Taller de Programación
Profesor: Ing. Cristian Campos Agüero

LABORATORIO 1

El laboratorio consiste en una serie de ejercicios para practicar la resolución de problemas usando el método visto en la clase de Introducción a la Programación. Además, se introducen los conceptos de documentación de funciones y de estructuras de control.

El laboratorio es individual, deberán crear un archivo con el nombre **Lab1.py** posteriormente será subido a la plataforma de **Github Classroom** con guía del profesor. Además, por cada función que se crea agregar sus **comentarios** respectivos tal como se traba en clases

Documentación de funciones

Son **comentarios** que se insertan dentro del código, con el propósito de hacer el código fuente más fácil de entender con vistas a su mantenimiento o reutilización. Los comentarios son potencialmente significativos para los programadores, pero usualmente ignorados por los compiladores o intérpretes.

Comentarios de una línea

Se definen con **#**, pueden usarse en cualquier parte del código

Ejemplo

```
# Nombre: que calcula el área del rectángulo
# Entradas: base, altura
# Salidas: área del rectángulo
# Restricciones: base y altura mayores que cero

def areaRectangulo (base, altura):
        area = base * altura #fórmula del cálculo del área
        return area
```

Comentarios multilínea

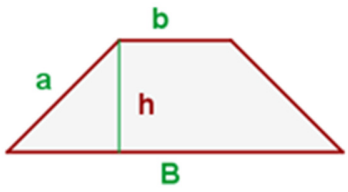
Se definen con tres comillas dobles para iniciar y cerrar

Ejemplo

```
def área_rectangulo (base, altura):
""" Este es un comentario
    de múltiples líneas
"""
        area = base * altura #fórmula del cálculo del área
        return area
```

Ejercicio #1

Hacer una función llamado **areaTrapecio** que contenga 3 parámetros de entrada (B, b, h), los tres valores deben ser **Enteros** y mayores a **CERO**.
Deben de calcular el área de un trapecio, para ellos es aplicar la siguiente fórmula:



$$A = \frac{(B + b) \cdot h}{2}$$

Donde **B** es base mayor, **b** base menor y **h** la altura

Forma de definición areaTrapecio(B, b, h)
>>>areaTrapecio(10, 5, 5)
37.5

Estructuras de Control

Cuando los problemas requieren: considerar restricciones y casos especiales, realizar acciones alternativas dependiendo de los valores de las entradas, y realizar acciones repetitivas para procesar cantidades de datos; se utilizan las estructuras de control, que pueden ser estructuras de decisión o repetitivas. Entre las estructuras de decisión más importantes se encuentra el **"if else".**

El "if else" tiene la siguiente estructura:

if condición:
    <estatutos> → se ejecutan si la condición es verdadera
[elif condición:
    <estatutos> → se ejecutan si la condición es verdadera
[else:
    <estatutos>

- La condición es una expresión que retorna un valor de verdad (True, False).
- Elif y else son opcionales. Elif puede estar cuantas veces se requiera, else sólo una vez.
- Solamente uno de los bloques de estatutos puede ser ejecutado.
- Los bloques pueden tener cualquier cantidad de estatutos.

Para obtener más información sobre el "if else" puede consultar:

- González, R. "Python para todos". Páginas 29 -31
- Marzal, A. y Gracia, I. "Introducción a la programación con Python". Páginas 81 – 107
- https://www.w3schools.com/python/gloss_python_if_statement.asp

Ejercicio #2

Implemente una función llamada **grupoPoblacion(edad)** que retorne el grupo de población que pertenece una persona según su edad.

Ejemplo:

>>>grupoPoblacion(15)
'Adolescentes'
>>>grupoPoblacion(5)
´Niños´

Tomar en cuenta que:

- El rango de edades va de **0 a 125**
- Debe ser **entero positivo**
- Si el rango de edad es de 0 a 10 su grupo de población es 'Niños'
- Si el rango de edad es mayor a 10 a 18 su grupo de población es 'Adolescentes'
- Si el rango de edad es mayor de 18 a 50 su grupo de población es 'Adultos'
- Si el rango de edad es mayor de 50 a 125 su grupo de población es 'Ancianos'
- Si es mayor a 125, retornar el mensaje "Es una edad poco probable, favor consultar"
- Si es negativo devolver el mensaje respectivo a valores entre 0 a 125

Ejercicio #3

Implemente una función llamada **sonImpares(num)** que reciba como argumentos un número entero positivo mayor a cero. La función debe retornar **True** si **todos** los dígitos que lo compone son Impares.
Ejemplos:
>>> **sonImpares (1357)**
True
>>> **sonImpares (4131)** #En este caso es falso por que el dígito 4 es par
False