 1 (piedra vence a tijeras)

Ejercicio 1.3.- Alfabéticamente (0.75 puntos)**Descargar con pypas: isalpha**

Python dispone de la función *isalpha()* para determinar si todos los caracteres de una cadena de texto son alfabéticos. El objetivo de este ejercicio es re-implementar dicha función (con ciertas restricciones).

Notas:

- No debes usar la función (predefinida) *isalpha()*.
- Usa la siguiente constante → **ALPHABET = 'abcdefghijklmnopqrstuvwxyz'**
- Hay que contemplar también las letras en mayúsculas.
- Puedes utilizar el bucle *for*, visto en el Tema 1, para recorrer *strings*.

Ejemplo:

- 'hello-world' → No es alfabética (tiene un guion).
- 'Computer' → Sí es alfabética.

Ejercicio 1.4.- Todo son caritas (0.75 puntos)**Descargar con pypas: [facemoji](#)**

Partiendo de un sentimiento, muestra el emoji correspondiente.

Notas:

- Sentimientos admitidos:
 - Happy.
 - Sad.
 - Angry.
 - Pensive.
 - Surprised.
- Los sentimientos pueden venir en mayúsculas, minúsculas o mezcla de ambas.
- Si aparece un sentimiento no contemplado, se debe asignar *None*.

Para su realización, es necesario conocer **algo de teoría**.

Los programas de ordenador deben manejar una **amplia variedad de caracteres**. Simplemente por el hecho de la internacionalización hay que mostrar mensajes en distintos idiomas (inglés, francés, japonés, español, etc.). También es posible incluir «emojis» u otros símbolos.

Python utiliza el estándar **Unicode** para representar caracteres. Eso significa que tenemos acceso a una amplia carta de caracteres que nos ofrece este estándar de codificación.

Unicode asigna a cada carácter dos atributos:

1. Un **código numérico** único (habitualmente en hexadecimal).
2. Un **nombre** representativo.

Supongamos un ejemplo sobre el típico «emoji» de un **cohete** definido en este cuadro:



Representación Unicode del carácter ROCKET

La función **chr()** permite representar un carácter **a partir de su código**:

```
rocket_code = 0x1F680
rocket = chr(rocket_code)
print(rocket)
```

Salida: 

La función `ord()` permite obtener el código (decimal) de un carácter **a partir de su representación**:

```
rocket = chr(0x1F680)  
rocket_code = hex(ord(rocket))  
print(rocket_code)
```

Salida: 0x1f680

El modificador `\N` permite representar un carácter **a partir de su nombre**:

```
print('\N{ROCKET}')
```

Salida: 

Ejercicio 1.5.- Año bisiesto (0.75 puntos)

Descargar con pypas: [leap-year](#)

Dada una variable que representa un año, comprueba si dicho año es bisiesto o no lo es.

Notas:

- Un año es bisiesto en el calendario Gregoriano si es divisible entre 4 y no divisible entre 100, o bien si es divisible entre 400.
- Puedes hacer la comprobación en esta [lista de años bisiestos](#).

Ejemplo:

- El año 2012 es bisiesto.
- El año 2015 no es bisiesto.

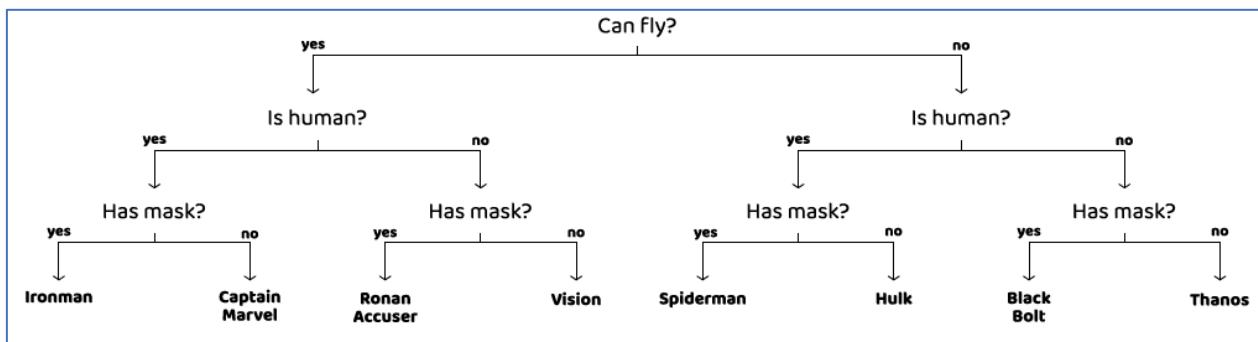
Ejercicio 1.6.- Marvel Akinator (1.5 puntos)

Descargar con pypas: [marvel-akinator](#)

Implementa un “clon” de [Akinator](#) que permita adivinar el personaje de [Marvel](#) en base a las tres preguntas siguientes:

1. ¿Puede volar? (variable `can_fly`)
2. ¿Es humano? (variable `is_human`)
3. ¿Tiene máscara? (variable `has_mask`)

Guíate por el siguiente árbol de decisión para resolverlo:



Ejercicio 1.7.- Operación simple (1.25 puntos)

Descargar con pypas: [simple-op](#)

El objetivo es poder computar una operación simple. Se dispondrá de tres datos de entrada:

- Primer valor numérico (entero).
- Segundo valor numérico (entero).
- Operador (*string*).

Debes calcular el resultado de la operación.

Notas:

- Utiliza la sentencia *match-case*.
- En caso de que la operación no sea '+', '-', '*' o '/' el resultado debe ser *None*.

Ejemplo:

- Entrada:
 - Primer valor → 7
 - Segundo valor → 4
 - Operador → '*'
- Salida: 28

Ejercicio 2.- Asignación condicional en una única línea (0.5 puntos)

Dados dos números **a** y **b** utiliza la asignación condicional en una única línea para asignar a la variable **maximo** el mayor de ambos. En otras palabras, escribe la línea que falta (# TODO) para que **maximo**, contenga el mayor de los dos números dados.

```
a = 5
b = 8
# TODO
print(maximo)
```

Salida: 8

Nota: Haz pruebas con varios números, no solo con 5 y 8. También es correcto que se recojan los valores de a y b con un *input*.

Ejercicio 3.- Aplicación de un descuento (0.5 puntos)

Escribe un programa que calcule el precio final de un producto aplicando un descuento basado en el precio original. El programa pedirá el precio de un producto y mostrará el precio final tras aplicar los siguientes descuentos:

- Menos de 100€: 5% de descuento.
- Entre 100€ y 500€: 10% de descuento.
- Más de 500€: 15% de descuento.

Ejemplo: Si el precio del producto es **1.000€** el precio tras el descuento será: $1000 - (1000 * 0.15) = 850€$

Ejercicio 4.- Calificación con letras (0.5 puntos)

Escribe un programa que pida una calificación numérica (0-100) e imprima su equivalente en letra. Estas equivalencias vienen dadas por:

- 90-100: A
- 80-89: B
- 70-79: C
- 60-69: D
- Menos de 60: F

Ejemplo: Si la calificación introducida es 82, la salida es la letra “B”.

Ejercicio 5.- De número a texto (0.5 puntos)

Escribe un programa que convierta un número introducido en su correspondiente texto. Solo serán válidos los números del 1 al 5.

Ejemplo: Si el número introducido es el 3, la salida será la cadena “Tres”.

Ejercicio 6.- Cálculo de IMC (0.5 puntos)

Escribe un programa que, en base a la altura (en metros) y el peso (en kilogramos) de una persona, calcule el IMC de una persona y clasifique el resultado. La altura y el peso serán de tipo *float*, el IMC es igual al *peso / altura²* y la clasificación de los resultados se recoge en la siguiente tabla:

ÍNDICE MASA CORPORAL	CLASIFICACIÓN
<16.00	Infrapeso: Delgadez Severa
16.00 - 16.99	Infrapeso: Delgadez moderada
17.00 - 18.49	Infrapeso: Delgadez aceptable
18.50 - 24.99	Peso Normal
25.00 - 29.99	Sobrepeso
30.00 - 34.99	Obeso: Tipo I
35.00 - 40.00	Obeso: Tipo II
>40.00	Obeso: Tipo III

En la siguiente página podemos contrastar los resultados obtenidos: [Calculadora de IMC](#).

Ejercicio 7.- Estaciones del año (0.5 puntos)

Escribe un programa que pida el número de un mes del año (1-12) e imprima la estación del año aproximada a la que pertenece. Se debe utilizar la sentencia “*match-case*” y patrones de comparación (ver el “*Ejemplo 2*” del apartado “4.4.- Ejemplos de uso” del documento de teoría del tema 2).

Ejemplo: Con el mes 12, la salida sería “Invierno” (esta es una aproximación).