Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Docente:

Ing. Coyla Idme Leonel

Alumno:

Ticona Miramira Roberto Angel

Introducción a la Programación Orientada a Objetos

» 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

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

    def clasificar(self):
        if self.valor == 0:
            return "Nulo"
        elif self.valor % 2 == 0:
            return "Par"
        else:
            return "Impar"

ejemplos = [Numero(0), Numero(2), Numero(5)]

for num in ejemplos:
    tipo = num.clasificar()
    print(f"El número {num.valor} es {tipo}")
```

Ejecución

```
El número 0 es Nulo
El número 2 es Par
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

```
class Persona:
    def __init__(self, Nombre, Edad):
        self.Nombre = Nombre
        self.Edad = Edad

    def Es_mayor_de_edad(self):
        if self.Edad > 18:
            return "Es mayor de edad"
        else:
            return "No es mayor de edad"

ejemplo = Persona("Maria", 25)
res = ejemplo.Es_mayor_de_edad()

print(ejemplo.Nombre, res)
```

Ejecución

```
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

```
class Empleado:
    def __init__(self, nombre, cargo, salario):
        self.nombre = nombre
        self.cargo = cargo
```

```
        self.salario = salario

    def AplicarAumento(self):
        if self.cargo == "Gerente":
            porcentaje = 0.10
        elif self.cargo == "Supervisor":
            porcentaje = 0.07
        elif self.cargo == "Operario":
            porcentaje = 0.05
        else:
            porcentaje = 0.0

        nuevoSalario = self.salario * (1 + porcentaje)
        return nuevoSalario

empleado1 = Empleado("Carlos", "Gerente", 2000)
empleado2 = Empleado("Maria", "Supervisor", 2000)
empleado3 = Empleado("Ana", "Interna", 800)
empleado4 = Empleado("Roberto", "Operario", 1600)

for emp in (empleado1, empleado2, empleado3, empleado4):
    nuevo = emp.AplicarAumento()
    print(f"{emp.nombre} {emp.cargo}: salario nuevo {nuevo:2f}")
```

Ejecución

```
Carlos Gerente: salario nuevo 2200.000000
Maria Supervisor: salario nuevo 2140.000000
Ana Interna: salario nuevo 800.000000
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

```
import tkinter as tk
from tkinter import ttk, messagebox

class Signo:
    def __init__(self, mes, dia):
        self.mes = mes
        self.dia = dia

    def zodiaco(self):
        if (self.mes == "Marzo" and self.dia >= 21 and self.dia <= 31) or (self.mes
            ↪ == "Abril" and self.dia <= 19 and self.dia > 0):
            return "  Aries"
        elif (self.mes == "Abril" and self.dia >= 20 and self.dia <= 30) or
            ↪ (self.mes == "Mayo" and self.dia <= 20 and self.dia > 0):
            return "  Tauro"
        elif (self.mes == "Mayo" and self.dia >= 21 and self.dia <= 31) or
            ↪ (self.mes == "Junio" and self.dia <= 20 and self.dia > 0):
            return "  Géminis"
```

```
        elif (self.mes == "Junio" and self.dia >= 21 and self.dia <= 30) or (self.mes == "Julio" and self.dia <= 22 and self.dia > 0):
            return "  Cáncer"
        elif (self.mes == "Julio" and self.dia >= 23 and self.dia <= 31) or (self.mes == "Agosto" and self.dia <= 22 and self.dia > 0):
            return "  Leo"
        elif (self.mes == "Agosto" and self.dia >= 23 and self.dia <= 31) or (self.mes == "Septiembre" and self.dia <= 22 and self.dia > 0):
            return "  Virgo"
        elif (self.mes == "Septiembre" and self.dia >= 23 and self.dia <= 30) or (self.mes == "Octubre" and self.dia <= 22 and self.dia > 0):
            return "  Libra"
        elif (self.mes == "Octubre" and self.dia >= 23 and self.dia <= 31) or (self.mes == "Noviembre" and self.dia <= 21 and self.dia > 0):
            return "  Escorpio"
        elif (self.mes == "Noviembre" and self.dia >= 22 and self.dia <= 30) or (self.mes == "Diciembre" and self.dia <= 21 and self.dia > 0):
            return "  Sagitario"
        elif (self.mes == "Diciembre" and self.dia >= 22 and self.dia <= 31) or (self.mes == "Enero" and self.dia <= 19 and self.dia > 0):
            return "  Capricornio"
        elif (self.mes == "Enero" and self.dia >= 20 and self.dia <= 31) or (self.mes == "Febrero" and self.dia <= 18 and self.dia > 0):
            return "  Acuario"
        elif (self.mes == "Febrero" and self.dia >= 19 and self.dia <= 29) or (self.mes == "Marzo" and self.dia <= 20 and self.dia > 0):
            return "  Piscis"
        else:
            return "Día invalido"


# ---------------- Tkinter UI ---------------- #

def calcular_signo():
    mes = combo_mes.get()
    try:
        dia = int(entry_dia.get())
    except ValueError:
        messagebox.showerror("Error", "Ingrese un día válido (número entero).")
        return

    if not mes:
        messagebox.showerror("Error", "Seleccione un mes de nacimiento.")
        return

    signo = Signo(mes, dia).zodiaco()
    label_resultado.config(text=f"Tu signo es:\n{signo}", foreground="#222", font=("Arial", 18, "bold"))

# Ventana principal
ventana = tk.Tk()
ventana.title("  Calculadora de Signo Zodiacal")
ventana.geometry("450x400")
ventana.config(bg="#f0f4f7")

# Estilo moderno
style = ttk.Style()
style.configure("TButton", font=("Arial", 12), padding=6)
style.configure("TLabel", font=("Arial", 12))

# Título
titulo = tk.Label(ventana, text="  Descubre tu Signo Zodiacal  ",
                  bg="#f0f4f7", fg="#444", font=("Arial", 16, "bold"))
```

```
titulo.pack(pady=15)

# Frame para inputs
frame = tk.Frame(ventana, bg="#f0f4f7")
frame.pack(pady=10)

# Mes
ttk.Label(frame, text="Mes: ").grid(row=0, column=0, padx=5, pady=5, sticky="e")
meses = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
         "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"]
combo_mes = ttk.Combobox(frame, values=meses, state="readonly", font=("Arial", 12))
combo_mes.grid(row=0, column=1, padx=5, pady=5)

# Día
ttk.Label(frame, text="Día: ").grid(row=1, column=0, padx=5, pady=5, sticky="e")
entry_dia = ttk.Entry(frame, font=("Arial", 12))
entry_dia.grid(row=1, column=1, padx=5, pady=5)

# Botón
btn_calcular = ttk.Button(ventana, text="  Calcular Signo", command=calcular_signo)
btn_calcular.pack(pady=15)

# Resultado
label_resultado = tk.Label(ventana, text="", bg="#f0f4f7", fg="#333",
    ↪ font=("Arial", 14))
label_resultado.pack(pady=20)

# Loop
ventana.mainloop()
```

Ejecución