Guía Práctica N.º 7: Estructuras de Selección Múltiple (Match-Case) en Python

Resultado De Aprendizaje:

- Conocer estructura selectiva match case para resolver situaciones propuestas en Python modo consola.
- Desarrollar ejercicios propuestos con la estructura selectiva match case.

Sentencia match-case

La sentencia **match-case** en Python, introducida en la versión 3.10, es una funcionalidad que permite controlar el flujo de los programas de manera más eficiente. Esta sentencia se utiliza para ejecutar ciertas partes del código si se cumplen determinadas condiciones o casos.

La estructura de **match-case** es similar a la de **switch-case** en otros lenguajes de programación, pero en Python es más potente y permite realizar coincidencias de patrones más complejas.

Ejemplo:

```python
comida_preferida = 'Pizza'
match comida_preferida:
    case 'Tacos':
        print('Te gustan los Tacos')
    case 'Pizza':
        print('Te gusta la Pizza')
    case 'Hamburguesa':
        print('Te gustan las hamburguesas')
    case _:
        print('Te gusta otra comida')
```

En este ejemplo, se definió una variable **comida_preferida** y se usa la palabra clave **match** para hacer coincidir su valor con los casos definidos después de cada palabra clave **case**. Si comida_preferida es 'Tacos', se imprimirá 'Te gustan los Tacos'. Si es 'Pizza', se imprimirá 'Te gusta la Pizza', y así sucesivamente. Si comida_preferida no coincide con ninguno de los casos especificados, se ejecutará el bloque de código después de **case _**, que es un comodín que coincide con cualquier valor.

Una de las novedades más esperadas (y quizás controvertidas) de Python 3.10 fue el llamado **Structural Pattern Matching** que introdujo en el lenguaje una nueva sentencia condicional. Ésta se podría asemejar a la sentencia **«switch»** que ya existe en otros lenguajes de programación.

Comparando valores

En su versión más simple, el **«pattern matching»** permite comparar un valor de entrada con una serie de literales. Algo así como un conjunto de sentencias **«if»** encadenadas.

Ejemplo:

¿Qué ocurre si el valor que se compara no existe entre las opciones disponibles? Pues en principio, nada, ya que este caso no está cubierto. Si se quiere controlar, hay que añadir una nueva regla utilizando el **subguión _** como patrón:

Patrones avanzados

La sentencia **match-case** va mucho más allá de una simple comparación de valores. Con ella podremos deconstruir estructuras de datos, capturar elementos o mapear valores.

Para ejemplificar varias de sus funcionalidades, vamos a partir de una tupla que representará un punto en el plano (2 coordenadas) o en el espacio (3 coordenadas). Lo primero que vamos a hacer es detectar en qué dimensión se encuentra el punto:

```python
point = (2, 5)

match point:
    case (x, y):
        print(f'({x},{y}) is in plane')
    case (x, y, z):
        print(f'({x},{y},{z}) is in space')
```

```python
point = (3, 1, 7)

match point:
    case (x, y):
        print(f'({x},{y}) is in plane')
    case (x, y, z):
        print(f'({x},{y},{z}) is in space')
```

En cualquier caso, esta aproximación permitiría un punto formado por **«strings»**:

```python
point = ('2', '5')

match point:
    case (x, y):
        print(f'({x},{y}) is in plane')
    case (x, y, z):
        print(f'({x},{y},{z}) is in space')
```

Por lo tanto, en un siguiente paso, se puede restringir los patrones a valores enteros:

```python
point = ('2', '5')

match point:
    case (int(), int()):
        print(f'{point} is in plane')
    case (int(), int(), int()):
        print(f'{point} is in space')
    case _:
        print('Unknown!')

point = (3, 9, 1)
match point:
    case (int(), int()):
        print(f'{point} is in plane')
    case (int(), int(), int()):
        print(f'{point} is in space')
    case _:
        print('Unknown!')
```

Imaginemos ahora que nos piden calcular la distancia del punto al origen. Debemos tener en cuenta que, en primera instancia, desconocemos si el punto está en el plano o en el espacio:

```python
point = (8, 3, 5)
match point:
    case (int(x), int(y)):
        dist_to_origin = (x ** 2 + y ** 2) ** (1 / 2)
    case (int(x), int(y), int(z)):
        dist_to_origin = (x ** 2 + y ** 2 + z ** 2) ** (1 / 2)
    case _:
        print('Unknown!')

print(dist_to_origin)
```

Con este enfoque, nos aseguramos que los puntos de entrada deben tener todas sus coordenadas como valores enteros:

```python
point = ('8', 3, 5)  # Nótese el 8 como "string"

match point:
    case (int(x), int(y)):
        dist_to_origin = (x ** 2 + y ** 2) ** (1 / 2)
    case (int(x), int(y), int(z)):
        dist_to_origin = (x ** 2 + y ** 2 + z ** 2) ** (1 / 2)
    case _:
        print('Unknown!')
```

Cambiando de ejemplo, a continuación, veremos un código que nos indica si, dada la edad de una persona, puede beber alcohol:

```python
age = 21

match age:
    case 0 | None:
        print('Not a person')
    case n if n < 17:
        print('Nope')
    case n if n < 22:
        print('Not in the US')
    case _:
        print('Yes')
```

En la línea 4 podemos observar el uso del operador OR.

En las líneas 6 y 8 podemos observar el uso de condiciones dando lugar a **cláusulas**.

GUIA DE EJERCICIOS.

1. **Calculadora de operaciones básicas:** Escribe un programa que pida al usuario dos números y una operación (suma, resta, multiplicación o división) y realice la operación correspondiente utilizando match-case.
2. **Determinación del tipo de triángulo:** Escribe un programa que pida al usuario las longitudes de los tres lados de un triángulo y determine si el triángulo es equilátero, isósceles o escaleno utilizando match-case.
3. **Determinación de la estación del año:** Escribe un programa que pida al usuario un mes (en forma de número del 1 al 12) y determine la estación del año correspondiente utilizando match-case.
4. **Determinación del cuadrante de un punto:** Escribe un programa que pida al usuario las coordenadas de un punto en el plano (x, y) y determine en qué cuadrante se encuentra el punto utilizando match-case.
5. **Determinación de la paridad de un número:** Escribe un programa que pida al usuario un número y determine si el número es par o impar utilizando match-case.
6. **Determinación del tipo de número:** Escribe un programa que pida al usuario un número y determine si el número es positivo, negativo o cero utilizando match-case.
7. **Determinación del día de la semana:** Escribe un programa que pida al usuario un día de la semana (en forma de número del 1 al 7) y determine el nombre del día de la semana correspondiente utilizando match-case.
8. **Determinación del tipo de figura geométrica:** Escribe un programa que pida al usuario el número de lados de una figura geométrica y determine el nombre de la figura geométrica correspondiente utilizando match-case.
9. **Determinación del tipo de animal:** Escribe un programa que pida al usuario el nombre de un animal y determine si el animal es un mamífero, un ave, un reptil, un pez o un anfibio utilizando match-case.
10. **Determinación del tipo de vehículo:** Escribe un programa que pida al usuario el número de ruedas de un vehículo y determine si el vehículo es una bicicleta, una motocicleta, un coche o un camión utilizando match-case.