A red sign with white text

Description automatically generated

**P3. Diccionarios y Clases en Python**

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

**Aplicaciones de Redes**

Profesor: Omar Vázquez González

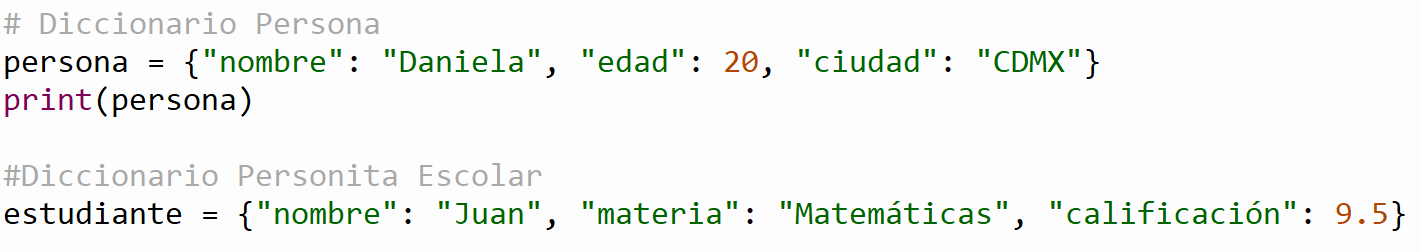
Fecha de Entrega: 28 de enero de 2025

**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**

****

1. **Contamos palabras**

**A screen shot of a computer code

Description automatically generated**

1. **Filtrar un diccionario**

**A computer screen shot of a code

Description automatically generated**

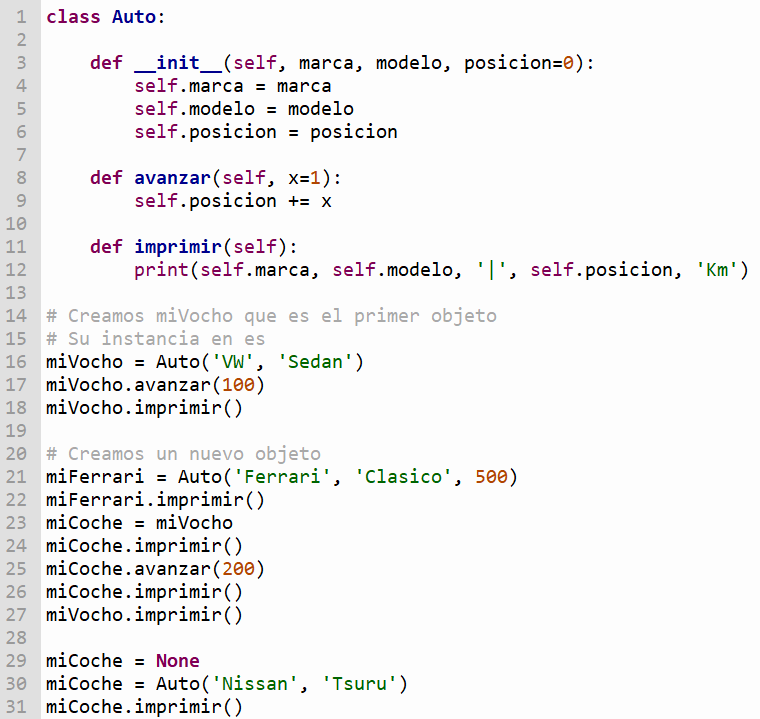
1. **Realizar operaciones**

**A group of numbers and symbols

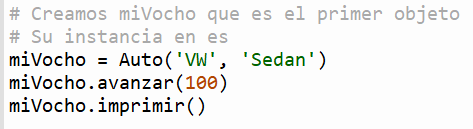
Description automatically generated** **A screenshot of a computer

Description automatically generated**

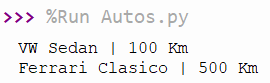
**Ejercicios de clase de POO (Autos):**

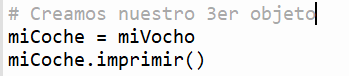
****

**Evaluamos las impresiones:**

****

**A close up of a text

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A close up of text

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Ejercicios de W3**

1. **Declaración:**

|  |  |
| --- | --- |
| **Código en Imágen** | **Salida** |
| **A screenshot of a computer code  Description automatically generated** | # 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:**

****

1. **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) |

**Salida:**

****

1. **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)

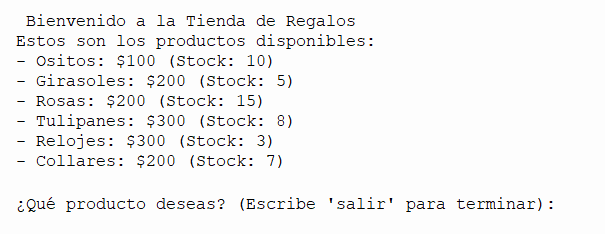
print("\n¡Gracias por tu compra! ")

else:

print("No has agregado nada al carrito. ¡Vuelve pronto!")

# Iniciar tienda

tienda()



A screenshot of a computer code

Description automatically generated

A white background with black text

Description automatically generated

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()

A close up of a computer screen

Description automatically generated

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.