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

Avanza 2024/25

# Módulo «Estructuras de control en Python»

# Unidad de trabajo 1.- 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_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" 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].

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

A continuación, se muestran algunos recursos Web que os pueden ser útiles para la realización de los ejercicios:

draw.io

pypas

[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 (6 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.- Mínimo de 3 valores (0.5 puntos)

| Descargar con pypas: **min3values** |
|---|

Dados 3 números calcula el valor mínimo.

## Ejercicio 1.2.- Donación de sangre (0.75 puntos)

| Descargar con pypas: **blood-donation** |
|---|

Escribe un programa que acepte edad, peso, pulso y número de plaquetas de una persona y determine si cumple con los requisitos para donar sangre.

Unidades de cada variable:

- La edad viene expresada en años.
- El peso viene expresado en kilogramos.
- El pulso viene expresado en pulsaciones.
- El número de plaquetas viene expresado en plaquetas por microlitro (mcL).

## Ejercicio 1.3.- Atajos de teclado (1 punto)

| 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.
- **En el Tema 2** se profundizará en diferentes formas en que se trabaja con la instrucción ***match-case*** que se podrían utilizar para desarrollar este ejercicio. Aún no las hemos visto por lo que recomiendo utilizar varias sentencias *if* junto con alguna *match-case* (si lo consideráis necesario).

### Ejercicio 1.4.- Sumando múltiplos (0.75 puntos)

Descargar con pypas: **m3-sum-limited**

Escribe un programa que encuentre la mínima secuencia de múltiplos de 3 (distintos) cuya suma sea igual o superior a un valor dado.

***Ejemplo:***

Si el valor de entrada es igual a 45, la salida debería ser: 0 3 6 9 15

Esto es así porque 0 + 3 + 6 + 9 + 12 + 15 = 45 por lo tanto es la mínima secuencia de múltiplos de 3 cuya suma es igual o superior a 45.

### Ejercicio 1.5.- El árbol del 1 (0.75 puntos)

Descargar con pypas: **one-tree**

Escribe un programa en Python que realice las siguientes 9 multiplicaciones:

$$1 \cdot 1$$

$$11 \cdot 11$$

$$111 \cdot 111$$

$$1111 \cdot 1111$$

$$11111 \cdot 11111$$

$$111111 \cdot 111111$$

$$1111111 \cdot 1111111$$

$$11111111 \cdot 11111111$$

$$111111111 \cdot 111111111$$

**Notas**

- ¿Notas algo especial en los resultados parciales?
- No es necesario "pintar" el árbol, solo obtener los resultados y mostrarlos por pantalla.

## Ejercicio 1.6.- Caballo de ajedrez (0.75 puntos)

Descargar con pypas: **chess-horse**

En este ejercicio vamos a simular el movimiento de un caballo de ajedrez sobre un plano.

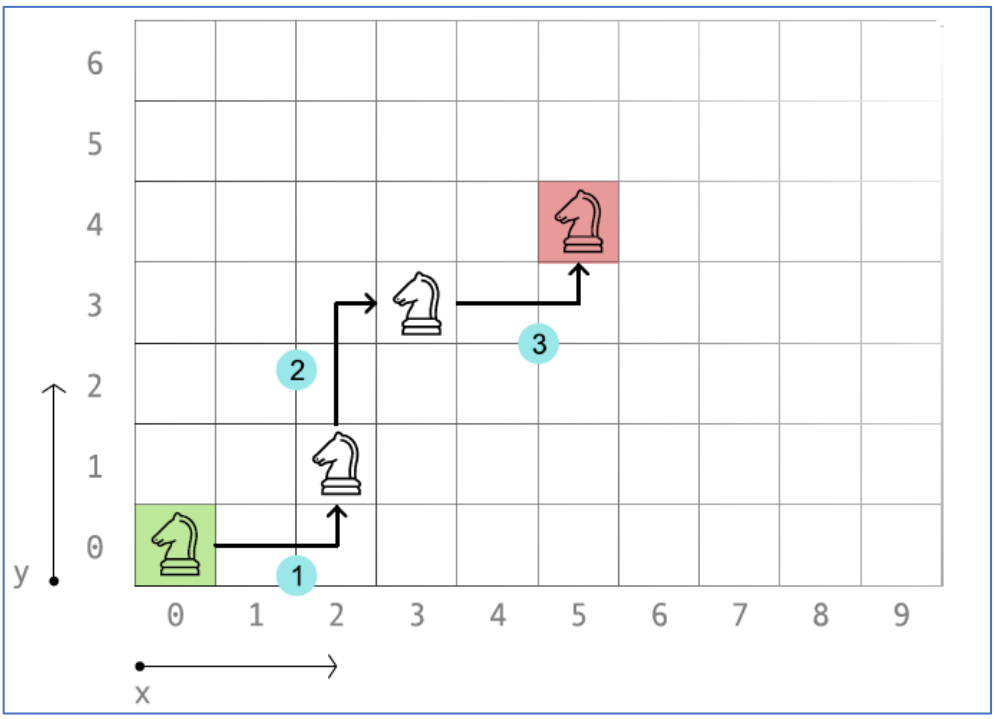
El dato de entrada será la casilla de destino, que vendrá dada por dos valores enteros x e y. El objetivo será determinar cuántos movimientos harán falta para llegar a la casilla de destino.

**Notas:**

- El caballo siempre parte de la celda (0, 0) y cada "turno" consta de dos movimientos.
- El primer movimiento será de 2 unidades en el eje x junto a 1 unidad en el eje y.
- El segundo movimiento será de 2 unidades en el eje y junto a 1 unidad en el eje x.
- Todos los movimientos son "en positivo".
- Si la casilla de destino es inalcanzable debemos devolver -1.

***Ejemplo:***

Supongamos que la casilla de destino es (5, 4). Los movimientos serían los siguientes:




Por lo tanto, harán falta 3 movimientos para llegar al destino indicado.

### Ejercicio 1.7.- Adivina un número (0.75 puntos)

Descargar con pypas: **guess-number**

Escribe un programa que permita al usuario adivinar un número (que será dato de entrada).

**Notas**:

- Habrá que indicar en cada "turno" si el número buscado es menor o mayor que el introducido.
- Una vez acertado, habrá que mostrar un mensaje con el número de intentos realizados.

***Ejemplo***:

Supongamos que el número a adivinar es 87. La salida debería quedar tal que así:

*Introduzca número: 50*

*Mayor*

*Introduzca número: 100*

*Menor*

*Introduzca número: 90*

*Menor*

*Introduzca número: 87*

*Enhorabuena has encontrado el número en 4 intentos*

### Ejercicio 1.8.- Distancia de Hammming (0.75 puntos)

Descargar con pypas: **hamming**

Calcula la distancia de hamming entre dos cadenas de texto dadas.

**Notas**:

- Si las cadenas de texto tienen distinta longitud habrá que devolver -1.

La distancia de Hamming entre dos cadenas dadas de la misma longitud es el número de caracteres que son diferentes comparando cada uno de ellos uno a uno (ocupan la misma posición en las cadenas).

## Ejercicio 2.- Diagrama de flujo de la sucesión de Fibonacci (1.5 puntos)

Para completar este ejercicio habrá que:

2.1.- Crear un **diagrama de flujo** que muestre cómo se resolvería el problema propuesto (0.75 puntos)

2.2.- Hacer el **programa en Python** que resuelva el problema (0.75 puntos)

El problema que debemos descargar también con *pypas* se define a continuación.

| Descargar con pypas: **fibonacci** |
|---|

En matemáticas, la sucesión de Fibonacci es una sucesión infinita de números naturales como la siguiente: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, …

El objetivo de este ejercicio es encontrar el "enésimo" término de la sucesión de Fibonacci.

**Notas**

- Siempre se cumple que el primer valor de la sucesión de Fibonacci es el cero y el segundo valor es el uno.
- A partir de ahí, cada nuevo valor se calcula como la suma de los dos valores anteriores.
- Utiliza un bucle para implementar la solución.

***Ejemplo***

Si n = 10 habrá que devolver 55 ya que es el valor que ocupa la posición 10 en la sucesión de Fibonacci (contando desde cero).

## Ejercicio 3.- Diagrama de flujo del Mínimo de una función (2 puntos)

Para completar este ejercicio habrá que:

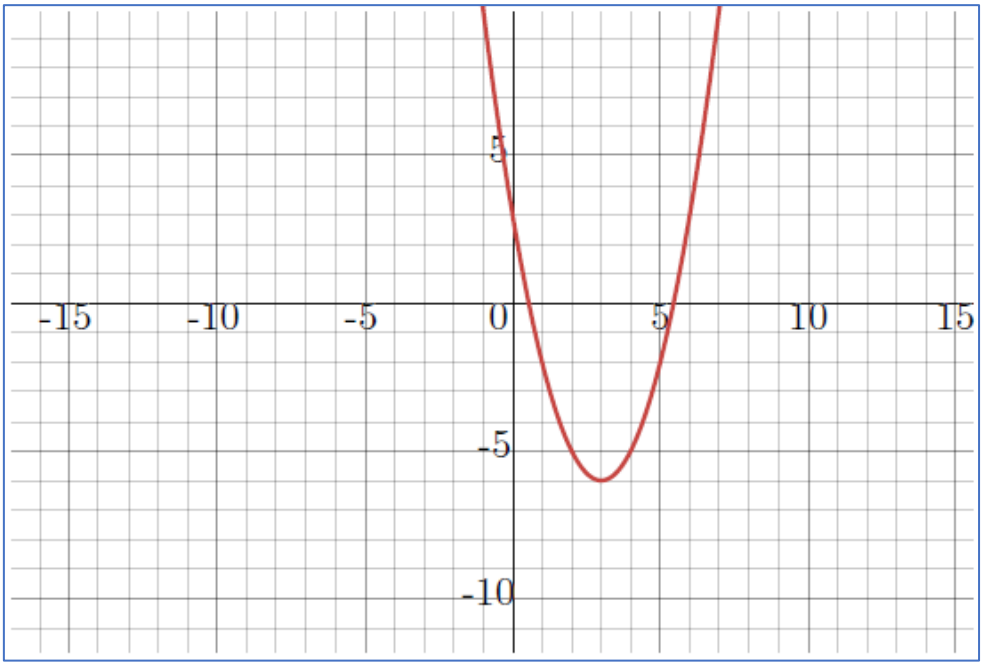
3.1.- Crear un **diagrama de flujo** que muestre cómo se resolvería el problema propuesto (1 puntos)

3.2.- Hacer el **programa en Python** que resuelva el problema (1 punto)

El problema que debemos descargar también con *pypas* se define a continuación.

| Descargar con pypas: **fmin** |
|---|

Partiendo de una función $f(x) = x^2 - 6x + 3$ que tiene la siguiente gráfica:




Encuentra x para el que el valor de la función sea el mínimo, dado un intervalo de búsqueda [x1, x2].

**Notas**:

- Realiza la búsqueda ´únicamente sobre valores enteros de x.
- Habrá que calcular tanto el valor x como el valor de la función f(x).
- Ojo que el intervalo de búsqueda es cerrado. Esto quiere decir que hay que incluir en la búsqueda los valores extremos.

***Ejemplo***:

Supongamos que el intervalo de búsqueda es x1 = −9, x2 = 9 → [−9, 9]. En dicho intervalo el menor valor de la función se da en x = 3 y la función vale f(3) = −6.

En otras palabras, la función *run* del programa de Python main.py recibirá un intervalo de número enteros, en base a dos valores, "*x1*" y "*x2*". Se deben recorrer todos los enteros entre "*x1*" y "*x2*", almacenando cada valor del intervalo en una variable "*x*", calcular para cada valor de "*x*" el valor de f(x), que será el resultado de sustituir el valor de "*x*" en x*x – 6x +3. Este valor lo podremos guardar en una variable "*y*", por ejemplo, y tener su valor en cuenta (usando variables intermedias) para saber finalmente con qué valor de "*x*", la función f(x), es decir nuestra variable "y", es el menor posible de todo el intervalo.

## Ejercicio 4.- Diagrama de flujo en base a un pseudocódigo (0.5 puntos)

Para calcular cuál de dos números dados es el mayor se ha diseñado el siguiente pseudocódigo:

1. Inicio
2. Inicializar variables: A = 0, B = 0
3. Solicitar la introducción de dos valores distintos
4. Leer los dos valores
5. Asignarlos a las variables A y B
6. Si A=B Entonces vuelve a 3 porque los valores deben ser distintos
7. Si A>B Entonces
   Escribir A, "es el mayor"
8. De lo contrario
   Escribir B, "es el mayor"
9. Fin Si
10. Fin

Crea el diagrama de flujo que represente el pseudocódigo dado.