

## Examen de promoción de Programación y Métodos Numéricos 2023

Nombre y Apellido:

Matricula:

1) (50 puntos) Analizar el siguiente programa y responder las preguntas.

A) (10 puntos) Sobre la función func1

- ¿Cuántos parámetros formales tiene? → **2**
- ¿De qué tipo de datos son los parámetros? → **Una lista y un entero**
- ¿De qué tipo de datos es el valor de retorno? → **Booleano (True o False)**
- ¿Para qué sirve esta función? → **Permite averiguar si una lista contiene o no un número.**

B) (10 puntos) Sobre la función func2

- ¿Cuántos parámetros formales tiene? → **2**
- ¿De qué tipo de datos son los parámetros? → **Una lista y un entero**
- ¿De qué tipo de datos es el valor de retorno? → **Un número entero**
- ¿Para qué sirve esta función? → **Cuenta la cantidad de veces que aparece un número en una lista**

C) (10 puntos) Cuánto vale la variable **vector** inmediatamente después de ejecutar las líneas 15, 22 y 29

**Inmediatamente después de la línea 15** → [1, 11, 1, -10, -12, 67, 9, 8, 9]

**Inmediatamente después de la línea 22** → [1, 10, 1, -10, -10, 10, 9, 8, 9]

**Inmediatamente después de la línea 29** → [1, 10, 1, -10, -10, 10, 9, 8, 9]

D) (10 puntos) Cuánto vale la variable **u** inmediatamente después de ejecutar las líneas 24 y 29

**Inmediatamente después de la línea 24** → []

**Inmediatamente después de la línea 29** → [1, 10, -10, 9, 8]

E) (10 puntos) Complete la salida por pantalla al ejecutar el programa.

**El valor 1 se repite 2 veces**  
**El valor 10 se repite 2 veces**  
**El valor -10 se repite 2 veces**  
**El valor 9 se repite 2 veces**  
**El valor 8 se repite 1 veces**

```
1. def func1(v, e):
2.     for ei in v:
3.         if ei == e:
4.             return True
5.     return False
6.
7.
8. def func2(v, e):
9.     cant = 0
10.    for ei in v:
11.        if ei == e:
12.            cant+=1
13.    return cant
14.
15. vector = [1, 11, 1, -10, -12, 67, 9, 8, 9]
16.
17. for i in range(len(vector)):
18.     if vector[i] > 10:
19.         vector[i] = 10
20.     if vector[i] < -10:
21.         vector[i] = -10
22. print(vector)
23.
24. u = []
25. for val in vector:
26.     if not func1(u, val):
27.         u.append(val)
28.
29. print(u)
30.
31. for e in u:
32.     c = func2(vector, e)
33.     print ("El valor %d se repite %d veces" % (e,c))
```

2) (25 puntos) Dado el siguiente arreglo de numpy completar el código para cumplir con lo siguientes:

- A) Eliminar los valores que sean 0.0.
- B) Calcular y mostrar por pantalla la media y la desviación estándar de los valores.
- C) Crear un nuevo arreglo que contenga aquellos valores del arreglo original que sean mayores a la media.
- D) Crear un nuevo arreglo que contenga aquellos valores del arreglo original que sean menores a la media.

```
import numpy

vals = numpy.array([51.15158804, 62.22974702, 0.0, 19.74917581, 52.93135669,
                    52.2471187, 47.11465944, 40.07714657, 0.0, 41.78177974,
                    39.35785313, 23.08072553, 32.15491727, 28.69310516,
                    62.29105077, 54.81178278, 0.0, 27.34538919, 26.82504438,
                    67.32490527, 51.26240147])

vals = vals[vals!=0.0]
print()
print("vals: ", vals)

x_barra = vals.mean()
desv = vals.std()

print()
print("La media de la muestra es: %.2f"%x_barra)
print("La desviación de la muestra es: %.2f"%desv)

vals_may = vals[vals>x_barra]
vals_men = vals[vals<x_barra]

print()
print("vals_may: ", vals_may)
print()
print("vals_men: ", vals_men)
```

**SALIDA POR PANTALLA:**

```
vals: [51.15158804 62.22974702 19.74917581 52.93135669 52.2471187 47.11465944
40.07714657 41.78177974 39.35785313 23.08072553 32.15491727 28.69310516
62.29105077 54.81178278 27.34538919 26.82504438 67.32490527 51.26240147]
```

La media de la muestra es: 43.36

La desviación de la muestra es: 14.13

```
vals_may: [51.15158804 62.22974702 52.93135669 52.2471187 47.11465944 62.29105077
54.81178278 67.32490527 51.26240147]
```

```
vals_men: [19.74917581 40.07714657 41.78177974 39.35785313 23.08072553 32.15491727
28.69310516 27.34538919 26.82504438]
```

3) (25 puntos) El siguiente programa corresponde a una prueba de hipótesis para la muestra almacenada en vals. Analizar el código y la salida por pantalla y responder:

A) (5 puntos) Completar las líneas 11 y 12.

```
11. sigma_cero = vals.std()/len(vals)**0.5          # desviación de la distribución de la media
12. x_barra = vals.mean()                          # Estadístico de prueba
```

B) (5 puntos) ¿Cuál es la hipótesis nula y cuál la hipótesis alternativa que supone la prueba?

**H0: Que la media muestral es 35**

**Ha: Que la media muestral no es 35**

C) (5 puntos) Luego de realizar la prueba se decide rechazar la hipótesis nula, ¿qué valores de la salida se

**utilizan para sustentar este resultado teniendo en cuenta el valor de significancia alfa?**

**Se utilizan: el límite superior de 42.03 comparado con el estadístico de prueba que es 43.36 y el p-valor que da muy chico (0.02)**

D) (5 puntos) Esta prueba es una prueba de cola superior, inferior o de doble cola?

**Es una prueba de doble cola**

E) (5 ) ¿Cómo deberíamos modificar la línea 24 si el estadístico de prueba es menor que mu cero?

```
24. p_valor = 2*scipy.stats.t.cdf(x_barra, df=gl, loc=mu_cero, scale=sigma_cero)
```

```
1. import numpy
2. import scipy.stats
3.
4. vals = numpy.array([51.15158804, 62.22974702, 19.74917581, 52.93135669,
5.                      52.2471187, 47.11465944, 40.07714657, 41.78177974,
6.                      39.35785313, 23.08072553, 32.15491727, 28.69310516,
7.                      62.29105077, 54.81178278, 27.34538919, 26.82504438,
8.                      67.32490527, 51.26240147])
9.
10. mu_cero = 35.0                                # media poblacional de la hipótesis nula
11. sigma_cero =                                # desviación de la distribución de la media
12. x_barra =                                    # Estadístico de prueba
13.
14.
15. alpha = 0.05                                  # 95% de confianza
16. gl = len(vals)-1                             # Grados de libertad de la distribución t de student
17.
18. lim_inf = scipy.stats.t.ppf(alpha/2, df=gl, loc=mu_cero, scale=sigma_cero)
19. lim_sup = (mu_cero-lim_inf)+mu_cero
20.
21. print("Estadístico de prueba: %.2f" % x_barra)
22. print("Puntos de corte: %.2f, %.2f" % (lim_inf,lim_sup))
23.
24. p_valor = 2*scipy.stats.t.sf(x_barra, df=gl, loc=mu_cero, scale=sigma_cero)
25. print("El p-valor es: %.2f"%p_valor)
26.
```

Al ejecutar el programa se obtiene la siguiente salida por pantalla:

Estadístico de prueba: 43.36

Puntos de corte: 27.97, 42.03

El p-valor es: 0.02