Pasos hacia la programación con Python (II)

Control de flujo

A continuación abordaremos una de las partes fundamentales de cualquier programa informático: el control del flujo de ejecución del programa y con ello de las acciones que realiza. Sin ello los programas carecerían de una lógica de decisión interna y las acciones que podríamos implementar serían muy limitadas.

Ya hemos tratado con anterioridad múltiples elementos de Python relacionados con el control de flujo como son: bucles, condicionales, funciones, las sentencias *continue*, *break*, etc.

En esta sección repasamos todos estos elementos y vamos un paso más haya explicando algunos conceptos nuevos y abordando los escenarios de control de flujo más comunes de una manera metódica.

Esta sección es fundamental y solo con las herramientas desarrolladas hasta aquí, conocer como usar librerías externas y unas nociones extra sobre objetos, ya podríamos desarrollar la mayor parte de tareas que un desarrollador de Python realiza en el día a día.

Por todo esto, no infravaloréis esta sección puesto que es probablemente la más importante junto a la de objetos. Esta sección esta basada en la propia documentación de Python3.8 en castellano, en concreto el capítulo 4 de la misma:

https://docs.python.org/es/3/tutorial/controlflow.html

Control del flujo (1)

Tenéis disponible el repaso en el repositorio de GitHub python_basics_1. Esta en el archivo control_flujo_1.py

Abordamos los puntos del 1 al 5 del citado tutorial.

ejercicio3.py

Empleando la sentencia if, bucles, la sentencia continue y el operador modulo (%). Crear 4 bucles donde se vayan llenando las siguientes listas:

1. Números múltiplos de 2 y 3 del 0 al 100.
2. Lista de listas con un número y su cuadrado, del 0 al 10. Es decir: *[[1, 1], [2, 4], [3, 9] …]*
3. Cada una de las letras en posiciones pares del string: *my_string = "Uno de los elementos fundamentales para el control del flujo es la sentencia IF"*.

4. A continuación itera de manera simultanea las tres listas usando la función zip y printa los tres primeros elementos, los tres segundos, … ¿En que momento se para el bucle? ¿Queda algún elemento sin printar?

ejercicio4.py

Crea un programa compuesto por dos bucles anidados que recorran el rango de 1 a 20. Usando continue y break debes imprimir en pantalla lo siguiente:

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4 5 6

1 2 3 4 5 6 7

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10 11

1 2 3 4 5 6 7 8 9 10 11 12

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13 14

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Recuerda usar print(m, end= " ").

ejercicio5.py

Copia en un programa el siguiente código de Python:

```
for n in range(1, 100):
  for m in range(1, 100):
    if n == m:
      continue
    if n % m == 0:
      break
  else:
    print(n, m)
```

¿Cual es la salida del mismo? ¿Porqué se da esta salida?

ejercicio6.py

Crea un programa que imprima por pantalla el siguiente triángulo de números impares:

```
1
3 1
5 3 1
7 5 3 1
9 7 5 3 1
```

Control del flujo (2)

Tenéis disponible el repaso en el repositorio de GitHub python_basics_1. Esta en el archivo control_del_flujo_2.py

Abordamos los puntos del 6 al 9.

ejercicio7.py

Crear las siguientes funciones:

1. Una función denominada funcion1 que reciba un argumento posicional y dos argumentos por nombre. Inicializa estos valores por nombre a algún valor por defecto. La función debe printar en pantalla las frase: “Estos son mis argumentos: (arg1, kwarg1, kwarg2)”, donde arg1, kwarg1, kwarg2 deben ser los valores de los argumentos. Pon un ejemplo de uso de la función.

2. Una función denominada suma_argumentos que tome un número arbitrario de argumentos y devuelva la suma de estos. Pon un ejemplo de uso de la función con un print.
3. Define una variable llamada var, crea una función que no reciba ni devuelva argumentos y que cada vez que sea llamada eleve al cuadrado la variable var. Pon ejemplos de uso.
4. Crea una función que se llame odenar_lista que usando la función sort y las funciones lambda permita ordenar la siguiente lista en base a al primer elemento de cada sublista:

   [[4, "tigre"], [2, "jirafa"], [6, "cebra"], [7, "cocodrilo"], [1, "puma"], [5, "cachalote"], [8, "cabra"]]

   La lista anterior debe asignarse a la variable lista_originial. La función debe recibir un único argumento y devolver la lista ordenada. Queremos asignar la salida de la función a una variable nueva llamada lista_nueva de tal forma que la lista original no sea modificada.
5. Crear una función llamada función externa que tome dos parámetros a y b. Dentro de esta función crear otra función llamada suma. Esta función debe tomar los valores de a y b, sumarlos y devolver el resultado. Asigna el resultado de esta función interna a una variable llamada suma y retorna esta variable. Comprueba que funciona correctamente. ¿Podrías hacer esto mismo de tal forma que la función suma no reciba argumentos, suma()?

ejercicio8.py

Clara ha incluido el código del boletín de calificaciones en una función generate_report_card para poder generar un boletín de calificaciones dinámicamente a partir de los datos del alumno que se le faciliten.

Ha pasado esta función a sus colegas para que puedan introducir los datos del alumno, es decir, ***la asignatura, la puntuación, el nombre del alumno y la fecha del examen***, y generar el boletín de notas. Sin embargo, a algunos de ellos les resultó difícil utilizarla porque no sabían en qué orden debían pasar los argumentos.

Para que los argumentos de la función queden claros, Clara ha decidido convertirlos en argumentos nombrados o argumentos keyword, de forma que el orden no importe. Ayuda a Clara convirtiendo los argumentos de generate_report_card en argumentos de palabra clave.

Una vez hecho esto, llama a la función para ver que todo a funcionado como esperábamos. Después trata de realizar el mismo proceso pero pasando todos los datos del alumno codificados en un diccionario. Podéis encontrar el código de partida en el programa ***ejercicio8.py*** contenido en el repo.

ejercicio9.py

Crea una función que sume los números de 1 a 100 empleando recursividad de funciones.

ejercicio10.py

Crea una función que devuelva los elementos de la sucesión de Fibonacci menores de 200 empleando recursividad.