



PROGRAMA: ESPECIALIZACION EN PYTHON
MÓDULO I

Quiz Final

Indicar Verdadero o Falso las siguientes sentencias según sea el caso (Por cada pregunta bien contestada: 1pto):

1. Python nos permite acortar el tiempo de desarrollo por su fácil entendimiento y lenguaje completamente intuitivo.
Lenguaje multiparadigma. Esto significa que combina propiedades de diferentes paradigmas de programación, lo que permite que sea muy flexible y fácil de aprender tanto de utilizarlo en campos aparentemente tan dispares como el diseño de aplicaciones web o la inteligencia artificial, por ejemplo.
(VERDADERO)
2. Para la gestión de dependencia en Python nos da a entender a la instalación de scripts que se suelen desarrollar apoyándose en librerías y/o **frameworks** de terceros a los cuales llamamos **dependencias** como es **Django** para el desarrollo web, para esto instalamos la librería del siguiente modo en nuestro terminal: **(VERDADERO)**

```
pip install django==1.10
```

3. Dentro de la conversión de tipos en Python tenemos las funciones operan de la misma manera: siempre esperan un argumento sobre el cual realizar la conversión. Como `str()`, `float()`, `int()` o `bool()`.
Al ejecutar la siguiente línea de código nos devolverá un entero: **(FALSO)**

```
print(int("23234e323732A2"))
```

4. En las estructuras de control de flujo en programación nos permite evaluar sobre una condición y especificar qué acciones ha de realizar el programa dependiendo del resultado de la evaluación de la condición, si es **verdadera o falsa**.

Las condicionales pueden ser simples o múltiples, y solo pueden devolver dos resultados, verdadero o falso (True o False).

Sabiendo esto en la siguiente condición múltiple nos estaría devolviendo falso: **(FALSO)**

```
3 == 3 or 15 < 7
```

5. El uso del bucle **for** sirve fundamental para recorrer los elementos de un objeto iterable (lista, tupla, conjunto, diccionario, etc) y poder ejecutar un bloque de código respectivamente.

Tenemos el siguiente ejemplo con el uso de **for**: **(VERDADERO)**

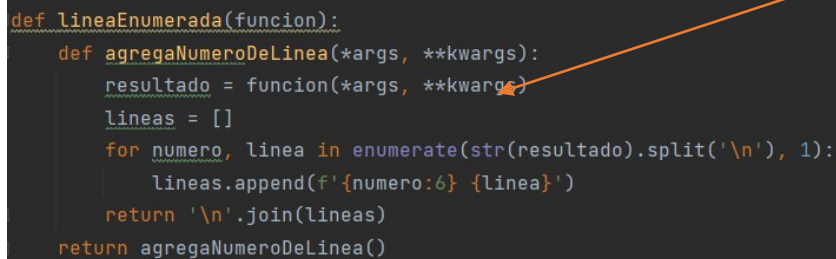
```
for num in range(0, 11, 2):  
    print(num)
```

Nos devolverá como resultado: **(FALSO)**, no devuelve el 5

```
0  
2  
4  
5  
6  
8  
10
```

6. Cuando se habla de POO en Python, se menciona a las Clases y los métodos por ende se definen o escriben de igual forma que las funciones normales, pero deben declararse dentro de la clase y su primer argumento siempre referencia a la instancia que la llama, de esta forma se afirma que los métodos son funciones. **(VERDADERO)**

7. Los decoradores son funciones que reciben otras funciones como parámetros y retornan funciones diferentes. Las cuales también nos permite modificar el comportamiento de otras funciones.
Entonces tenemos entendido que para el uso de decoradores primero se va a ejecutar la lógica del decorador y luego la lógica de una función decorada. **(VERDADERO)**
8. Los decoradores en Python se utilizan para alterar el funcionamiento de una determinada pieza o bloque de código; ya sea una función, o una clase, sin la necesidad de emplear otros mecanismos como la herencia.
En el siguiente ejemplo:



```
def lineaEnumerada(funcion):  
    def agregaNumeroDeLinea(*args, **kwargs):  
        resultado = funcion(*args, **kwargs)  
        lineas = []  
        for numero, linea in enumerate(str(resultado).split('\n'), 1):  
            lineas.append(f'{numero:6} {linea}')  
        return '\n'.join(lineas)  
    return agregaNumeroDeLinea()
```

El uso especial de ***args** nos permite contener todos los argumentos ordenados por la posición en la que fueron enviados desde la función original al momento de usarlo en la función decoradora.
(VERDADERO)