Objectives:

* Classes and Objects

**There are 6 Challenge Exercises, each worth 16.6%.**

**Project #1 (Single Class)**

In this example, we define a class called Person. **The \_\_init\_\_ method initializes the name and age attributes to default values.** The get\_input method prompts the user to enter their name and age from the console and assigns the values to the corresponding attributes. The display method prints the entered name and age to the console.

We create an instance of the Person class and call the get\_input method to take input from the user for name and age. Finally, we call the display method to show the entered name and age on the console.

A screenshot of a computer program

Description automatically generated

**Project #2 (Two Classes)**

In this example, we define two classes: Student and Course. The student class has attributes name and student\_id, and the Course class has attributes course\_number and course\_name. Each class has get\_input() and display() methods.

We create an instance of the Student class, get input for the student's name and ID using the get\_input() method, and display the student's information using the display() method.

We also create an instance of the Course class, get input for the course number and name using the get\_input() method, and display the course information using the display() method.

A computer screen with text and images

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Project #3 (Three Classes) using separate individual classes.**

**Customer Class**

A screen shot of a computer

Description automatically generated

**Order Class**

A screen shot of a computer program

Description automatically generated

**Main Class**

A computer screen shot of a program

Description automatically generated

**Project #4**

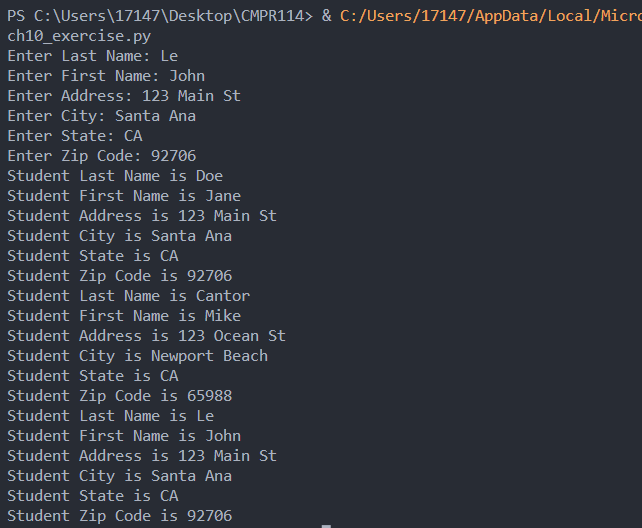
This class is an example of creating objects and using the *self*-keyword to access variables within this class.

Text

Description automatically generated

**Challenge Exercise #1:** Continuing from project #4, create a third object (Student3) where the user will be able to enter the student’s information.

**#1 Print screen the input/output with the code below here.**

****

Code:

class Student:

def \_\_init\_\_(self):

self.lastname = ""

self.firstname = ""

self.address = ""

self.city = ""

self.state = ""

self.zipcode = ""

def get\_input(self):

self.lastname = input("Enter Last Name: ")

self.firstname = input("Enter First Name: ")

self.address = input("Enter Address: ")

self.city = input("Enter City: ")

self.state= input("Enter State: ")

self.zipcode= input("Enter Zip Code: ")

def GetInformation(self):

print(f"Student Last Name is {self.lastname}")

print(f"Student First Name is {self.firstname}")

print(f"Student Address is {self.address}")

print(f"Student City is {self.city}")

print(f"Student State is {self.state}")

print(f"Student Zip Code is {self.zipcode}")

student1 = Student()

student1.lastname = "Doe"

student1.firstname = "Jane"

student1.address = "123 Main St"

student1.city = "Santa Ana"

student1.state = "CA"

student1.zipcode = "92706"

student2 = Student()

student2.lastname = "Cantor"

student2.firstname = "Mike"

student2.address = "123 Ocean St"

student2.city = "Newport Beach"

student2.state = "CA"

student2.zipcode = "65988"

student3 = Student()

student3.get\_input()

student1.GetInformation()

student2.GetInformation()

student3.GetInformation()

**Project #5**

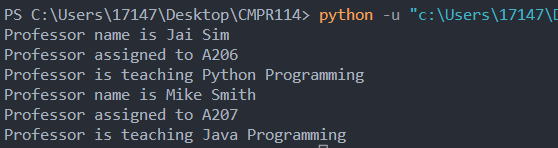
Here we are creating a customized constructor by using the **\_\_init\_\_** function and passing three arguments starting with name, classroom, and then course. This is a lot cleaner than project #1 but function or behaves the same way.

**Text

Description automatically generated**

**Challenge Exercise #2:** Add a second object for Teacher 2 and add any teacher name, room #, and course assigned.

**#2 Print screen the input/output with the code below here.**



Code:

class Teacher:

def \_\_init\_\_(self, name, classroom, course):

self.name = name

self.classroom = classroom

self.course = course

def GetProfessor(self):

print(f"Professor name is {self.name}")

print(f"Professor assigned to {self.classroom}")

print(f"Professor is teaching {self.course}")

teacher1 = Teacher("Jai Sim", "A206", "Python Programming")

teacher1.GetProfessor()

teacher2 = Teacher("Mike Smith", "A207", "Java Programming")

teacher2.GetProfessor()

**Project #6**

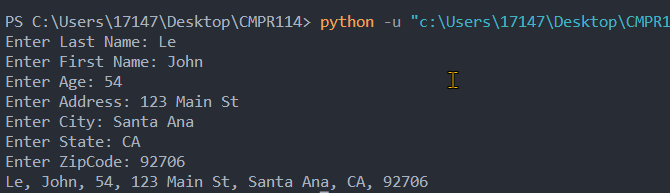
This class is an example of passing parameters and accessing those parameters to get an output.

Text

Description automatically generated

**Challenge Exercise #3:** for project #6, add the address, city, and state with the zip code.

**#3 Print screen the output with the code below here.**

****

Code:

class PI:

def GetInformation(self, LN, FN, Age, Address, City, State, ZipCode):

return LN + ", " + FN + ", " + Age + ", " + Address + ", " + City + ", " + State + ", " + ZipCode

PIData = PI()

PIData.LastName = input("Enter Last Name: ")

PIData.FirstName = input("Enter First Name: ")

PIData.Age = input("Enter Age: ")

PIData.Address = input("Enter Address: ")

PIData.City = input("Enter City: ")

PIData.State = input("Enter State: ")

PIData.ZipCode = input("Enter ZipCode: ")

print(PIData.GetInformation(PIData.LastName,

PIData.FirstName,

PIData.Age,

PIData.Address,

PIData.City,

PIData.State,

PIData.ZipCode))

**Project #7**

The Customer class creates functions and objects.

Text

Description automatically generated

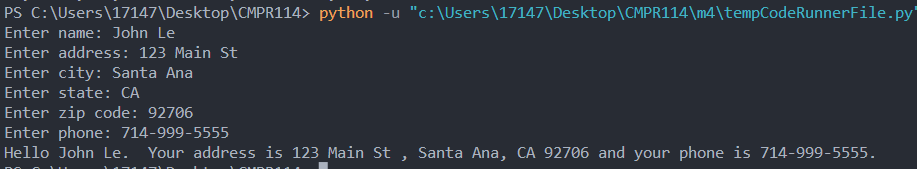
**The Second Class**

Text

Description automatically generated

**Challenge Exercise #4:** for project #7, add the city, and zip code with the person’s age.

**#4 Print screen the output with the code below here.**



Code:

class Customers:

def \_\_init\_\_(self, name, address, city, state, zipcode, phone):

self.name = name

self.address = address

self.city = city

self.state = state

self.zipcode = zipcode

self.phone= phone

def set\_name(self, name):

self.\_\_name = name

def set\_address(self, address):

self.\_\_address = address

def set\_city(self, city):

self.\_\_city = city

def set\_state(self, state):

self.\_\_state = state

def set\_zipcode(self, zipcode):

self.\_\_zipcode = zipcode

def set\_phone(self, phone):

self.\_\_phone = phone

import Customers

def main():

name = input("Enter name: ")

address = input("Enter address: ")

city = input("Enter city: ")

state = input("Enter state: ")

zipcode = input("Enter zip code: ")

phone = input("Enter phone: ")

var1 = Customers.Customers.set\_name = name

var2 = Customers.Customers.set\_address = address

var3 = Customers.Customers.set\_city = city

var4 = Customers.Customers.set\_state = state

var5 = Customers.Customers.set\_zipcode = zipcode

var6 = Customers.Customers.set\_phone = phone

print(f"Hello {var1}. Your address is {var2} , {var3}, {var4} {var5} and your phone is {var6}.")

main()

**Project #8**

**First Class:** Here we have a class named BankAccount which will be called from another class or an external class. The \_\_init\_\_ allows us to create a custom constructor.

Graphical user interface, text, application, email

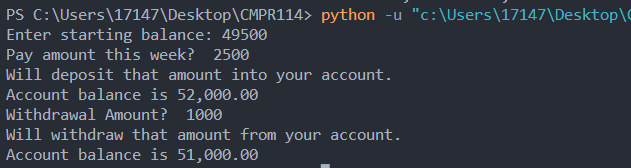
Description automatically generated

**Second Class**

Text

Description automatically generated

**#5 Print screen the input/output with the code below here.**



Code:

class BankAccount:

def \_\_init\_\_ (self, balance):

self.\_\_balance = balance

def deposit(self, amount):

self.\_\_balance+=amount

def withdraw(self, amount):

if self.\_\_balance >= amount:

self.\_\_balance-=amount

else:

print("Insufficient funds.")

def get\_balance(self):

return self.\_\_balance

import BankAccounts

def main():

start\_bal = float(input("Enter starting balance: "))

#Initialize class BankAccount

savings = BankAccounts.BankAccount(start\_bal)

pay = float(input("Pay amount this week? "))

print("Will deposit that amount into your account.")

#add to account based upon pay

savings.deposit(pay)

print(f"Account balance is {savings.get\_balance():,.2f}")

cash = float(input("Withdrawal Amount? "))

print("Will withdraw that amount from your account.")

savings.withdraw(cash)

print(f"Account balance is {savings.get\_balance():,.2f}")

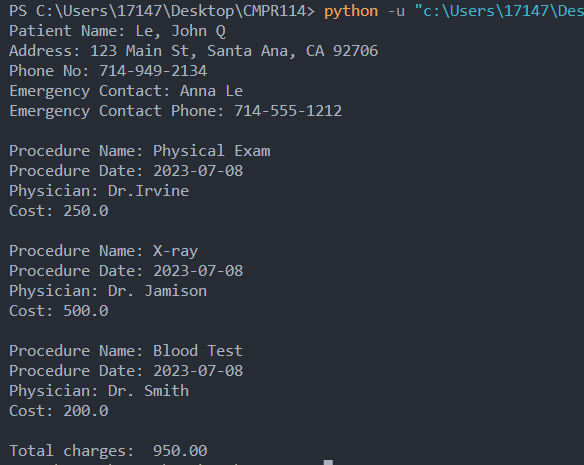
main()

**Challenge Exercise: Complete the following Patient Charges Application below.**

A blue medical form with white text

Description automatically generated

**#6 Print screen the input/output with the code below here. (Need print screen of the code for all classes).**



Code:

class Patient:

def \_\_init\_\_(self, firstname, middlename, lastname, address, city, state, zipcode, phone, emergency\_contact, emergency\_phone):

self.firstname = firstname

self.middlename = middlename

self.lastname = lastname

self.address = address

self.city = city

self.state = state

self.zipcode = zipcode

self.phone = phone

self.emergency\_contact = emergency\_contact

self.emergency\_phone = emergency\_phone

def get\_firstname(self):

return self.firstname

def get\_middlename(self, value):

return self.middlename

def get\_lastname(self, value):

return self.lastname

def get\_address(self, value):

return self.address

def get\_city(self, value):

return self.city

def get\_state(self, value):

return self.state

def get\_zipcode(self, value):

return self.zipcode

def get\_phone(self, value):

return self.phone

def get\_emergency\_contact(self, value):

return self.emergency\_contact

def get\_emergency\_phone(self, value):

return self.emergency\_phone

def get\_patient\_info(self):

msg = f"Patient Name: {self.lastname}, {self.firstname} {self.middlename}\n"

msg = msg + f"Address: {self.address}, {self.city}, {self.state} {self.zipcode}\n"

msg = msg + f"Phone No: {self.phone}\n"

msg = msg + f"Emergency Contact: {self.emergency\_contact}\n"

msg = msg + f"Emergency Contact Phone: {self.emergency\_phone}\n"

return msg

def set\_firstname(self, value):

self.firstname = value

def set\_middlename(self, value):

self.middlename = value

def set\_lastname(self, value):

self.lastname = value

def set\_address(self, value):

self.address = value

def set\_city(self, value):

self.city = value

def set\_state(self, value):

self.state = value

def set\_zipcode(self, value):

self.zipcode = value

def set\_phone(self, value):

self.phone = value

def set\_emergency\_contact(self, value):

self.emergency\_contact = value

def set\_emergency\_phone(self, value):

self.emergency\_phone = value

import Patients

import datetime as datetime

class Procedure:

def \_\_init\_\_(self, procname, procdate, doctor, charges):

self.procedurename = procname

self.proceduredate = procdate

self.doctor = doctor

self.charges = charges

#Assessor attributes

def get\_procedurename(self):

return self.procedurename

def get\_proceduredate(self):

return self.proceduredate

def get\_doctor(self):

return self.doctor

def get\_charges(self):

return self.charges

def get\_procedure\_info(self):

msg = f"Procedure Name: {self.procedurename}\n"

msg = msg + f"Procedure Date: {self.proceduredate}\n"

msg = msg + f"Physician: {self.doctor}\n"

msg = msg + f"Cost: {self.charges}\n"

return msg

def set\_procedurename(self, value):

self.procedurename = value

#Mutator attributes

def set\_proceduredate(self, value):

self.proceduredate = value

def set\_doctor(self, value):

self.doctor = value

def set\_charge(self, value):

self.charges = value

patient1 = Patients.Patient("John", "Q", "Le", "123 Main St", "Santa Ana", "CA", "92706", "714-949-2134", "Anna Le", "714-555-1212")

proc1 = Procedure("Physical Exam", datetime.date.today(), "Dr.Irvine", 250.00)

proc2 = Procedure("X-ray", datetime.date.today(), "Dr. Jamison", 500.00)

proc3 = Procedure("Blood Test", datetime.date.today(), "Dr. Smith", 200.00)

#Print Patient Info

print(patient1.get\_patient\_info())

#Print Procedure

print(proc1.get\_procedure\_info())

print(proc2.get\_procedure\_info())

print(proc3.get\_procedure\_info())

totalcharges = proc1.get\_charges() + proc2.get\_charges() + proc3.get\_charges()

print(f"Total charges: {totalcharges:,.2f}")

**Submit this document to the Module 4 Class Exercise.**