P3. Diccionarios y Clases en Python

Alumna: Daniela Mendez Ramirez Número de Cuenta: 258331-9

Aplicaciones de Redes

Profesor: Omar Vázquez González

Fecha de Entrega: 28 de enero de 2025

Abstract: Este documento presenta una serie de ejercicios prácticos diseñados para fortalecer el conocimiento en Programación Orientada a Objetos (POO) y estructuras de datos en Python. Se abordan conceptos fundamentales como diccionarios, listas, tuplas y clases, aplicados en diversos contextos, incluyendo la modelación de objetos cotidianos como autos y la simulación de una tienda de regalos. Como parte central de la práctica, se desarrolla un sistema de gestión de calificaciones escolares, utilizando clases y relaciones entre objetos para almacenar y manipular información de estudiantes, materias y notas. Este enfoque facilita el aprendizaje de la organización de datos en estructuras reutilizables y escalables, promoviendo una mejor comprensión de los principios de la POO.

Introducción: Esta práctica tiene como objetivo aplicar los principios de POO en la creación de un sistema de gestión escolar, donde los estudiantes, sus materias y calificaciones son representados mediante clases interconectadas. Además, se refuerzan conocimientos sobre diccionarios, listas y tuplas a través de ejercicios complementarios inspirados en ejemplos de la plataforma W3C y situaciones prácticas como la simulación de una tienda de regalos. Al finalizar, los estudiantes habrán adquirido una comprensión más profunda de la organización de datos y el diseño de programas modulares en Python.

Ejercicios de clase de Diccionario:

1. **Practicando con los diccionarios**

```
# Diccionario Persona
persona = {"nombre": "Daniela", "edad": 20, "ciudad": "CDMX"}
print(persona)

#Diccionario Personita Escolar
estudiante = {"nombre": "Juan", "materia": "Matemáticas", "calificación": 9.5}
```

2. **Contamos palabras**

```
productos = {"manzana": 10, "plátano": 5, "naranja": 7}

def contar_palabras(frase):
    palabras = frase.split()
    frecuencia = {}
    for palabra in palabras:
        frecuencia[palabra] = frecuencia.get(palabra, 0) + 1
    return frecuencia

print(contar_palabras("hola mundo hola"))
```

3. **Filtrar un diccionario**

```python
# Funcion de filtrado en diccionario, esto
# nos permite almacenar solo aquellas letras,
#cuya aparicion sea mayor a 5 veces

def filtrar(diccionario):
    return {k: v for k, v in diccionario.items() if v > 5} # Ciclo para recorrer

datos = {"a": 3, "b": 7, "c": 9, "d": 2} # Diccionario que se enviara
print(filtrar(datos))
```

4. **Realizar operaciones**

```python
operaciones = {
    "suma": lambda x, y: x + y,
    "resta": lambda x, y: x - y,
}

print(operaciones["suma"](10, 5))  # Debería imprimir 15
print(operaciones["resta"](20, 15)) # Deberia imprimir 5
```

```
>>> %Run dic
  15
  5
```

Ejercicios de clase de POO (Autos):

```python
class Auto:

    def __init__(self, marca, modelo, posicion=0):
        self.marca = marca
        self.modelo = modelo
        self.posicion = posicion

    def avanzar(self, x=1):
        self.posicion += x

    def imprimir(self):
        print(self.marca, self.modelo, '|', self.posicion, 'Km')

# Creamos miVocho que es el primer objeto
# Su instancia en es
miVocho = Auto('VW', 'Sedan')
miVocho.avanzar(100)
miVocho.imprimir()

# Creamos un nuevo objeto
miFerrari = Auto('Ferrari', 'Clasico', 500)
miFerrari.imprimir()
miCoche = miVocho
miCoche.imprimir()
miCoche.avanzar(200)
miCoche.imprimir()
miVocho.imprimir()

miCoche = None
miCoche = Auto('Nissan', 'Tsuru')
miCoche.imprimir()
```

Evaluamos las impresiones:

```python
# Creamos miVocho que es el primer objeto
# Su instancia en es
miVocho = Auto('VW', 'Sedan')
miVocho.avanzar(100)
miVocho.imprimir()
```

```
VW Sedan | 100 Km
```

```python
# Creamos un nuevo objeto
miFerrari = Auto('Ferrari', 'Clasico', 500)
miFerrari.imprimir()
```

```
>>> %Run Autos.py
VW Sedan | 100 Km
Ferrari Clasico | 500 Km
```

```python
# Creamos nuestro 3er objeto
miCoche = miVocho
miCoche.imprimir()
```

```
>>> %Run Autos.py
VW Sedan | 100 Km
Ferrari Clasico | 500 Km
VW Sedan | 100 Km
```

```python
miCoche = None
miCoche = Auto('Nissan', 'Tsuru')
miCoche.imprimir()
```

```
>>> %Run Autos.py
VW Sedan | 100 Km
Ferrari Clasico | 500 Km
VW Sedan | 100 Km
VW Sedan | 300 Km
VW Sedan | 300 Km
Nissan Tsuru | 0 Km
```

Ejercicios de W3

1. **Declaración:**

Código en Imágen	Salida

`# Ejercicios elaborados de la pag:`
 	
`thisdic = {`
 	
`"marca" : "Ford",`
`"modelo" : "Mustang",`
`"anio" : "2024",`
`}`
 	
`# Asi se imprime un diccionario`
 	
`print(thisdic)`
	# Ejercicios elaborados de la página:
	#
	http://www.w3schools.com/python/python_dictionaries.asp
	
	thisdic = {
	
	"marca" : "Ford",
	"modelo" : "Mustang",
	"anio" : "2024",
	}
	
	# Asi se imprime un diccionario
	print(thisdic)

Salida:

```
{'marca': 'Ford', 'modelo': 'Mustang', 'anio': '2024'}
```

2. **Comprobar que no se permiten duplicados:**

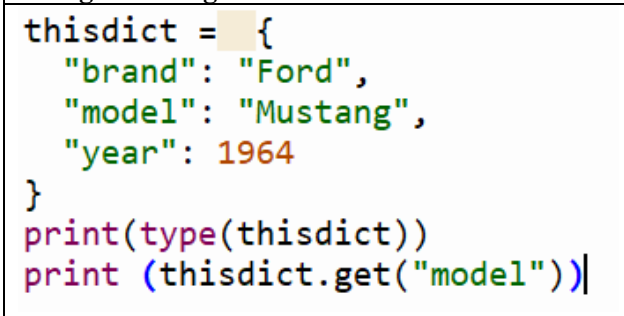
Código en Imágen	Salida

`# Los diccionarios no permiten duplicados`
`# Se queda el ultimo valor ingresado`
`thisdict = {`
`"brand": "Ford",`
`"model": "Mustang",`
`"year": 1964, # En este caso, este ya no se guarda`
`"year": 2020`
`}`
`print(thisdict)`
	# Los diccionarios no permiten duplicados
	# Se queda el ultimo valor ingresado
	thisdict = {
	"brand": "Ford",
	"model": "Mustang",
	"year": 1964, # En este caso, este ya no se guarda
	"year": 2020
	}
	print(thisdict)

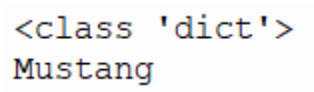
Salida:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

3. **Obtener un valor especifico de un diccionario**

Código en Imágen	Salida
	thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 } print(type(thisdict)) print (thisdict.get("model"))

Salida:



Còdigo de tienda de Regalos

```
# Quiero mandar un regalo a su noviaa, tienda con productos y precios fijos.

# Mostrar lo que la tienda tiene disponible, que deje elegir cuantos tiene


# Inventario inicial con precios y cantidad disponibles
tienda_regalos = {
    "Ositos": {"precio": 100, "stock": 10},
    "Girasoles": {"precio": 200, "stock": 5},
    "Rosas": {"precio": 200, "stock": 15},
    "Tulipanes": {"precio": 300, "stock": 8},
    "Relojes": {"precio": 300, "stock": 3},
    "Collares": {"precio": 200, "stock": 7}
}


def mostrar_inventario():
    print("\n Bienvenido a la Tienda de Regalos")
    print("Estos son los productos disponibles:")
    for producto, detalles in tienda_regalos.items():
        print(f"- {producto}: ${detalles['precio']} (Stock: {detalles['stock']})")


def elegir_producto():
    carrito = {}
    while True:
        mostrar_inventario()
        producto = input("\n¿Qué producto deseas? (Escribe 'salir' para terminar): ").capitalize()
        if producto == "Salir":
            break
        elif producto in tienda_regalos:
            cantidad = int(input(f"¿Cuántos '{producto}' deseas? "))
            if cantidad <= tienda_regalos[producto]["stock"]:
                carrito[producto] = cantidad
                tienda_regalos[producto]["stock"] -= cantidad
                print(f'Has agregado {cantidad} '{producto}' al carrito.")
            else:
                print(f"X No tenemos suficiente stock. Solo hay {tienda_regalos[producto]['stock']} disponibles.")
        else:
            print("X Producto no encontrado. Intenta de nuevo.")
    return carrito


def calcular_total(carrito):
    total = 0
    print("\n Resumen de tu compra:")
    for producto, cantidad in carrito.items():
        precio = tienda_regalos[producto]["precio"]
        total += precio * cantidad
        print(f"- {producto} x{cantidad}: ${precio * cantidad}")
    print(f"\n Total a pagar: ${total}")
    return total
```

```
# Función principal
def tienda():
    carrito = elegir_producto()
    if carrito:
        calcular_total(carrito)
        print("\n¡Gracias por tu compra! ")
    else:
        print("No has agregado nada al carrito. ¡Vuelve pronto!")

# Iniciar tienda
tienda()
```

```
 Bienvenido a la Tienda de Regalos
Estos son los productos disponibles:
- Ositos: $100 (Stock: 10)
- Girasoles: $200 (Stock: 5)
- Rosas: $200 (Stock: 15)
- Tulipanes: $300 (Stock: 8)
- Relojes: $300 (Stock: 3)
- Collares: $200 (Stock: 7)

¿Qué producto deseas? (Escribe 'salir' para terminar):
```

```
¿Qué producto deseas? (Escribe 'salir' para terminar): Ositos
¿Cuántos 'Ositos' deseas? 5
Has agregado 5 'Ositos' al carrito.

 Bienvenido a la Tienda de Regalos
Estos son los productos disponibles:
- Ositos: $100 (Stock: 5)
- Girasoles: $200 (Stock: 5)
- Rosas: $200 (Stock: 15)
- Tulipanes: $300 (Stock: 8)
- Relojes: $300 (Stock: 3)
- Collares: $200 (Stock: 7)

¿Qué producto deseas? (Escribe 'salir' para terminar):
```

```
¿Qué producto deseas? (Escribe 'salir' para terminar): salir

 Resumen de tu compra:
- Ositos x5: $500

 Total a pagar: $500

¡Gracias por tu compra!
```

Código escuelita

```
class Personita:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
        self.materias = []

    def agregar_materia(self, materia):
        self.materias.append(materia)

    def mostrar_calificaciones(self):
        print(f"Calificaciones de {self.nombre}:")
        for materia in self.materias:
            print(f"{materia.nombre}: {materia.obtener_promedio():.2f} (Notas: {materia.listar_calificaciones()})")

class Materia:
    def __init__(self, nombre):
        self.nombre = nombre
        self.calificaciones = []

    def agregar_calificacion(self, calificacion):
        self.calificaciones.append(calificacion)

    def obtener_promedio(self):
        if self.calificaciones:
            return sum(self.calificaciones) / len(self.calificaciones)
        return 0.0

    def listar_calificaciones(self):
        return ", ".join(map(str, self.calificaciones))

# Ejemplo de uso
ejemplo = Personita("Daniela Mendez", 20)
materia1 = Materia("Matemáticas")
materia2 = Materia("Historia")

materia1.agregar_calificacion(90)
materia1.agregar_calificacion(85)
materia2.agregar_calificacion(78)
materia2.agregar_calificacion(82)

ejemplo.agregar_materia(materia1)
ejemplo.agregar_materia(materia2)

ejemplo.mostrar_calificaciones()
```

```
>>> %Run escuelita.py
  Calificaciones de Daniela Mendez:
  Matemáticas: 87.50 (Notas: 90, 85)
  Historia: 80.00 (Notas: 78, 82)
>>>
```

Conclusión:

Durante esta práctica pude ser capaz de desarrollar diferentes habilidades con los diccionarios, y comprender más a fondo como usar la programación orientada a objetos en diversos lenguajes de programación, en lo personal he disfrutado mucho del lenguaje Python, ya que considero que es muy fácil de tener un acercamiento con él, ya que es de fácil manejo, se presta a elaborar un sinfín de cosas y resulta increíblemente divertido descubrir nuevas funcionalidades del sistema.