

Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Docente:

Ing. Coyla Idme Leonel

Alumno:

Ticona Miramira Roberto Angel

Estructuras selectivas

» DESCRIPCIÓN

En el contexto de programación orientado a objetos con Python, las estructuras selectivas son aquellas que permiten tomar decisiones durante la ejecución del programa, es decir, ejecutar diferentes bloques de código según se cumpla o no una condición. Aunque las estructuras selectivas no son exclusivas de la POO, también se usan en programación estructurada, en un programa orientado a objetos se usan dentro de métodos o funciones que pertenecen a clases para controlar el flujo de ejecución.

» EJERCICIO 1

Identificar si un número es nulo, par o impar.

- Clase: Número.
- Atributo: Valor.
- Acción: Clasificar el número.
- Objetos: Numero(0), Número(2), Número(5)

Código

```
1 class Numero:
2     def __init__(self, valor):
3         self.valor = valor
4
5     def clasificar(self):
6         if self.valor == 0:
7             return "Nulo"
8         elif self.valor % 2 == 0:
9             return "Par"
10        else:
11            return "Impar"
12
13 ejemplos = [Numero(0), Numero(2), Numero(5)]
14
15 for num in ejemplos:
16     tipo = num.clasificar()
17     print(f"El número {num.valor} es {tipo}")
```

Ejecución

```
1 El número 0 es Nulo
2 El número 2 es Par
3 El número 5 es Impar
```

» EJERCICIO 2

Determinar si una persona es mayor de edad.

- Clase: Persona.
- Atributos: Nombre, Edad.
- Comportamiento: Es_mayor_de_edad().
- Objeto: Persona("Maria", 25).

Código

```
1 class Persona:
2     def __init__(self, Nombre, Edad):
3         self.Nombre = Nombre
4         self.Edad = Edad
5
6     def Es_mayor_de_edad(self):
7         if self.Edad > 18:
8             return "Es mayor de edad"
9         else:
10            return "No es mayor de edad"
11
12 ejemplo = Persona("Maria", 25)
13 res = ejemplo.Es_mayor_de_edad()
14
15 print(ejemplo.Nombre, res)
```

Ejecución

```
1 Maria Es mayor de edad
```

» EJERCICIO 3

Determinar el aumento de un sueldo de un trabajador dependiendo del cargo.

- Clase: Empleado.
- Atributos: Nombre, Cargo, Salario.
- AplicarAumento():
 - Gerente = 10 %
 - Supervisor = 7 %
 - Operario = 5 %
- Objetos:
 - Empleado("Carlos", "Gerente", 2000)
 - Empleado("Maria", "Supervisor", 2000)
 - Empleado("Ana", "Interna", 800)
 - Empleado("Roberto", "Operario", 1600)

Código

```
1 class Empleado:
2     def __init__(self, nombre, cargo, salario):
3         self.nombre = nombre
4         self.cargo = cargo
```

```

5         self.salario = salario
6
7     def AplicarAumento(self):
8         if self.cargo == "Gerente":
9             porcentaje = 0.10
10        elif self.cargo == "Supervisor":
11            porcentaje = 0.07
12        elif self.cargo == "Operario":
13            porcentaje = 0.05
14        else:
15            porcentaje = 0.0
16
17        nuevoSalario = self.salario * (1 + porcentaje)
18        return nuevoSalario
19
20 empleado1 = Empleado("Carlos", "Gerente", 2000)
21 empleado2 = Empleado("Maria", "Supervisor", 2000)
22 empleado3 = Empleado("Ana", "Interna", 800)
23 empleado4 = Empleado("Roberto", "Operario", 1600)
24
25 for emp in (empleado1, empleado2, empleado3, empleado4):
26     nuevo = emp.AplicarAumento()
27     print(f"{emp.nombre} {emp.cargo}: salario nuevo {nuevo:2f}")

```

Ejecución

```

1 Carlos Gerente: salario nuevo 2200.000000
2 Maria Supervisor: salario nuevo 2140.000000
3 Ana Interna: salario nuevo 800.000000
4 Roberto Operario: salario nuevo 1680.000000

```

» EJERCICIO 4

Diseñar un código que según el día y el mes indique el signo zodiacal.

Clase: Signo.

Atributos: Mes, Día.

Comportamiento: Determinar signo zodiacal.

Objeto: Signo().

Código

```

1 import tkinter as tk
2 from tkinter import ttk, messagebox
3
4 class Signo:
5     def __init__(self, mes, dia):
6         self.mes = mes
7         self.dia = dia
8
9     def zodiaco(self):
10        if (self.mes == "Marzo" and self.dia >= 21 and self.dia <= 31) or (self.mes
11            ↳ == "Abril" and self.dia <= 19 and self.dia > 0):
12            return " Aries"
13        elif (self.mes == "Abril" and self.dia >= 20 and self.dia <= 30) or
14            ↳ (self.mes == "Mayo" and self.dia <= 20 and self.dia > 0):
15            return " Tauro"
16        elif (self.mes == "Mayo" and self.dia >= 21 and self.dia <= 31) or
17            ↳ (self.mes == "Junio" and self.dia <= 20 and self.dia > 0):
18            return " Géminis"

```

```

16     elif (self.mes == "Junio" and self.dia >= 21 and self.dia <= 30) or
17         ↪ (self.mes == "Julio" and self.dia <= 22 and self.dia > 0):
18         return " Cáncer"
19     elif (self.mes == "Julio" and self.dia >= 23 and self.dia <= 31) or
20         ↪ (self.mes == "Agosto" and self.dia <= 22 and self.dia > 0):
21         return " Leo"
22     elif (self.mes == "Agosto" and self.dia >= 23 and self.dia <= 31) or
23         ↪ (self.mes == "Septiembre" and self.dia <= 22 and self.dia > 0):
24         return " Virgo"
25     elif (self.mes == "Septiembre" and self.dia >= 23 and self.dia <= 30) or
26         ↪ (self.mes == "Octubre" and self.dia <= 22 and self.dia > 0):
27         return " Libra"
28     elif (self.mes == "Octubre" and self.dia >= 23 and self.dia <= 31) or
29         ↪ (self.mes == "Noviembre" and self.dia <= 21 and self.dia > 0):
30         return " Escorpio"
31     elif (self.mes == "Noviembre" and self.dia >= 22 and self.dia <= 30) or
32         ↪ (self.mes == "Diciembre" and self.dia <= 21 and self.dia > 0):
33         return " Sagitario"
34     elif (self.mes == "Diciembre" and self.dia >= 22 and self.dia <= 31) or
35         ↪ (self.mes == "Enero" and self.dia <= 19 and self.dia > 0):
36         return " Capricornio"
37     elif (self.mes == "Enero" and self.dia >= 20 and self.dia <= 31) or
38         ↪ (self.mes == "Febrero" and self.dia <= 18 and self.dia > 0):
39         return " Acuario"
40     elif (self.mes == "Febrero" and self.dia >= 19 and self.dia <= 29) or
41         ↪ (self.mes == "Marzo" and self.dia <= 20 and self.dia > 0):
42         return " Piscis"
43     else:
44         return "Día invalido"
45
46 # ----- Tkinter UI ----- #
47
48 def calcular_signo():
49     mes = combo_mes.get()
50     try:
51         dia = int(entry_dia.get())
52     except ValueError:
53         messagebox.showerror("Error", "Ingrese un día válido (número entero).")
54         return
55
56     if not mes:
57         messagebox.showerror("Error", "Seleccione un mes de nacimiento.")
58         return
59
60     signo = Signo(mes, dia).zodiaco()
61     label_resultado.config(text=f"Tu signo es:\n{signo}", foreground="#222",
62                             ↪ font=("Arial", 18, "bold"))
63
64 # Ventana principal
65 ventana = tk.Tk()
66 ventana.title(" Calculadora de Signo Zodiacal")
67 ventana.geometry("450x400")
68 ventana.config(bg="#f0f4f7")
69
70 # Estilo moderno
71 style = ttk.Style()
72 style.configure("TButton", font=("Arial", 12), padding=6)
73 style.configure("TLabel", font=("Arial", 12))
74
75 # Título
76 titulo = tk.Label(ventana, text=" Descubre tu Signo Zodiacal ",
77                     bg="#f0f4f7", fg="#444", font=("Arial", 16, "bold"))

```

```
69 titulo.pack(pady=15)
70
71 # Frame para inputs
72 frame = tk.Frame(ventana, bg="#f0f4f7")
73 frame.pack(pady=10)
74
75 # Mes
76 ttk.Label(frame, text="Mes: ").grid(row=0, column=0, padx=5, pady=5, sticky="e")
77 meses = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
78          "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"]
79 combo_mes = ttk.Combobox(frame, values=meses, state="readonly", font=("Arial", 12))
80 combo_mes.grid(row=0, column=1, padx=5, pady=5)
81
82 # Día
83 ttk.Label(frame, text="Día: ").grid(row=1, column=0, padx=5, pady=5, sticky="e")
84 entry_dia = ttk.Entry(frame, font=("Arial", 12))
85 entry_dia.grid(row=1, column=1, padx=5, pady=5)
86
87 # Botón
88 btn_calcular = ttk.Button(ventana, text="  Calcular Signo", command=calcular_signo)
89 btn_calcular.pack(pady=15)
90
91 # Resultado
92 label_resultado = tk.Label(ventana, text="", bg="#f0f4f7", fg="#333",
93                             ↪ font=("Arial", 14))
94 label_resultado.pack(pady=20)
95
96 # Loop
97 ventana.mainloop()
```

Ejecución

