Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Docente:

Ing. Coyla Idme Leonel

Alumno:

Ticona Miramira Roberto Angel

Estructuras repetitivas

» Descripción

En Python, las estructuras repetitivas támbien llamadas bucles o loops son instrucciones que permiten ejecutar un bloque de código varias veces, dependiendo de una condición o una secuencia de elementos. Los tipos de estructuras repetitivas en Python son:

- While, ejecuta un bloque de código mientras una condición sea verdadera.
- For, se utiliza para iterar una secuencia como una lista, tupla, string o rango de números.

» Ejercicio 1

Determinar el factorial de un número.

- Clase: Factorial.
- Atributo: Número.
- Acción: Calcular.
- Objetos: miFactorial = Factorial(5)

Código

```python
class Factorial:
    def __init__(self, numero):
        self.numero = numero
        self.resultado = 1

    def calcular(self):
        if self.numero < 0:
            print("El factorail no esta definido para número negativos")
            return None

        for i in range(1, self.numero + 1):
            self.resultado *= i
        return self.resultado

# Crear main

def main():
    miFactorial = Factorial(5)
    resultado = miFactorial.calcular()
    if resultado is not None:
        print(f"El factorial de {miFactorial.numero} es: {resultado}")
if __name__=="__main__":
    main()
```

Ejecución

```
El factorial de 5 es: 120
```

» Ejercicio 2

Determinar la serie de Fibonacci

- Clase: Fibonacci.
- Atributos: cantidad, serie
- Acción: generarSerie.
- Objeto: miFibonacci(10).

Código

```python
class Fibonacci:
    def __init__(self, cantidad):
        self.cantidad = cantidad
        self.serie = []

    def generarSerie(self):
        a, b = 0, 1
        for _ in range(self.cantidad):
            self.serie.append(a)
            a, b = b, a + b
        return self.serie

def main():
    cantidad = int(input("Ingrese la cantidad de la serie: "))
    miFibonacci = Fibonacci(cantidad)
    resultado = miFibonacci.generarSerie()
    print(resultado)

if __name__=="__main__":
    main()
```

Ejecución

```
Ingrese la cantidad de la serie: 10
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

» Ejercicio 3

Crear una tabla de multiplicar

- Clase: Tabla.
- Atributos: Numero.
- Acción: Multiplicación.
- Objetos: Resultado.

Código

```python
class Tabla:
    def __init__(self, numero):
        self.numero = numero

    def multiplicacion(self):
        for i in range (1, 11):
            resultado = self.numero * i
```

```
            print(f"{self.numero} X {i} = {resultado}")

def main():
    numero = int(input("Ingrese un número: "))
    resultado = Tabla(numero)
    resultado.multiplicacion()
if __name__=="__main__":
    main()
```

Ejecución

```
Ingrese un número: 8
8 X 1 = 8
8 X 2 = 16
8 X 3 = 24
8 X 4 = 32
8 X 5 = 40
8 X 6 = 48
8 X 7 = 56
8 X 8 = 64
8 X 9 = 72
8 X 10 = 80
```

» Ejercicio 4

Determinar la suma de números naturales.

- Clase: SumaNaturales.
- Atributo: limite, suma.
- Acción: calcularSuma().
- Objeto: miSuma = SumaNaturales(10).

Código

```
class SumaNaturales:
    def __init__(self, limite):
        self.limite = limite
        self.suma = 0

    def calcularSuma(self):
        for i in range (1, self.limite + 1):
            self.suma = self.suma + i
        return self.suma

def main():

    miSuma = SumaNaturales(10)
    resultado = miSuma.calcularSuma()
    print(f"La suma de los primeros {miSuma.limite} números naturales es
        ↪ {resultado}")

if __name__=="__main__":
    main()
```

Ejecución

```
La suma de los primeros 10 números naturales es 55
```

» Ejercicio 5

Determinar en una lista de números consecutivos que números son pares o impares.

- Clase: Números.
- Atributo: Número.
- Acción: Clasificar().
- Objeto: misNumeros(10).

Código

```
class Numeros:
    def __init__(self, numero):
        self.numero = numero

    def clasificar(self):
        for i in range (0, self.numero + 1):
            if i == 0:
                print("El número es 0")
            elif i % 2 == 0:
                print(f"{i} es par")
            else:
                print(f"{i} es impar")

def main():

    misNumeros = Numeros(10)
    resultado = misNumeros.clasificar()

if __name__=="__main__":
    main()
```

Ejecución

```
El número es 0
1 es impar
2 es par
3 es impar
4 es par
5 es impar
6 es par
7 es impar
8 es par
9 es impar
10 es par
```

» Ejercicio 6

Crear un programa que permita ingresar una cierta cantidad de datos y determine el promedio, la varianza y la desviación estándar de todos los datos ingresados.

- Clase: Estadistica.
- Atributo: Cantidad.
- Acción: promedio(), varianza1P(), varianza2P(), varianza1M(), varianza2M().
- Objeto: misCalculos = Estadistica(cant)

Código

```
class Estadistica:
    def __init__(self, cantidad):
```

```
        self.cantidad = cantidad
        self.datos = []

    def promedio(self):
        suma = 0
        for i in range(1, self.cantidad + 1):
            dato = float(input(f"Dato {i}: "))
            self.datos.append(dato)
            suma += dato

        return suma / self.cantidad

    def varianza1P(self):
        prom = sum(self.datos) / self.cantidad
        suma_dif = 0
        for x in self.datos:
            suma_dif += (x - prom) ** 2
        return suma_dif / self.cantidad

    def varianza2P(self):
        suma = sum(self.datos)
        suma_cuadrados = 0
        for z in self.datos:
            suma_cuadrados += z ** 2
        return (suma_cuadrados - (suma ** 2) / self.cantidad) / self.cantidad

    def varianza1M(self):
        prom = sum(self.datos) / self.cantidad
        suma_dif = 0
        for x in self.datos:
            suma_dif += (x - prom) ** 2
        return suma_dif / (self.cantidad - 1)

    def varianza2M(self):
        suma = sum(self.datos)
        suma_cuadrados = 0
        for z in self.datos:
            suma_cuadrados += z ** 2
        return (suma_cuadrados - (suma ** 2) / self.cantidad) / (self.cantidad - 1)

def main():

    cant = int(input("Ingrese la cantidad de datos: "))
    misCalculos = Estadistica(cant)
    promedio = misCalculos.promedio()

    varianza1 = misCalculos.varianza1P()
    varianza2 = misCalculos.varianza2P()
    varianza1M = misCalculos.varianza1M()
    varianza2M = misCalculos.varianza2M()

    print(f"El Promedio es: {promedio}")
    print(f"La Varianza Poblacional calculada con la formula 1 es: {varianza1}")
    print("Su Desviación Estándar es: ", varianza1 ** (1/2))
    print(f"La Varianza Poblacional calculada con la formula 2 es: {varianza2}")
    print("Su Desviación Estándar es: ", varianza2 ** (1/2))
    print(f"La Varianza Muestral calculada con la formula 1 es: {varianza1M}")
    print("Su Desviación Estándar es: ", varianza1 ** (1/2))
    print(f"La Varianza Muestral calculada con la formula 2 es: {varianza2M}")
    print("Su Desviación Estándar es: ", varianza2 ** (1/2))

if __name__=="__main__":
    main()
```

Ejecución

```
Ingrese la cantidad de datos: 10
Dato 1: 16
Dato 2: 14
Dato 3: 18
Dato 4: 15
Dato 5: 16
Dato 6: 14
Dato 7: 16
Dato 8: 18
Dato 9: 19
Dato 10: 15
El Promedio es: 16.1
La Varianza Poblacional calculada con la formula 1 es: 2.6900000000000004
Su Desviación Estándar es:  1.6401219466856727
La Varianza Poblacional calculada con la formula 2 es: 2.6900000000000093
Su Desviación Estándar es:  1.6401219466856753
La Varianza Muestral calculada con la formula 1 es: 2.988888888888889
Su Desviación Estándar es:  1.6401219466856727
La Varianza Muestral calculada con la formula 2 es: 2.988888888888899
Su Desviación Estándar es:  1.6401219466856753
```

» Ejercicio 7

Imprimir números del 1 al 10, usando la estructura while.

- Clase: NumeroNatural.
- Atributo: Valor.
- Acción: Mostrar.
- Objeto: numero = NumeroNatural.

Código

```
    class NumeroNatural:
    def __init__(self, valor):
        self.valor = valor

    def mostrar(self):
        print(f"Número natural: {self.valor}")

def main():

    i = 1
    while i <= 10:
        numero = NumeroNatural(i)
        numero.mostrar()
        i += 1

if __name__=="__main__":
    main()
```

Ejecución

```
Número natural: 1
Número natural: 2
Número natural: 3
Número natural: 4
Número natural: 5
Número natural: 6
Número natural: 7
```

```
Número natural: 8
Número natural: 9
Número natural: 10
```

» Ejercicio 8

En una lista del 1 al 10 calcular si es múltiplo de 3, 5 o ninguno.

- Clase: NumeroMultiplo.
- Atributo: Valor.
- Acción: mostrarMultiplo.
- Objeto: numero = NumeroMultiplo().

Código

```python
class NumeroMultiplo:
    def __init__(self, valor):
        self.valor = valor

    def mostrarMultiplo(self):
        if self.valor == 0:
            print(f"El {self.valor} es número nulo")
        elif self.valor % 3 == 0 and self.valor % 5 == 0:
            print(f"El {self.valor} es múltiplo de 3 y de 5")
        elif self.valor % 3 == 0:
            print(f"El {self.valor} es múltiplo de 3")
        elif self.valor % 5 == 0:
            print(f"El {self.valor} es múltiplo de 5")
        else:
            print(f"El {self.valor} no es múltiplo de 3 ni de 5")

        print("*"*60)

def main():

    i = 0
    while i <= 10:
        numero = NumeroMultiplo(i)
        numero.mostrarMultiplo()
        i += 1
if __name__=="__main__":
    main()
```

Ejecución

```
El 0 es número nulo
************************************************************
El 1 no es múltiplo de 3 ni de 5
************************************************************
El 2 no es múltiplo de 3 ni de 5
************************************************************
El 3 es múltiplo de 3
************************************************************
El 4 no es múltiplo de 3 ni de 5
************************************************************
El 5 es múltiplo de 5
************************************************************
El 6 es múltiplo de 3
************************************************************
El 7 no es múltiplo de 3 ni de 5
************************************************************
```

```
El 8 no es múltiplo de 3 ni de 5
************************************************************
El 9 es múltiplo de 3
************************************************************
El 10 es múltiplo de 5
************************************************************
```

» Ejercicio 9

Desarrolle la serie de fibonacci utilizando la sentencia while.

- Clase: Fibonacci.
- Atributo: cantidad.
- Acción: generarSerie.
- Objeto: miFibonacci = Fibonacci().

Código

```
class Fibonacci:
    def __init__(self, cantidad):
        self.cantidad = cantidad
        self.a = 0
        self.b = 1
        self.contador = 0

    def generarSerie(self):
        print(f"Serie de Fibonacci")
        while self.contador < self.cantidad:
            print(self.a, end = " ")
            c = self.a + self.b
            self.a = self.b
            self.b = c
            self.contador += 1

def main():
    cant = int(input("Ingrese una cantidad: "))
    miFibonacci = Fibonacci(cant)
    miFibonacci.generarSerie()
if __name__=="__main__":
    main()
```

Ejecución

```
Ingrese una cantidad: 10
Serie de Fibonacci
0 1 1 2 3 5 8 13 21 34
```

» Ejercicio 10

Generar números del 1 al 10 usando el ciclo while.

- Clase: Numeros.
- Atributo: cantidad.
- Acción: imprimir.
- Objeto: miObjeto = Numeros().

Código

```python
class Numeros:
    def __init__(self, cantidad):
        self.cantidad = cantidad
        self.contador = 1

    def imprimir(self):
        while self.contador <= 10:
            print(self.contador)
            self.contador += 1

def main():
    cantidad = int(input("Ingrese un número: "))
    miObjeto = Numeros(cantidad)
    miObjeto.imprimir()
if __name__=="__main__":
    main()
```

Ejecución

```
Ingrese un número: 10
1
2
3
4
5
6
7
8
9
10
```

» Ejercicio 11

Hacer una calculadora que sume números y pare cuando escribas fin.

- Clase: CalculadoraSuma.
- Atributo: total.
- Acción: sumarNumeros.
- Objeto: calculadora = CalculadoraSuma().

Código

```python
class CalculadoraSuma:
    def __init__(self):
        self.total = 0

    def sumarNumeros(self):
        print("Calcula la suma de números ingresados")
        print("Escribe números para sumar. Escribe 'fin' para terminar")
        entrada = ""
        while entrada.lower() != 'fin':
            entrada = input("Ingrese un número: ")
            if entrada.isdigit():
                self.total += int(entrada)
            elif entrada.lower() != "fin":
                print("Entrada invalida: Escriba un número o 'fin' ")
        print(f"La suma total es: {self.total}")

def main():
    calculadora = CalculadoraSuma()
```

```
    calculadora.sumarNumeros()
if __name__=="__main__":
    main()
```

Ejecución

```
Calcula la suma de números ingresados
Escribe números para sumar. Escribe 'fin' para terminar
Ingrese un número: 8
Ingrese un número: 9
Ingrese un número: 4
Ingrese un número: 5
Ingrese un número: 10
Ingrese un número: 5
Ingrese un número: fin
La suma total es: 41
```

» Ejercicio 12

Verificar si un número es nulo, par o impar; escribir 'fin' para terminar.

- Clase: Verificar.
- Atributo: pass.
- Acción: respuesta.
- Objeto: numeros = Verificar()

Código

```
class Verificar:
    def __init__(self):
        pass

    def respuesta(self):
        print("Verificar si un número es nulo, par o impar")
        print("Escribe números o 'fin' para terminar")
        entrada = ""
        while entrada.lower() != "fin":
            entrada = input("Ingrese un número: ")
            if entrada.isdigit():
                if int(entrada) == 0:
                    print("Es nulo")
                elif int(entrada) % 2 == 0:
                    print("Es par")
                else:
                    print("Es impar")
            elif entrada.lower() != "fin":
                print("Entrada invalida: Escriba un número o 'fin' ")

def main():
    numeros = Verificar()
    numeros.respuesta()
if __name__=="__main__":
    main()
```

Ejecución

```
Verificar si un número es nulo, par o impar
Escribe números o 'fin' para terminar
Ingrese un número: 0
Es nulo
```

```
Ingrese un número: 2
Es par
Ingrese un número: 3
Es impar
Ingrese un número: 5
Es impar
Ingrese un número: 4
Es par
Ingrese un número: fin
```

» Ejercicio 13

Desarrollar un gestor de tareas que tenga un menu donde se pueda escoger si agregar tareas, mostrar tareas o salir.

- Clase: GestorTareas.
- Atributo: tareas[].
- Acción: agregar_tarea, mostrar_tareas.
- Objeto: mi_gestor = GestorTareas()

Código

```python
class GestorTareas:
    def __init__(self):
        self.tareas = []

    def agregar_tarea(self, tarea):
        self.tareas.append(tarea)
        print("Tarea agregada")

    def mostrar_tareas(self):
        if not self.tareas:
            print("No hay tareas pendientes")
        else:
            print("Tareas pendientes")
            for i, tarea in enumerate(self.tareas,1):
                print(f"{i}.- {tarea}")

def main():
    mi_gestor = GestorTareas()
    while True:
        print("\n. ---- MENU ----")
        print("1.- Agregar tarea: ")
        print("2.- Mostrar tarea: ")
        print("3.- Salir")
        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            tarea = input("Escribe la tarea: ")
            mi_gestor.agregar_tarea(tarea)

        elif opcion == "2":
            mi_gestor.mostrar_tareas()

        elif opcion == "3":
            print("Saliendo del gestor de tareas")
            break
        else:
            print("Opción no valida. Intenta de nuevo")
if __name__=="__main__":
    main()
```

Ejecución

```
. ---- MENU ----
1.- Agregar tarea:
2.- Mostrar tarea:
3.- Salir
Seleccione una opción: 1
Escribe la tarea: Clases de programación
Tarea agregada

. ---- MENU ----
1.- Agregar tarea:
2.- Mostrar tarea:
3.- Salir
Seleccione una opción: 1
Escribe la tarea: Almorzar
Tarea agregada

. ---- MENU ----
1.- Agregar tarea:
2.- Mostrar tarea:
3.- Salir
Seleccione una opción: 1
Escribe la tarea: Realizar tareas
Tarea agregada

. ---- MENU ----
1.- Agregar tarea:
2.- Mostrar tarea:
3.- Salir
Seleccione una opción: 2
Tareas pendientes
1.- Clases de programación
2.- Almorzar
3.- Realizar tareas

. ---- MENU ----
1.- Agregar tarea:
2.- Mostrar tarea:
3.- Salir
Seleccione una opción: 3
Saliendo del gestor de tareas
```