**Exercise 9: Classes**

Please work through as many of the following exercises as you’re able to in the allotted time and upload the results of the last task you were able to complete

1. Make a class called, “Restaurant”. The \_\_init\_\_() method for Restaurant should store two attributes: a restaurant\_name and a cuisine\_type. Make a method called describe\_restaurant() that prints these two pieces of information, and a method called open\_restaurant() that prints a message indicating that the restaurant is open. Make an instance called, “restaurant” from your class. Print the two attributes individually, and then call both methods.

* **Stretch and Challenge:** Start with your Restaurant class. Create three different instances from the class, and call describe\_restaurant() for each instance
* **Stretch and Challenge 2:** Start with your Restaurant Class program. Add an attribute called number\_served with a default value of 0. Create an instance called restaurant from this class. Print the number of customers the restaurant has served, and then change this value and print it again. Add a method called set\_number\_served() that lets you set the number of customers that have been served. Call this method with a new number and print the value again. Add a method called increment\_number\_served() that lets you increment the number of customers who’ve been served. Call this method with any number you like that could represent how many customers were served in, say, a day of business.

class Restaurant:

def \_\_init\_\_(self, restaurant\_name, cuisine\_type):

self.restaurant\_name = restaurant\_name

self.cuisine\_type = cuisine\_type

self.number\_served = 0

def describe\_restaurant(self):

print(f"Restaurant Name: {self.restaurant\_name}")

print(f"Cuisine Type: {self.cuisine\_type}")

def open\_restaurant(self):

print(f"{self.restaurant\_name} is open!")

def set\_number\_served(self, number):

self.number\_served = number

def increment\_number\_served(self, increment):

self.number\_served += increment

# Creating an instance of Restaurant

restaurant = Restaurant("Food Haven", "Italian")

# Printing individual attributes

print("Restaurant Name:", restaurant.restaurant\_name)

print("Cuisine Type:", restaurant.cuisine\_type)

# Calling methods

restaurant.describe\_restaurant()

restaurant.open\_restaurant()

# Stretch and Challenge

restaurant1 = Restaurant("Taste of India", "Indian")

restaurant2 = Restaurant("Sushi Palace", "Japanese")

restaurant3 = Restaurant("Mexican Fiesta", "Mexican")

restaurants = [restaurant1, restaurant2, restaurant3]

for res in restaurants:

res.describe\_restaurant()

1. Make a class called User. Create two attributes called first\_name and last\_name, and then create several other attributes that are typically stored in a user profile. Make a method called describe\_user() that prints a summary of the user’s information. Make another method called greet\_user() that prints a personalized greeting to the user. Create several instances representing different users, and call both methods for each user.

* **Stretch and Challenge:** Add an attribute called login\_attempts to your User class. Write a method called increment\_ login\_attempts() that increments the value of login\_attempts by 1. Write another method called reset\_login\_attempts() that resets the value of login\_ attempts to 0. Make an instance of the User class and call increment\_login\_attempts() several times. Print the value of login\_attempts to make sure it was incremented properly, and then call reset\_login\_attempts(). Print login\_attempts again to make sure it was reset to 0.

class User:

def \_\_init\_\_(self, first\_name, last\_name, email, age):

self.first\_name = first\_name

self.last\_name = last\_name

self.email = email

self.age = age

self.login\_attempts = 0

def describe\_user(self):

print(f"First Name: {self.first\_name}")

print(f"Last Name: {self.last\_name}")

print(f"Email: {self.email}")

print(f"Age: {self.age}")

def greet\_user(self):

print(f"Hello, {self.first\_name} {self.last\_name}!")

def increment\_login\_attempts(self):

self.login\_attempts += 1

def reset\_login\_attempts(self):

self.login\_attempts = 0

# Creating instances of User

user1 = User("John", "Doe", "john@example.com", 30)

user2 = User("Alice", "Smith", "alice@example.com", 25)

# Calling methods

user1.describe\_user()

user1.greet\_user()

user2.describe\_user()

user2.greet\_user()

# Stretch and Challenge

user3 = User("Bob", "Johnson", "bob@example.com", 35)

user3.increment\_login\_attempts()

user3.increment\_login\_attempts()

user3.increment\_login\_attempts()

print("Login Attempts:", user3.login\_attempts)

user3.reset\_login\_attempts()

print("Login Attempts Reset:", user3.login\_attempts)

1. Create a class called "Car" with attributes for make, model, and year. Implement a method called get\_info() to print out the details of the car. Create instances representing different cars and call the get\_info() method for each instance.

* **Stretch and Challenge:** Add an attribute called "mileage" to the Car class and implement methods to set and update the mileage of the car.

class Car:

def \_\_init\_\_(self, make, model, year):

self.make = make

self.model = model

self.year = year

self.mileage = 0

def get\_info(self):

print(f"Car Details:")

print(f"Make: {self.make}")

print(f"Model: {self.model}")

print(f"Year: {self.year}")

print(f"Mileage: {self.mileage} miles")

def update\_mileage(self, mileage):

self.mileage = mileage

# Creating instances of Car

car1 = Car("Toyota", "Camry", 2019)

car2 = Car("Honda", "Accord", 2020)

# Calling methods

car1.get\_info()

car2.get\_info()

# Stretch and Challenge

car1.update\_mileage(15000)

car1.get\_info()

1. Define a class called "Book" with attributes for title, author, and genre. Implement a method called display\_info() to print out the details of the book. Create instances representing different books and call the display\_info() method for each instance.

* **Stretch and Challenge:** Add an attribute called "pages" to the Book class and implement methods to set and update the number of pages in the book.

class Book:

def \_\_init\_\_(self, title, author, genre):

self.title = title

self.author = author

self.genre = genre

self.pages = 0

def display\_info(self):

print(f"Book Details:")

print(f"Title: {self.title}")

print(f"Author: {self.author}")

print(f"Genre: {self.genre}")

print(f"Pages: {self.pages} pages")

def update\_pages(self, pages):

self.pages = pages

# Creating instances of Book

book1 = Book("To Kill a Mockingbird", "Harper Lee", "Fiction")

book2 = Book("1984", "George Orwell", "Dystopian")

# Calling methods

book1.display\_info()

book2.display\_info()

# Stretch and Challenge

book1.update\_pages(281)

book1.display\_info()

1. Create a class called "BankAccount" with attributes for account number, account holder, and balance. Implement methods for depositing and withdrawing money from the account, as well as checking the balance. Create instances representing different bank accounts and perform transactions on them.

* **Stretch and Challenge:** Add an attribute called "interest\_rate" to the BankAccount class and implement methods to calculate and apply interest to the account balance.

class BankAccount:

def \_\_init\_\_(self, account\_number, account\_holder, balance):

self.account\_number = account\_number

self.account\_holder = account\_holder

self.balance = balance

self.interest\_rate = 0.05

def deposit(self, amount):

self.balance += amount

def withdraw(self, amount):

if amount <= self.balance:

self.balance -= amount

else:

print("Insufficient funds")

def check\_balance(self):

print(f"Account Balance: ${self.balance}")

def apply\_interest(self):

interest\_amount = self.balance \* self.interest\_rate

self.balance += interest\_amount

# Creating instances of BankAccount

account1 = BankAccount("123456789", "John Doe", 1000)

account2 = BankAccount("987654321", "Alice Smith", 2000)

# Performing transactions

account1.deposit(500)

account1.withdraw(200)

account1.check\_balance()

# Stretch and Challenge

account1.apply\_interest()

account1.check\_balance()