

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 también 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

```
1 class Factorial:
2     def __init__(self, numero):
3         self.numero = numero
4         self.resultado = 1
5
6     def calcular(self):
7         if self.numero < 0:
8             print("El factorail no esta definido para número negativos")
9             return None
10
11         for i in range(1, self.numero + 1):
12             self.resultado *= i
13         return self.resultado
14
15 # Crear main
16
17 def main():
18     miFactorial = Factorial(5)
19     resultado = miFactorial.calcular()
20     if resultado is not None:
21         print(f"El factorial de {miFactorial.numero} es: {resultado}")
22 if __name__ == "__main__":
23     main()
```

Ejecución

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

» EJERCICIO 2

Determinar la serie de Fibonacci

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

Código

```
1 class Fibonacci:
2     def __init__(self, cantidad):
3         self.cantidad = cantidad
4         self.serie = []
5
6     def generarSerie(self):
7         a, b = 0, 1
8         for _ in range(self.cantidad):
9             self.serie.append(a)
10            a, b = b, a + b
11        return self.serie
12
13 def main():
14     cantidad = int(input("Ingrese la cantidad de la serie: "))
15     miFibonacci = Fibonacci(cantidad)
16     resultado = miFibonacci.generarSerie()
17     print(resultado)
18
19 if __name__ == "__main__":
20     main()
```

Ejecución

```
1 Ingrese la cantidad de la serie: 10
2 [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

```
1 class Tabla:
2     def __init__(self, numero):
3         self.numero = numero
4
5     def multiplicacion(self):
6         for i in range(1, 11):
7             resultado = self.numero * i
```

```
8         print(f"{self.numero} X {i} = {resultado}")
9
10 def main():
11     numero = int(input("Ingrese un número: "))
12     resultado = Tabla(numero)
13     resultado.multiplicacion()
14 if __name__ == "__main__":
15     main()
```

Ejecución

```
1 Ingrese un número: 8
2 8 X 1 = 8
3 8 X 2 = 16
4 8 X 3 = 24
5 8 X 4 = 32
6 8 X 5 = 40
7 8 X 6 = 48
8 8 X 7 = 56
9 8 X 8 = 64
10 8 X 9 = 72
11 8 X 10 = 80
```

» EJERCICIO 4

Determinar la suma de números naturales.

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

Código

```
1 class SumaNaturales:
2     def __init__(self, limite):
3         self.limite = limite
4         self.suma = 0
5
6     def calcularSuma(self):
7         for i in range(1, self.limite + 1):
8             self.suma = self.suma + i
9         return self.suma
10
11 def main():
12
13     miSuma = SumaNaturales(10)
14     resultado = miSuma.calcularSuma()
15     print(f"La suma de los primeros {miSuma.limite} números naturales es
16           ↳ {resultado}")
17
18 if __name__ == "__main__":
19     main()
```

Ejecución

```
1 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

```
1 class Numeros:
2     def __init__(self, numero):
3         self.numero = numero
4
5     def clasificar(self):
6         for i in range (0, self.numero + 1):
7             if i == 0:
8                 print("El número es 0")
9             elif i % 2 == 0:
10                print(f"{i} es par")
11            else:
12                print(f"{i} es impar")
13
14 def main():
15
16     misNumeros = Numeros(10)
17     resultado = misNumeros.clasificar()
18
19 if __name__=="__main__":
20     main()
```

Ejecución

```
1 El número es 0
2 1 es impar
3 2 es par
4 3 es impar
5 4 es par
6 5 es impar
7 6 es par
8 7 es impar
9 8 es par
10 9 es impar
11 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: Estadística.
- Atributo: Cantidad.
- Acción: promedio(), varianza1P(), varianza2P(), varianza1M(), varianza2M().
- Objeto: misCalculos = Estadistica(cant)

Código

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

```
3         self.cantidad = cantidad
4         self.datos = []
5
6     def promedio(self):
7         suma = 0
8         for i in range(1, self.cantidad + 1):
9             dato = float(input(f"Dato {i}: "))
10            self.datos.append(dato)
11            suma += dato
12
13        return suma / self.cantidad
14
15    def varianza1P(self):
16        prom = sum(self.datos) / self.cantidad
17        suma_dif = 0
18        for x in self.datos:
19            suma_dif += (x - prom) ** 2
20        return suma_dif / self.cantidad
21
22    def varianza2P(self):
23        suma = sum(self.datos)
24        suma_cuadrados = 0
25        for z in self.datos:
26            suma_cuadrados += z ** 2
27        return (suma_cuadrados - (suma ** 2) / self.cantidad) / self.cantidad
28
29    def varianza1M(self):
30        prom = sum(self.datos) / self.cantidad
31        suma_dif = 0
32        for x in self.datos:
33            suma_dif += (x - prom) ** 2
34        return suma_dif / (self.cantidad - 1)
35
36    def varianza2M(self):
37        suma = sum(self.datos)
38        suma_cuadrados = 0
39        for z in self.datos:
40            suma_cuadrados += z ** 2
41        return (suma_cuadrados - (suma ** 2) / self.cantidad) / (self.cantidad - 1)
42
43    def main():
44
45        cant = int(input("Ingrese la cantidad de datos: "))
46        misCalculos = Estadistica(cant)
47        promedio = misCalculos.promedio()
48
49        varianza1 = misCalculos.varianza1P()
50        varianza2 = misCalculos.varianza2P()
51        varianza1M = misCalculos.varianza1M()
52        varianza2M = misCalculos.varianza2M()
53
54        print(f"El Promedio es: {promedio}")
55        print(f"La Varianza Poblacional calculada con la formula 1 es: {varianza1}")
56        print("Su Desviación Estándar es: ", varianza1 ** (1/2))
57        print(f"La Varianza Poblacional calculada con la formula 2 es: {varianza2}")
58        print("Su Desviación Estándar es: ", varianza2 ** (1/2))
59        print(f"La Varianza Muestral calculada con la formula 1 es: {varianza1M}")
60        print("Su Desviación Estándar es: ", varianza1M ** (1/2))
61        print(f"La Varianza Muestral calculada con la formula 2 es: {varianza2M}")
62        print("Su Desviación Estándar es: ", varianza2M ** (1/2))
63
64    if __name__ == "__main__":
65        main()
```

Ejecución

```
1 Ingrese la cantidad de datos: 10
2 Dato 1: 16
3 Dato 2: 14
4 Dato 3: 18
5 Dato 4: 15
6 Dato 5: 16
7 Dato 6: 14
8 Dato 7: 16
9 Dato 8: 18
10 Dato 9: 19
11 Dato 10: 15
12 El Promedio es: 16.1
13 La Varianza Poblacional calculada con la formula 1 es: 2.6900000000000004
14 Su Desviación Estándar es: 1.6401219466856727
15 La Varianza Poblacional calculada con la formula 2 es: 2.6900000000000093
16 Su Desviación Estándar es: 1.6401219466856753
17 La Varianza Muestral calculada con la formula 1 es: 2.988888888888889
18 Su Desviación Estándar es: 1.6401219466856727
19 La Varianza Muestral calculada con la formula 2 es: 2.988888888888899
20 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

```
1 class NumeroNatural:
2     def __init__(self, valor):
3         self.valor = valor
4
5     def mostrar(self):
6         print(f"Número natural: {self.valor}")
7
8 def main():
9
10     i = 1
11     while i <= 10:
12         numero = NumeroNatural(i)
13         numero.mostrar()
14         i += 1
15
16 if __name__ == "__main__":
17     main()
```

Ejecución

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

```

8 Número natural: 8
9 Número natural: 9
10 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

```

1 class NumeroMultiplo:
2     def __init__(self, valor):
3         self.valor = valor
4
5     def mostrarMultiplo(self):
6         if self.valor == 0:
7             print(f"El {self.valor} es número nulo")
8         elif self.valor % 3 == 0 and self.valor % 5 == 0:
9             print(f"El {self.valor} es múltiplo de 3 y de 5")
10        elif self.valor % 3 == 0:
11            print(f"El {self.valor} es múltiplo de 3")
12        elif self.valor % 5 == 0:
13            print(f"El {self.valor} es múltiplo de 5")
14        else:
15            print(f"El {self.valor} no es múltiplo de 3 ni de 5")
16
17        print("*"*60)
18
19 def main():
20
21     i = 0
22     while i <= 10:
23         numero = NumeroMultiplo(i)
24         numero.mostrarMultiplo()
25         i += 1
26 if __name__ == "__main__":
27     main()

```

Ejecución

```

1 El 0 es número nulo
2 *****
3 El 1 no es múltiplo de 3 ni de 5
4 *****
5 El 2 no es múltiplo de 3 ni de 5
6 *****
7 El 3 es múltiplo de 3
8 *****
9 El 4 no es múltiplo de 3 ni de 5
10 *****
11 El 5 es múltiplo de 5
12 *****
13 El 6 es múltiplo de 3
14 *****
15 El 7 no es múltiplo de 3 ni de 5
16 *****

```

```
17 El 8 no es múltiplo de 3 ni de 5
18 *****
19 El 9 es múltiplo de 3
20 *****
21 El 10 es múltiplo de 5
22 *****
```

» EJERCICIO 9

Desarrolle la serie de fibonacci utilizando la sentencia while.

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

Código

```
1 class Fibonacci:
2     def __init__(self, cantidad):
3         self.cantidad = cantidad
4         self.a = 0
5         self.b = 1
6         self.contador = 0
7
8     def generarSerie(self):
9         print(f"Serie de Fibonacci")
10        while self.contador < self.cantidad:
11            print(self.a, end = " ")
12            c = self.a + self.b
13            self.a = self.b
14            self.b = c
15            self.contador += 1
16
17 def main():
18     cant = int(input("Ingrese una cantidad: "))
19     miFibonacci = Fibonacci(cant)
20     miFibonacci.generarSerie()
21 if __name__ == "__main__":
22     main()
```

Ejecución

```
1 Ingrese una cantidad: 10
2 Serie de Fibonacci
3 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


```
1 class Numeros:
2     def __init__(self, cantidad):
3         self.cantidad = cantidad
4         self.contador = 1
5
6     def imprimir(self):
7         while self.contador <= 10:
8             print(self.contador)
9             self.contador += 1
10
11 def main():
12     cantidad = int(input("Ingrese un número: "))
13     miObjeto = Numeros(cantidad)
14     miObjeto.imprimir()
15 if __name__ == "__main__":
16     main()
```

Ejecución

```
1 Ingrese un número: 10
2 1
3 2
4 3
5 4
6 5
7 6
8 7
9 8
10 9
11 10
```

» EJERCICIO 11

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

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

Código

```
1 class CalculadoraSuma:
2     def __init__(self):
3         self.total = 0
4
5     def sumarNumeros(self):
6         print("Calcula la suma de números ingresados")
7         print("Escribe números para sumar. Escribe 'fin' para terminar")
8         entrada = ""
9         while entrada.lower() != 'fin':
10             entrada = input("Ingrese un número: ")
11             if entrada.isdigit():
12                 self.total += int(entrada)
13             elif entrada.lower() != "fin":
14                 print("Entrada invalida: Escriba un número o 'fin' ")
15             print(f"La suma total es: {self.total}")
16
17 def main():
18     calculadora = CalculadoraSuma()
```

```

19     calculadora.sumarNumeros()
20 if __name__=="__main__":
21     main()

```

Ejecución

```

1 Calcula la suma de números ingresados
2 Escribe números para sumar. Escribe 'fin' para terminar
3 Ingrese un número: 8
4 Ingrese un número: 9
5 Ingrese un número: 4
6 Ingrese un número: 5
7 Ingrese un número: 10
8 Ingrese un número: 5
9 Ingrese un número: fin
10 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

```

1 class Verificar:
2     def __init__(self):
3         pass
4
5     def respuesta(self):
6         print("Verificar si un número es nulo, par o impar")
7         print("Escribe números o 'fin' para terminar")
8         entrada = ""
9         while entrada.lower() != "fin":
10             entrada = input("Ingrese un número: ")
11             if entrada.isdigit():
12                 if int(entrada) == 0:
13                     print("Es nulo")
14                 elif int(entrada) % 2 == 0:
15                     print("Es par")
16                 else:
17                     print("Es impar")
18             elif entrada.lower() != "fin":
19                 print("Entrada invalida: Escriba un número o 'fin' ")
20
21 def main():
22     numeros = Verificar()
23     numeros.respuesta()
24 if __name__=="__main__":
25     main()

```

Ejecución

```

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

```

```
5 Ingrese un número: 2
6 Es par
7 Ingrese un número: 3
8 Es impar
9 Ingrese un número: 5
10 Es impar
11 Ingrese un número: 4
12 Es par
13 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

```
1 class GestorTareas:
2     def __init__(self):
3         self.tareas = []
4
5     def agregar_tarea(self, tarea):
6         self.tareas.append(tarea)
7         print("Tarea agregada")
8
9     def mostrar_tareas(self):
10        if not self.tareas:
11            print("No hay tareas pendientes")
12        else:
13            print("Tareas pendientes")
14            for i, tarea in enumerate(self.tareas,1):
15                print(f"{i}.- {tarea}")
16
17 def main():
18     mi_gestor = GestorTareas()
19     while True:
20         print("\n. ---- MENU ----")
21         print("1.- Agregar tarea: ")
22         print("2.- Mostrar tarea: ")
23         print("3.- Salir")
24         opcion = input("Seleccione una opción: ")
25
26         if opcion == "1":
27             tarea = input("Escribe la tarea: ")
28             mi_gestor.agregar_tarea(tarea)
29
30         elif opcion == "2":
31             mi_gestor.mostrar_tareas()
32
33         elif opcion == "3":
34             print("Saliendo del gestor de tareas")
35             break
36         else:
37             print("Opción no valida. Intenta de nuevo")
38 if __name__ == "__main__":
39     main()
```

Ejecución

```
1 . ---- MENU ----
2 1.- Agregar tarea:
3 2.- Mostrar tarea:
4 3.- Salir
5 Seleccione una opción: 1
6 Escribe la tarea: Clases de programación
7 Tarea agregada
8
9 . ---- MENU ----
10 1.- Agregar tarea:
11 2.- Mostrar tarea:
12 3.- Salir
13 Seleccione una opción: 1
14 Escribe la tarea: Almorzar
15 Tarea agregada
16
17 . ---- MENU ----
18 1.- Agregar tarea:
19 2.- Mostrar tarea:
20 3.- Salir
21 Seleccione una opción: 1
22 Escribe la tarea: Realizar tareas
23 Tarea agregada
24
25 . ---- MENU ----
26 1.- Agregar tarea:
27 2.- Mostrar tarea:
28 3.- Salir
29 Seleccione una opción: 2
30 Tareas pendientes
31 1.- Clases de programación
32 2.- Almorzar
33 3.- Realizar tareas
34
35 . ---- MENU ----
36 1.- Agregar tarea:
37 2.- Mostrar tarea:
38 3.- Salir
39 Seleccione una opción: 3
40 Saliendo del gestor de tareas
```

» EJERCICIO 14

Verificar si los números ingresados están dentro del rango de 0 a 10.

- Clase: Comprobar.
- Atributos: pass.
- Acción: rpta().
- Objeto: numeros = Comprobar().

Código

```
1 class Comprobar:
2     def __init__(self):
3         pass
4
5     def rpta(self):
```

```
6         print("Verificar números")
7         entrada = int(input("Ingrese un número: "))
8         while entrada > 0 and entrada <= 10:
9             print(f"El número {entrada} está en el rango (1-10)")
10            entrada = int(input("Ingrese otro número (fuera de 1-10) para salir: "))
11
12 def main():
13     numeros = Comprobar()
14     numeros.rpta()
15
16 if __name__=="__main__":
17     main()
```

Ejecución

```
1 Verificar números
2 Ingrese un número: 5
3 El número 5 está en el rango (1-10)
4 Ingrese otro número (fuera de 1-10) para salir: 1
5 El número 1 está en el rango (1-10)
6 Ingrese otro número (fuera de 1-10) para salir: 8
7 El número 8 está en el rango (1-10)
8 Ingrese otro número (fuera de 1-10) para salir: 11
```