

# SILMULACRO DE PARCIAL

## PREGUNTAS TEÓRICAS

### Pregunta\_1

En Python no podemos utilizar esta palabra como referencia o variable dado que es una palabra reservada (seleccionar la correcta, respetando mayúsculas y minúsculas):

☐ variable

☐ except

☐ true

☐ AND

☐ landa

### Pregunta\_2

Indicar cuál de los siguientes **no** hace referencia a un tipo de literal:

☐ Float

☐ Lista

☐ Función

☐ String

☐ Booleano

### Pregunta\_3

Señala cuál de los siguientes, **no** corresponde a un operador de asignación:

☐ x \*= 5

☐ x = 5

☐ x /= 5

☐ x == 5

☐ x \*\*= 5

## ANÁLISIS DE CÓDIGO

Observe el siguiente código:

```
1 def EsEntero():
2     _edad = "e"
3     while(True):
4         _edad = input("ingrese una edad válida ")
5         if _edad.isnumeric() == False:
6             print("La edad no es numerica! ")
7             continue
8         else:
9             break
10    return int(_edad)
11 def FuncionEdadPersona():
12     edad = EsEntero()
13     while (True):
14         if (int(edad) > 0 and int(edad) < 100):
15             break
16         else:
17             print("La edad no es un valor entre 0 y 100!")
18             edad = EsEntero()
19     print(f"Ha ingresado correctamente la edad:{edad} años.")
20     return edad
21
```

```
22 def GeneroPersona():
23     _genero = input("¿Qué género tiene masculino [M] o femenino [F]?")
24     if _genero.upper() == "M":
25         print("Ha ingresado correctamente el género: Masculino")
26         return _genero.upper()
27     elif _genero.upper() == "F":
28         print("Ha ingresado correctamente el género: Femenino")
29         return _genero.upper()
30     else:
31         print("genero incorrecto, vuelva a intentar")
32         GeneroPersona()
```

```
33
34 def progamaJubilatorio(_edad, _genero):
35     if _genero.upper() == "M":
36         if int(_edad) > 64:
37             print("Usted está jubilado")
38         else:
39             print(f"Le quedan por aportar {65 - _edad} año/s")
40     else:
41         if int(_edad) > 60:
42             print("Usted está jubilado")
43         else:
44             print(f"Le quedan por aportar {60 - _edad} año/")
45
46 print("Bienvenido al programa jubilatorio")
47 edad = FuncionEdadPersona()
48 genero = GeneroPersona()
49 progamaJubilatorio(edad, genero)
50
51
52
```

## Responder Verdadero o Falso

1 El *continue* de la línea 7 permite volver a la línea 3 las veces que sean necesarias hasta que el usuario ingrese un valor numérico.

☐ Verdadero

☐ Falso

2 La Función *EsEntero()* retornará a la línea 47 un valor solamente cuando el usuario ingrese un valor entero.

☐ Verdadero

☐ Falso

3 El *break* de la línea 15 hará que el bucle *while* de la función se rompa únicamente cuando el valor ingresado sea un valor alfanumérico.

☐ Verdadero

☐ Falso

4 La única función que requiere parámetros es *FuncionEdadPersona()*

☐ Verdadero

☐ Falso

5 La función *GeneroPersona()* tiene una estructura selectiva que conduce a dos caminos posibles.

☐ Verdadero

☐ Falso

6 El cuerpo del programa va de las líneas de código 46 a 49

☐ Verdadero

☐ Falso

## EJERCICIO POR DESARROLLAR

Junto con unos amigos estás desarrollando un juego de ruleta para utilizarlo en un sorteo. El programa inicia con el siguiente menú:

-----

- Sorteo los buenos amigos -

- Bienvenido -

Menú:

[1] Cargar Amigos al sorteo

[2] Sortear!

[3] Vaciar lista de amigos

[4] Subir la apuesta!

[0] Salir del programa

-----

Se te proveerá de un archivo *template.py* con la estructura del menú armada, en donde deberás desarrollar el código de las funciones que se solicitan a continuación (aparecen tres puntos ... en todos los lugares donde se espera tu código). Ten en cuenta que el programa se compone de varios bloques:

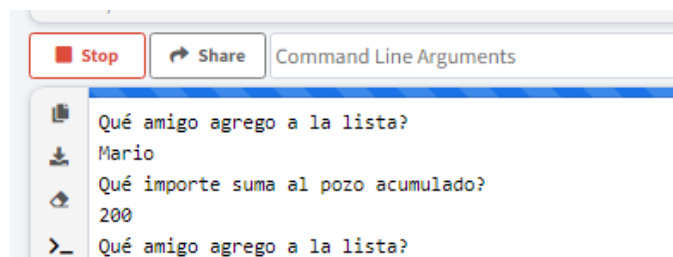
- 1- Bloque de importación: Aquí deberás agregar las importaciones necesarias para que el programa funcione.
- 2- Bloque de funciones: Esta será la sección en donde desarrollarás el código para cada función.
- 3- Bloque de declaración de variables: Aquí deberás declarar las variables y listas que necesites. En principio, necesitarás dos listas. A una la llamarás **listaDeAmigos** y a la otra **listaDeApuestas**, las cuales funcionarán de manera paralela, es decir que un mismo índice en ambas listas corresponderán a los datos del mismo amigo.
- 4- Bloque de menú: Aquí deberás llamar a las correspondientes funciones.

### [1] Cargar Amigos al sorteo

Al seleccionar esta opción, el usuario podrá cargar todos los amigos que participarán del sorteo y el valor en pesos que cada amigo agrega al pozo. Si se ingresa "fin" como nombre se regresará al menú.

Desarrollar una función que se llame **agregarAmigosAlSorteo(\_listaDeAmigos, \_listaDeApuestas)**

Pantalla ejemplo



### [2] Sortear!

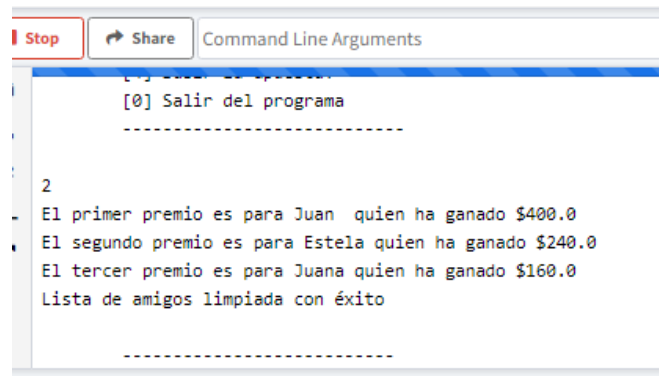
Se sortean aleatoriamente tres amigos de la lista a través de la función **realizarSorteo(\_listaDeAmigos, \_listaDeApuestas)**

El primero de ellos se lleva el 50% del pozo, el segundo el 30% y el tercero el 20%

Si un amigo ya salió sorteado no puede obtener otro premio (hacer las validaciones para que esto no ocurra). Se imprimen los resultados del sorteo y se limpian ambas listas al volver al menú. Utiliza la variable (privada en la función) **pozoAcumulado** para almacenar el valor del pozo total sobre el que harás los cálculos de los premios.

IMPORTANTE: Se deberá mostrar un mensaje si hay menos de 4 participantes, ya que el sorteo sólo se realizará si hay 4 o más amigos participando.

#### Pantalla ejemplo



```
Stop Share Command Line Arguments
[0] Salir del programa
-----
2
El primer premio es para Juan quien ha ganado $400.0
El segundo premio es para Estela quien ha ganado $240.0
El tercer premio es para Juana quien ha ganado $160.0
Lista de amigos limpiada con éxito
-----
```

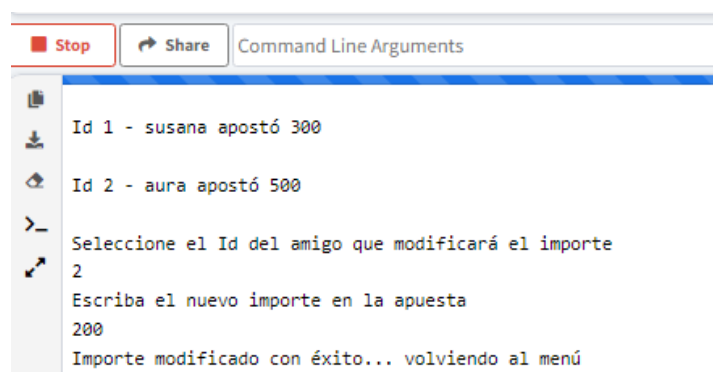
### [3] Dar de baja un amigo

Desarrollar la función **bajaDeAmigo(\_listaDeAmigos, \_listaDeApuestas)** que muestre por pantalla un listado con: la posición del jugador (el índice 0 es válido); el nombre del jugador; y el valor que aportó al pozo. Luego deberá preguntar por la posición del amigo a eliminar del juego, para de esa manera sacarlo de las listas. Validar que se ingrese una posición válida de amigo.

### [4] Subir la apuesta!

Para la función **subirApuesta(\_listaDeAmigos, \_listaDeApuestas)** se deberá mostrar por pantalla el mismo listado del punto anterior. Luego el programa deberá preguntar por la posición del amigo que quiere subir la apuesta, y luego preguntar por el monto que quiere agregar al que ya apostó, para luego modificar el importe de su apuesta. Se mostrará un mensaje de éxito y se regresará al menú principal.

#### Pantalla ejemplo



```
Stop Share Command Line Arguments
Id 1 - susana apostó 300
Id 2 - aura apostó 500
> Seleccione el Id del amigo que modificará el importe
2
Escriba el nuevo importe en la apuesta
200
Importe modificado con éxito... volviendo al menú
```