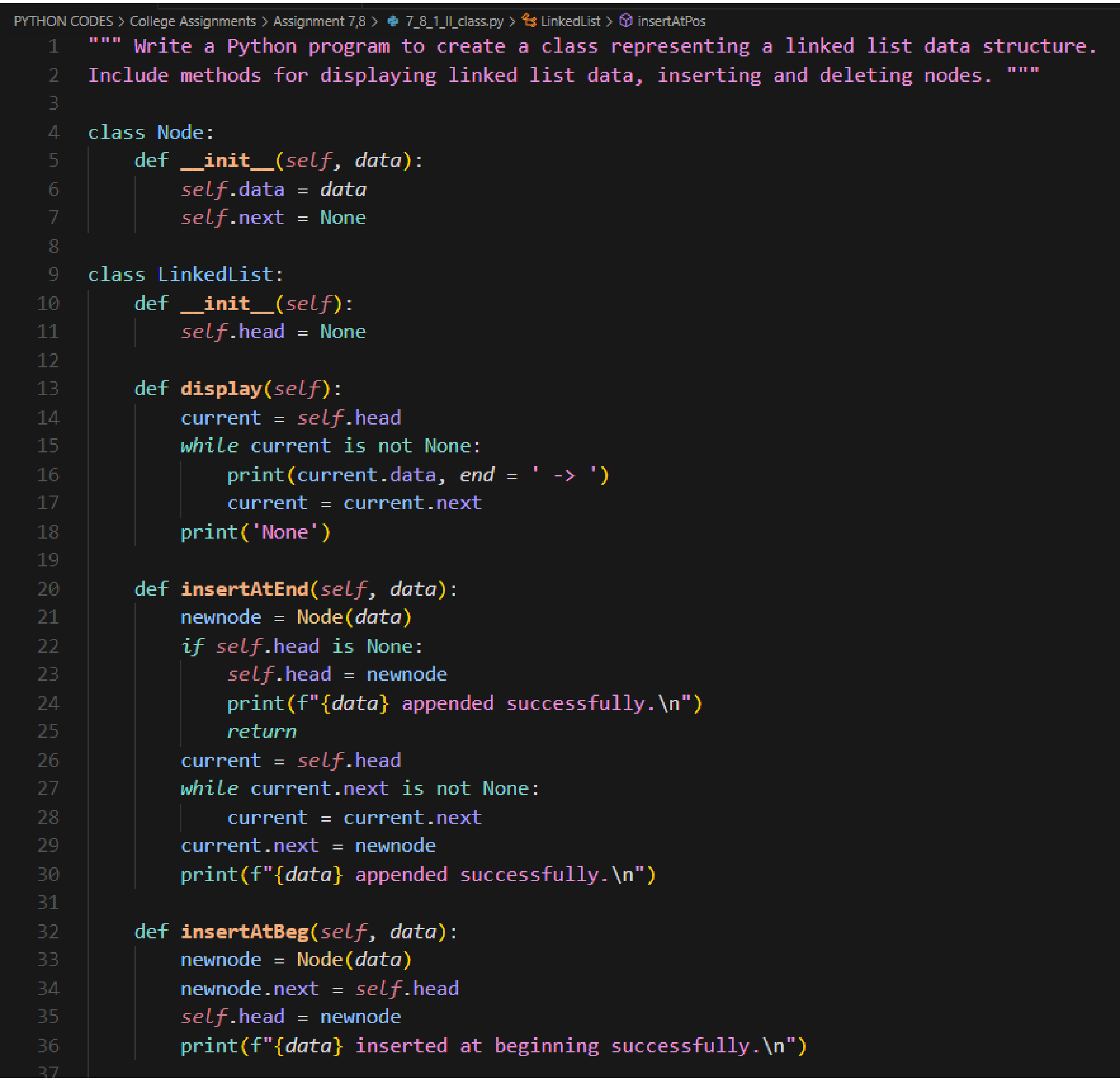
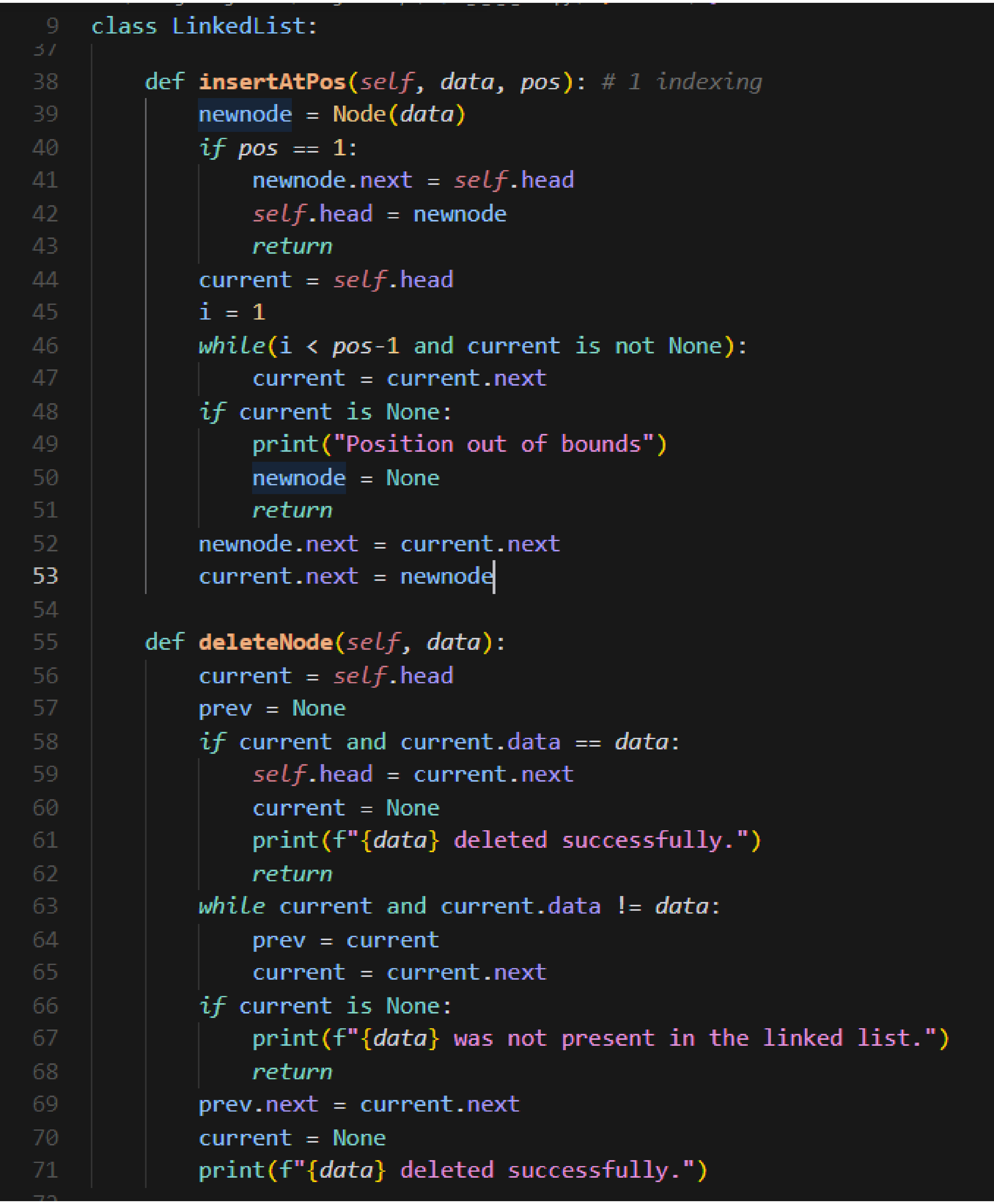
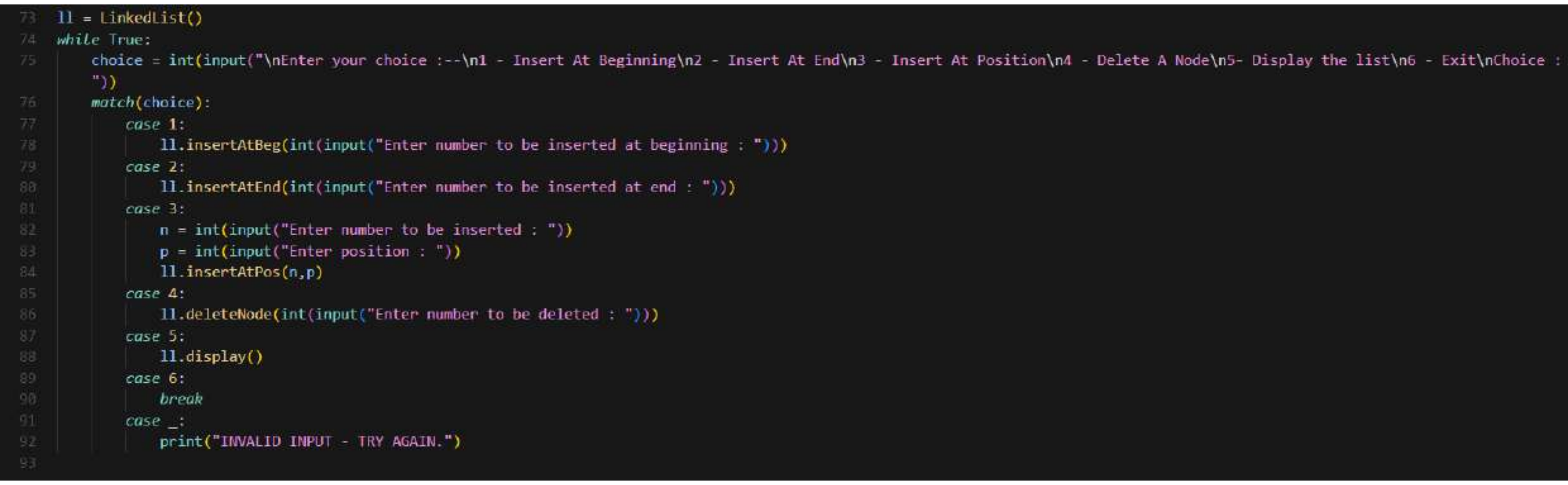

PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_1_ll_class.py > LinkedList > insertAtPos

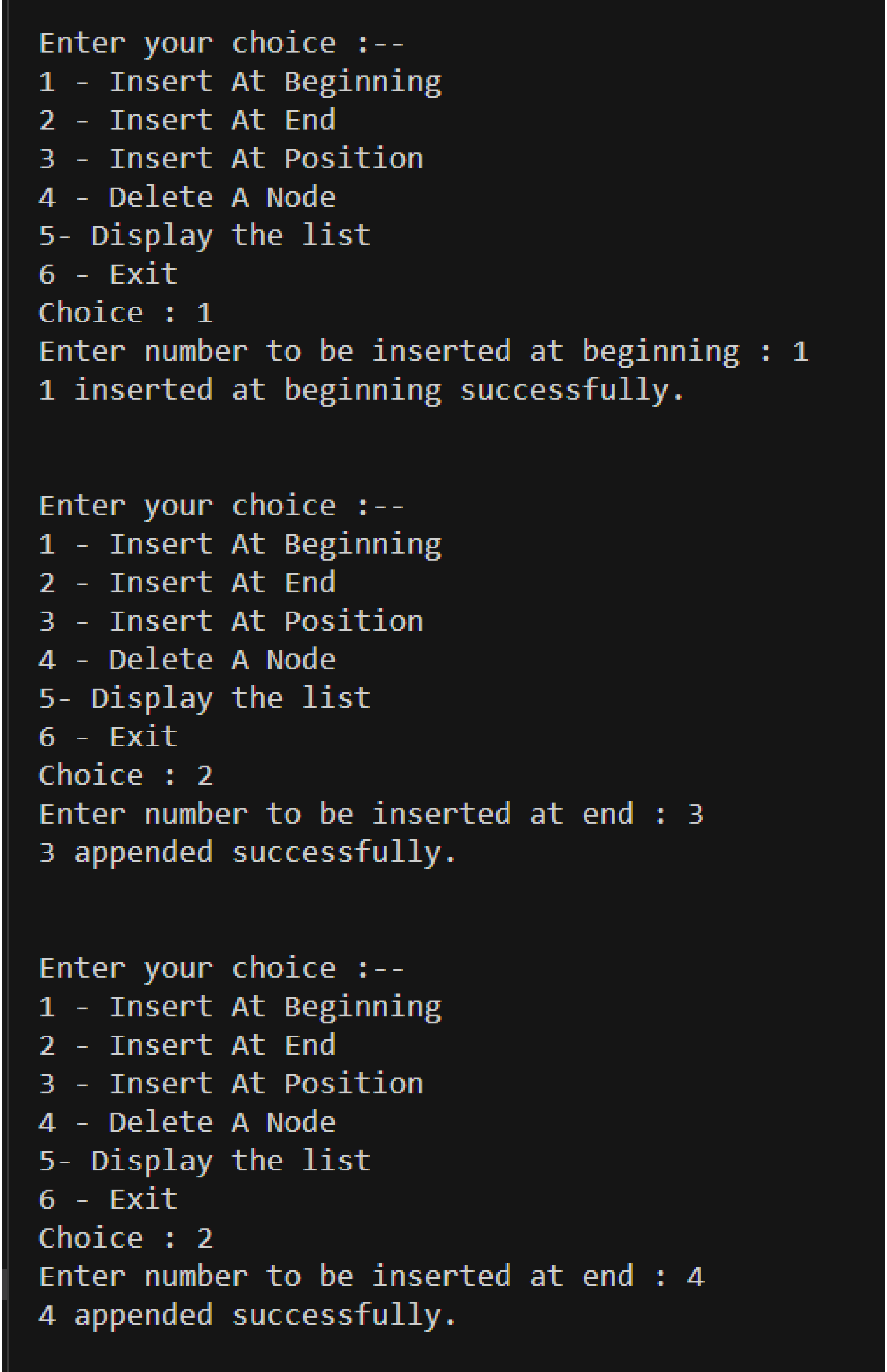
```python
""" Write a Python program to create a class representing a linked list data structure.
Include methods for displaying linked list data, inserting and deleting nodes. """

class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def display(self):
        current = self.head
        while current is not None:
            print(current.data, end = ' -> ')
            current = current.next
        print('None')

    def insertAtEnd(self, data):
        newnode = Node(data)
        if self.head is None:
            self.head = newnode
            print(f"{data} appended successfully.\n")
            return
        current = self.head
        while current.next is not None:
            current = current.next
        current.next = newnode
        print(f"{data} appended successfully.\n")

    def insertAtBeg(self, data):
        newnode = Node(data)
        newnode.next = self.head
        self.head = newnode
        print(f"{data} inserted at beginning successfully.\n")
```



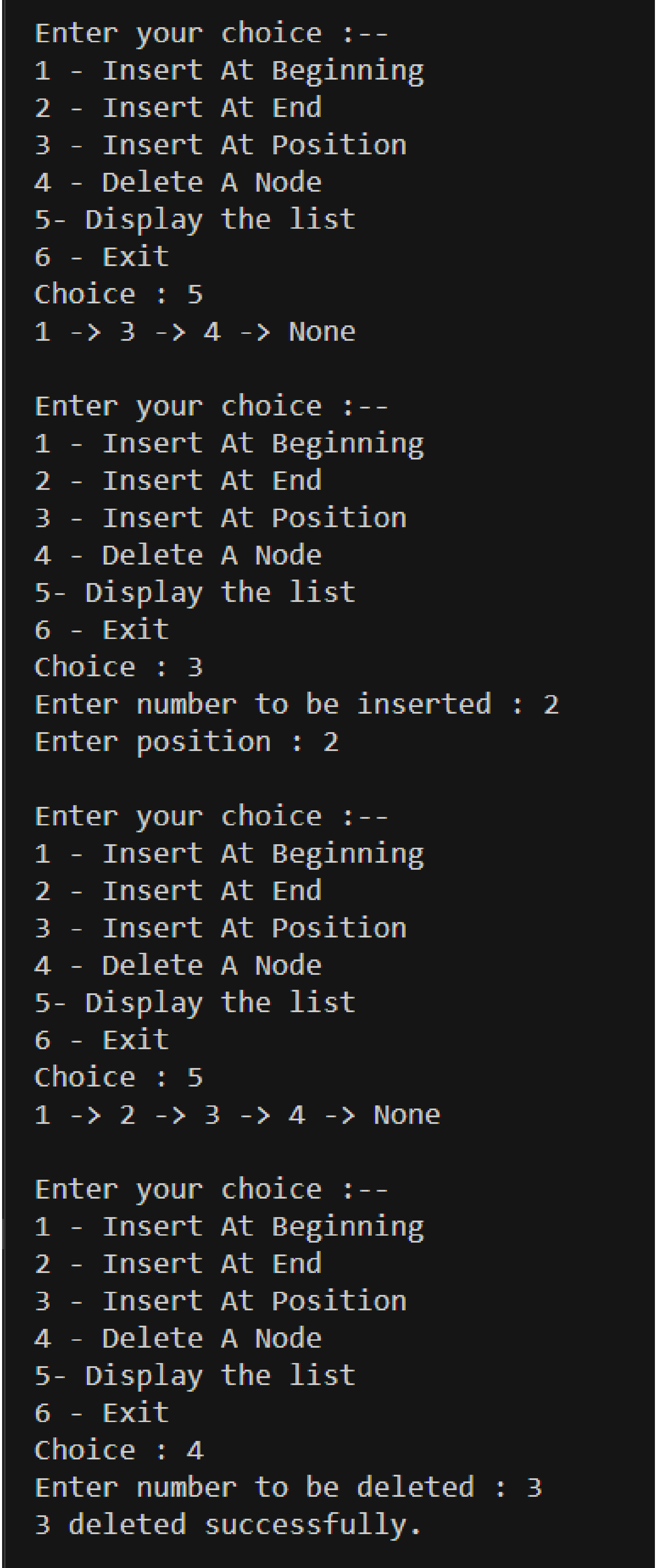
```python
class LinkedList:

    def insertAtPos(self, data, pos): # 1 indexing
        newnode = Node(data)
        if pos == 1:
            newnode.next = self.head
            self.head = newnode
            return
        current = self.head
        i = 1
        while(i < pos-1 and current is not None):
            current = current.next
        if current is None:
            print("Position out of bounds")
            newnode = None
            return
        newnode.next = current.next
        current.next = newnode

    def deleteNode(self, data):
        current = self.head
        prev = None
        if current and current.data == data:
            self.head = current.next
            current = None
            print(f"{data} deleted successfully.")
            return
        while current and current.data != data:
            prev = current
            current = current.next
        if current is None:
            print(f"{data} was not present in the linked list.")
            return
        prev.next = current.next
        current = None
        print(f"{data} deleted successfully.")
```



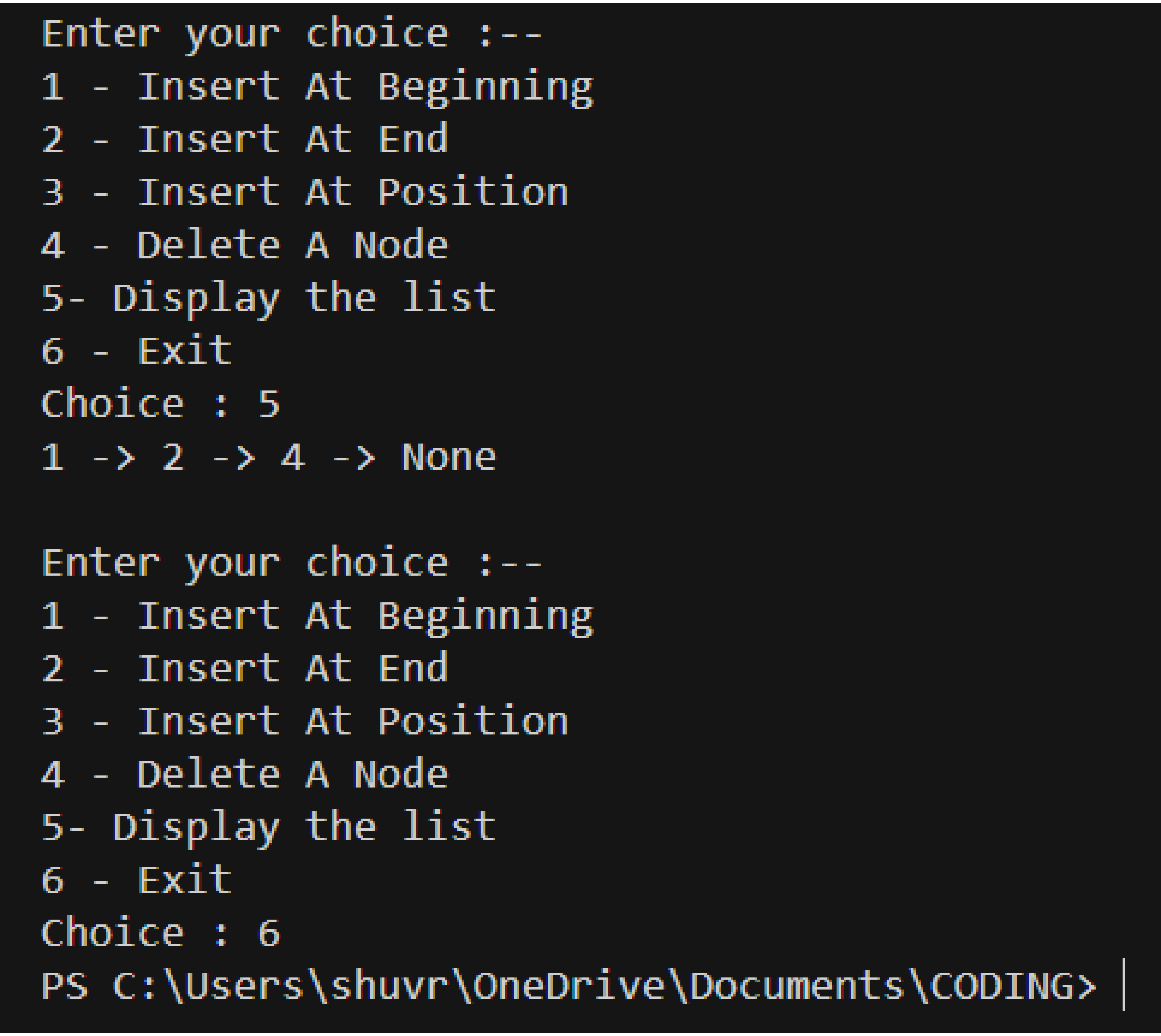
```python
ll = LinkedList()
while True:
    choice = int(input("\nEnter your choice :--\n1 - Insert At Beginning\n2 - Insert At End\n3 - Insert At Position\n4 - Delete A Node\n5- Display the list\n6 - Exit\nChoice : "))
    match(choice):
        case 1:
            ll.insertAtBeg(int(input("Enter number to be inserted at beginning : ")))
        case 2:
            ll.insertAtEnd(int(input("Enter number to be inserted at end : ")))
        case 3:
            n = int(input("Enter number to be inserted : "))
            p = int(input("Enter position : "))
            ll.insertAtPos(n,p)
        case 4:
            ll.deleteNode(int(input("Enter number to be deleted : ")))
        case 5:
            ll.display()
        case 6:
            break
        case _:
            print("INVALID INPUT - TRY AGAIN.")
```



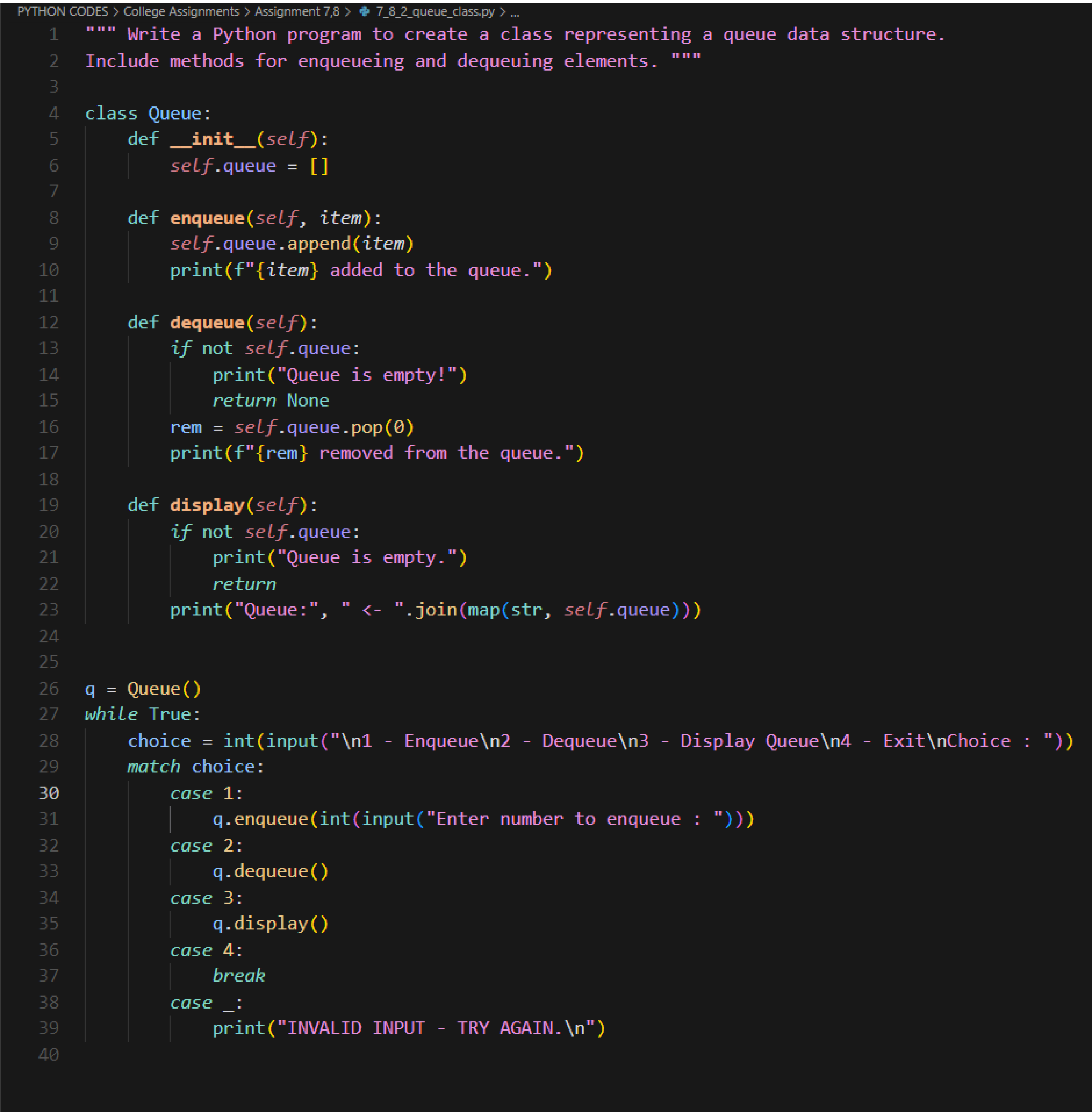
```
Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 1
Enter number to be inserted at beginning : 1
1 inserted at beginning successfully.


Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 2
Enter number to be inserted at end : 3
3 appended successfully.


Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 2
Enter number to be inserted at end : 4
4 appended successfully.
```



```
Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 5
1 -> 3 -> 4 -> None

Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 3
Enter number to be inserted : 2
Enter position : 2

Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 5
1 -> 2 -> 3 -> 4 -> None

Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 4
Enter number to be deleted : 3
3 deleted successfully.
```

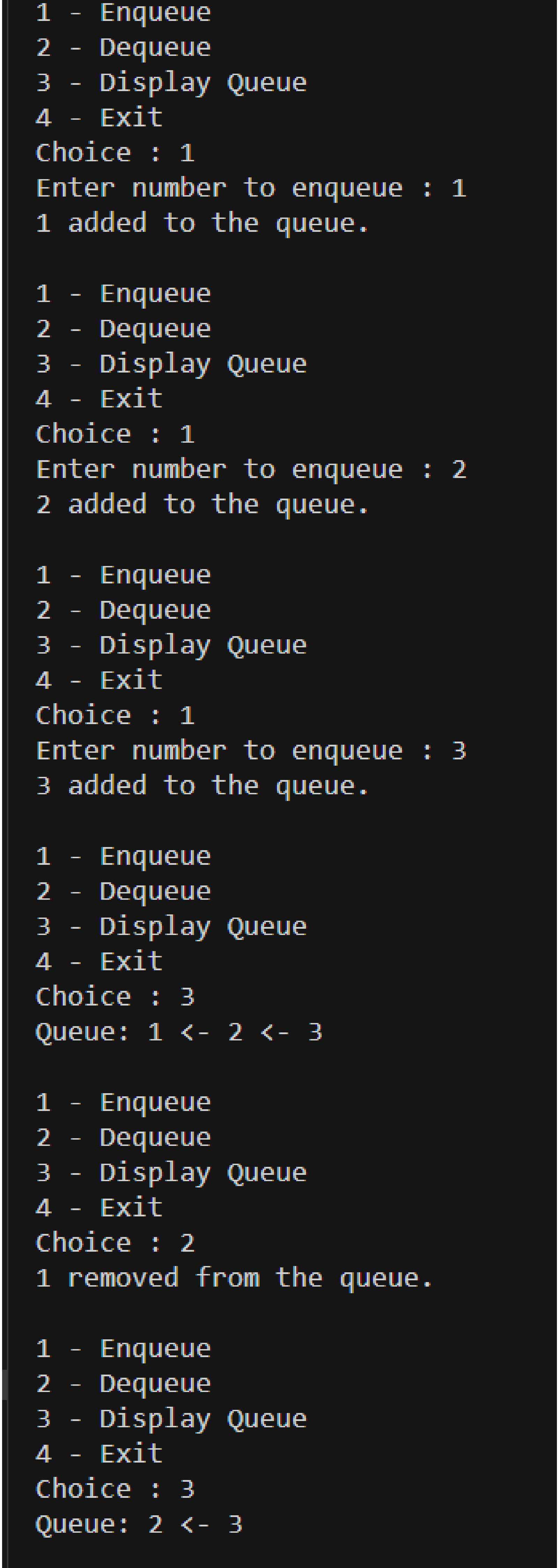


```
Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 5
1 -> 2 -> 4 -> None

Enter your choice :--
1 - Insert At Beginning
2 - Insert At End
3 - Insert At Position
4 - Delete A Node
5- Display the list
6 - Exit
Choice : 6
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```

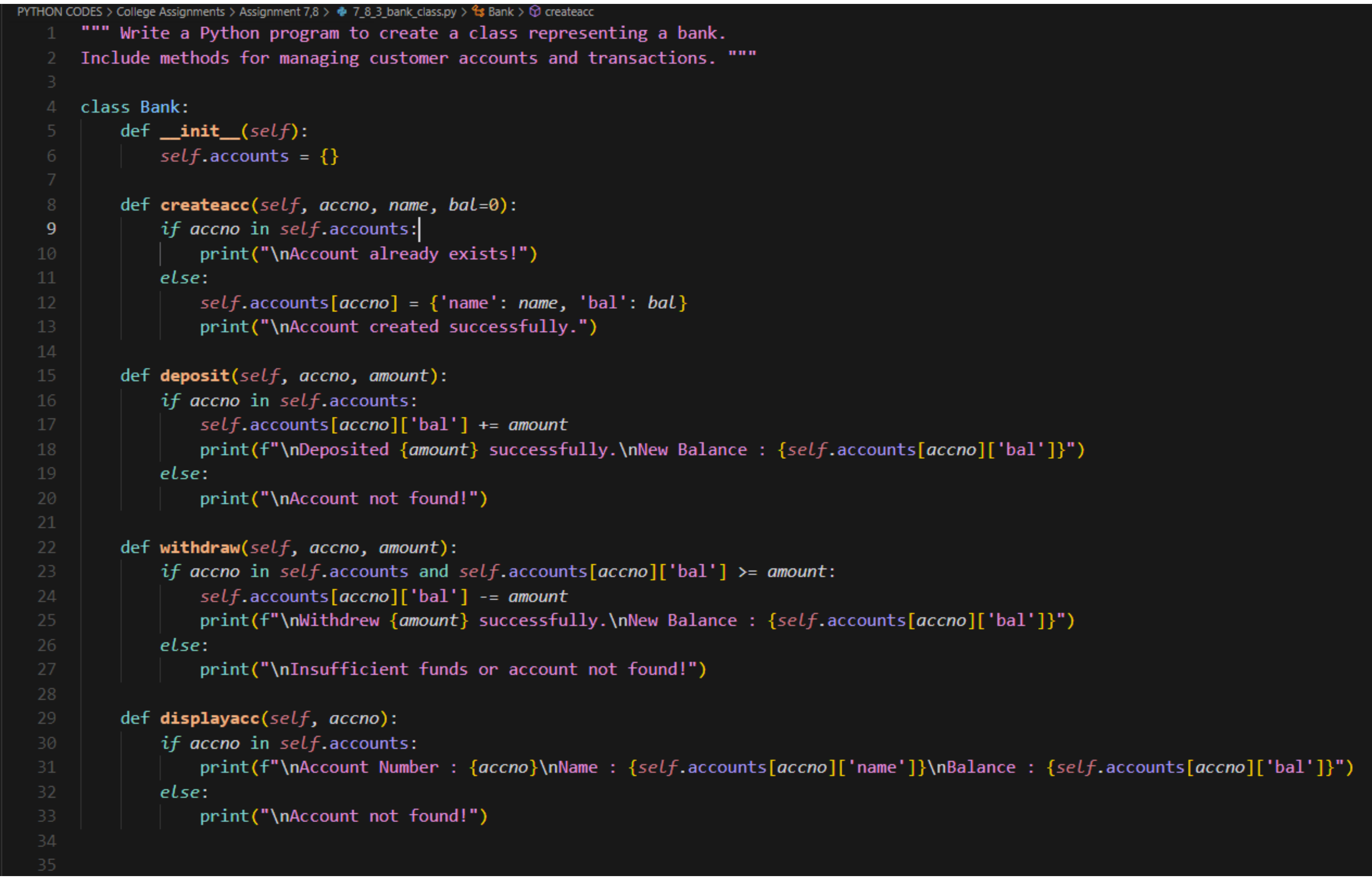


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_2_queue_class.py > ...

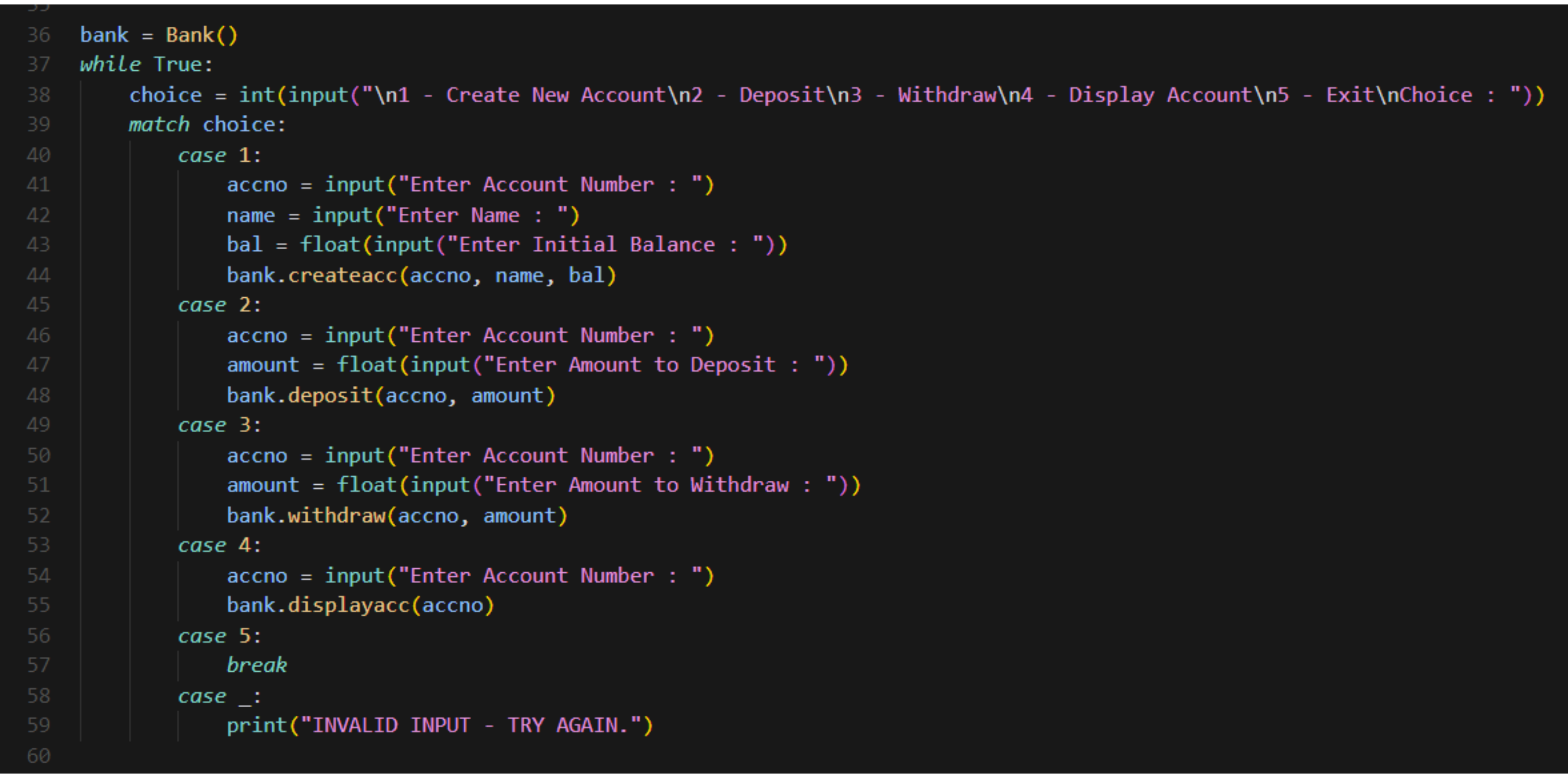
```python
""" Write a Python program to create a class representing a queue data structure.
Include methods for enqueueing and dequeuing elements. """

class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, item):
        self.queue.append(item)
        print(f"{item} added to the queue.")

    def dequeue(self):
        if not self.queue:
            print("Queue is empty!")
            return None
        rem = self.queue.pop(0)
        print(f"{rem} removed from the queue.")

    def display(self):
        if not self.queue:
            print("Queue is empty.")
            return
        print("Queue:", " <- ".join(map(str, self.queue)))


q = Queue()
while True:
    choice = int(input("\n1 - Enqueue\n2 - Dequeue\n3 - Display Queue\n4 - Exit\nChoice : "))
    match choice:
        case 1:
            q.enqueue(int(input("Enter number to enqueue : ")))
        case 2:
            q.dequeue()
        case 3:
            q.display()
        case 4:
            break
        case _:
            print("INVALID INPUT - TRY AGAIN.\n")

```



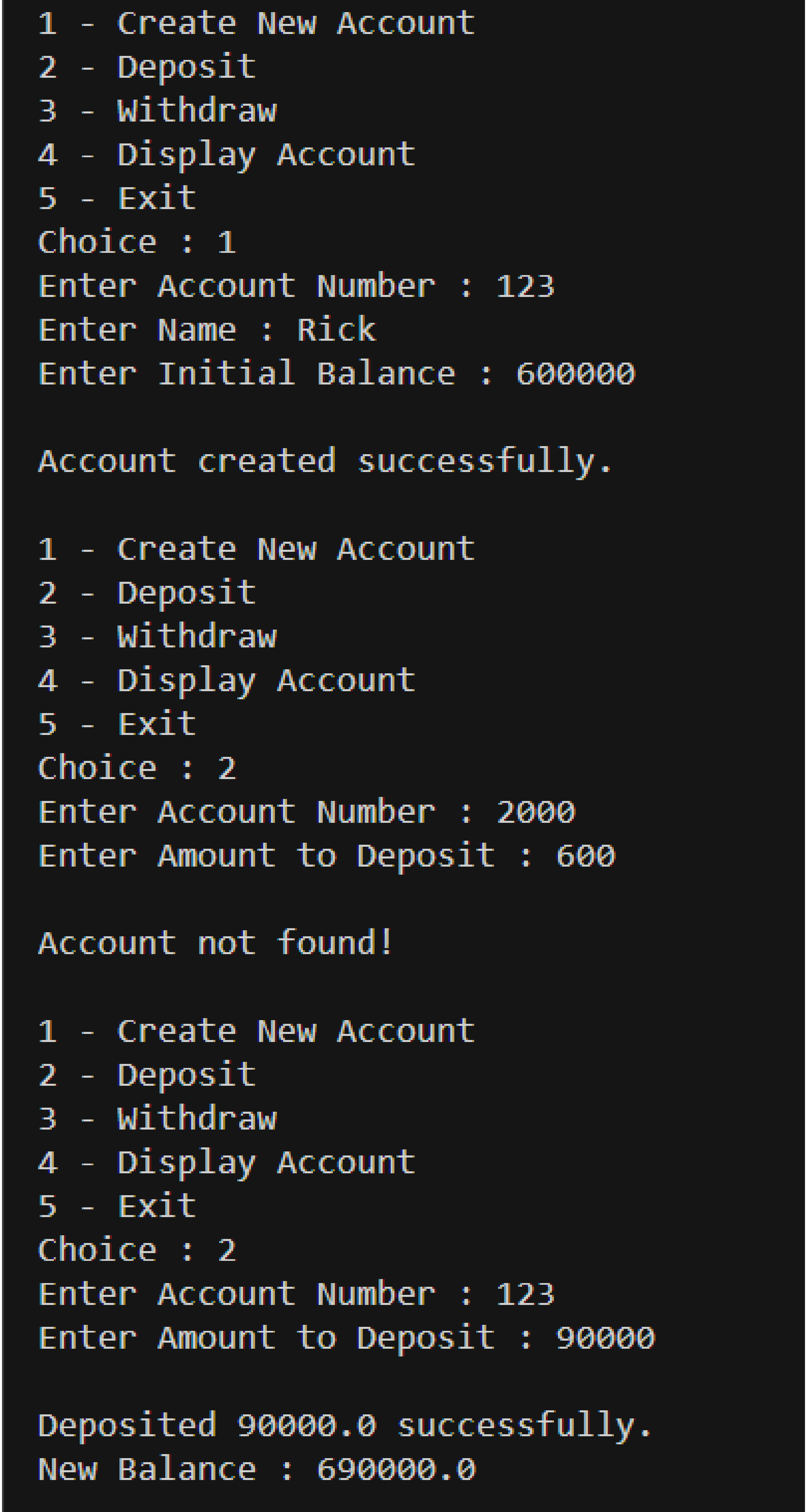
```
1 - Enqueue
2 - Dequeue
3 - Display Queue
4 - Exit
Choice : 1
Enter number to enqueue : 1
1 added to the queue.

1 - Enqueue
2 - Dequeue
3 - Display Queue
4 - Exit
Choice : 1
Enter number to enqueue : 2
2 added to the queue.

1 - Enqueue
2 - Dequeue
3 - Display Queue
4 - Exit
Choice : 1
Enter number to enqueue : 3
3 added to the queue.

1 - Enqueue
2 - Dequeue
3 - Display Queue
4 - Exit
Choice : 3
Queue: 1 <- 2 <- 3

1 - Enqueue
2 - Dequeue
3 - Display Queue
4 - Exit
Choice : 2
1 removed from the queue.

1 - Enqueue
2 - Dequeue
3 - Display Queue
4 - Exit
Choice : 3
Queue: 2 <- 3
```


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_3_bank_class.py > Bank > createacc

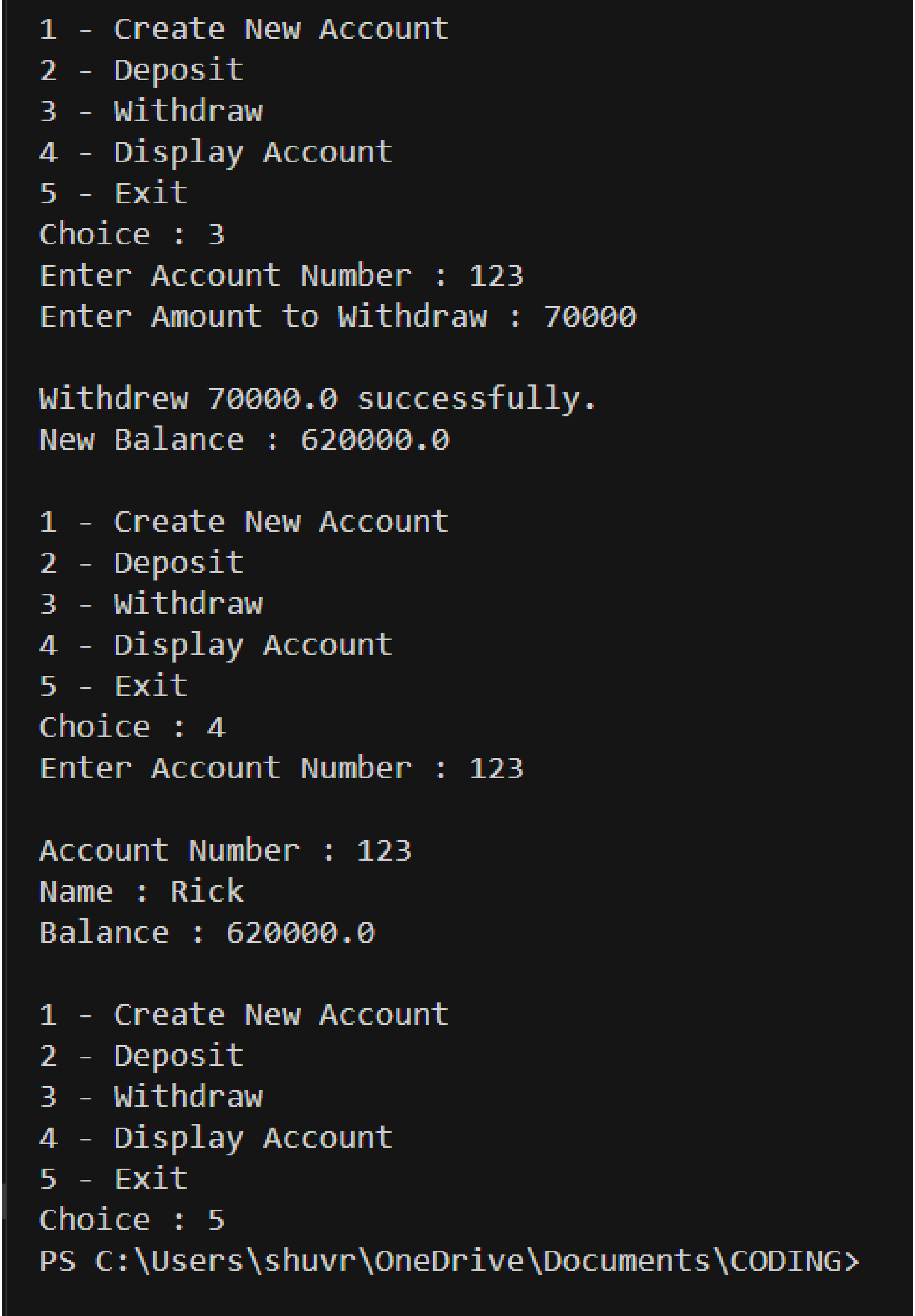
```python
""" Write a Python program to create a class representing a bank.
Include methods for managing customer accounts and transactions. """

class Bank:
    def __init__(self):
        self.accounts = {}

    def createacc(self, accno, name, bal=0):
        if accno in self.accounts:
            print("\nAccount already exists!")
        else:
            self.accounts[accno] = {'name': name, 'bal': bal}
            print("\nAccount created successfully.")

    def deposit(self, accno, amount):
        if accno in self.accounts:
            self.accounts[accno]['bal'] += amount
            print(f"\nDeposited {amount} successfully.\nNew Balance : {self.accounts[accno]['bal']}")
        else:
            print("\nAccount not found!")

    def withdraw(self, accno, amount):
        if accno in self.accounts and self.accounts[accno]['bal'] >= amount:
            self.accounts[accno]['bal'] -= amount
            print(f"\nWithdrew {amount} successfully.\nNew Balance : {self.accounts[accno]['bal']}")
        else:
            print("\nInsufficient funds or account not found!")

    def displayacc(self, accno):
        if accno in self.accounts:
            print(f"\nAccount Number : {accno}\nName : {self.accounts[accno]['name']}\nBalance : {self.accounts[accno]['bal']}")
        else:
            print("\nAccount not found!")


```



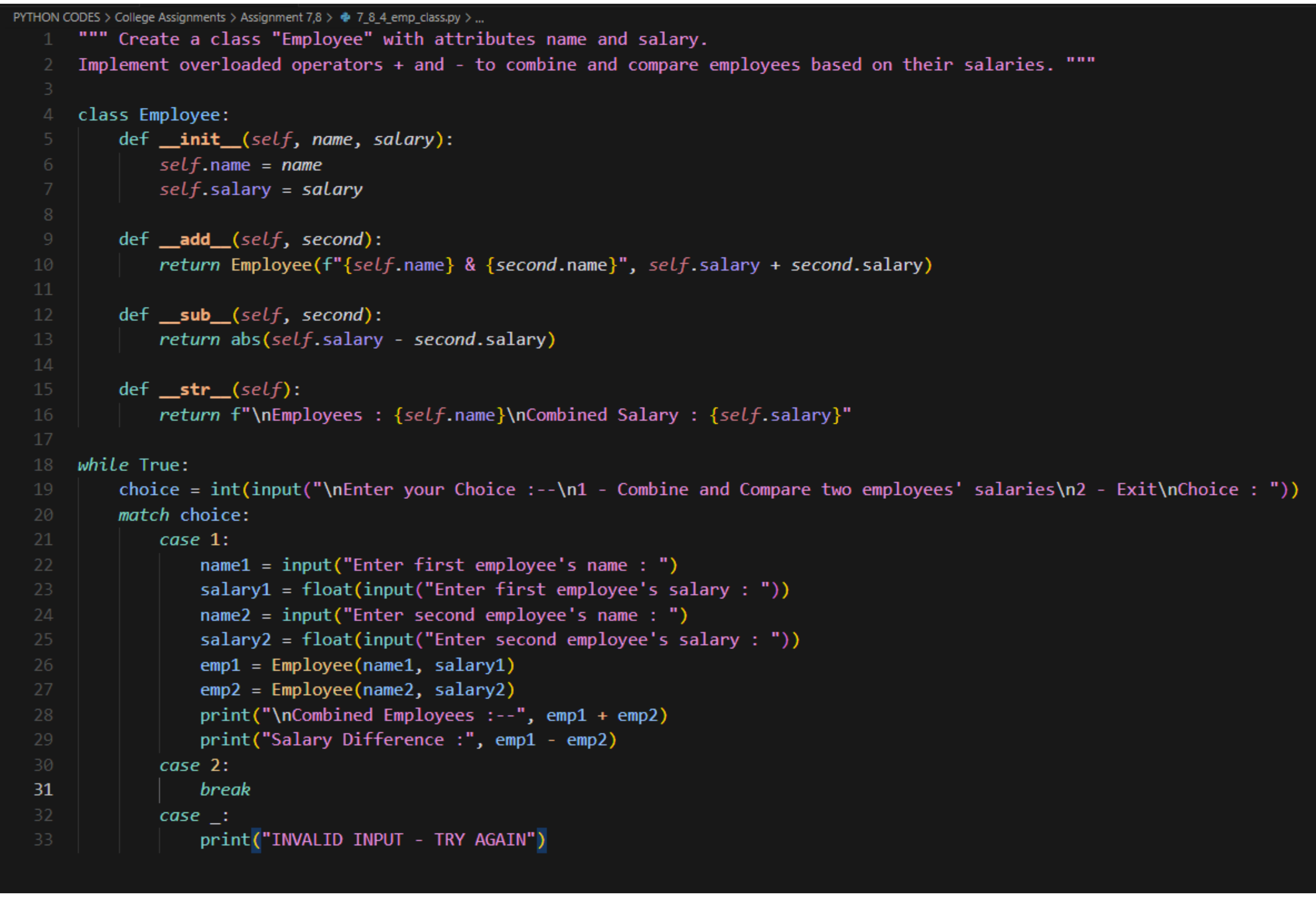
```python
bank = Bank()
while True:
    choice = int(input("\n1 - Create New Account\n2 - Deposit\n3 - Withdraw\n4 - Display Account\n5 - Exit\nChoice : "))
    match choice:
        case 1:
            accno = input("Enter Account Number : ")
            name = input("Enter Name : ")
            bal = float(input("Enter Initial Balance : "))
            bank.createacc(accno, name, bal)
        case 2:
            accno = input("Enter Account Number : ")
            amount = float(input("Enter Amount to Deposit : "))
            bank.deposit(accno, amount)
        case 3:
            accno = input("Enter Account Number : ")
            amount = float(input("Enter Amount to Withdraw : "))
            bank.withdraw(accno, amount)
        case 4:
            accno = input("Enter Account Number : ")
            bank.displayacc(accno)
        case 5:
            break
        case _:
            print("INVALID INPUT - TRY AGAIN.")
```



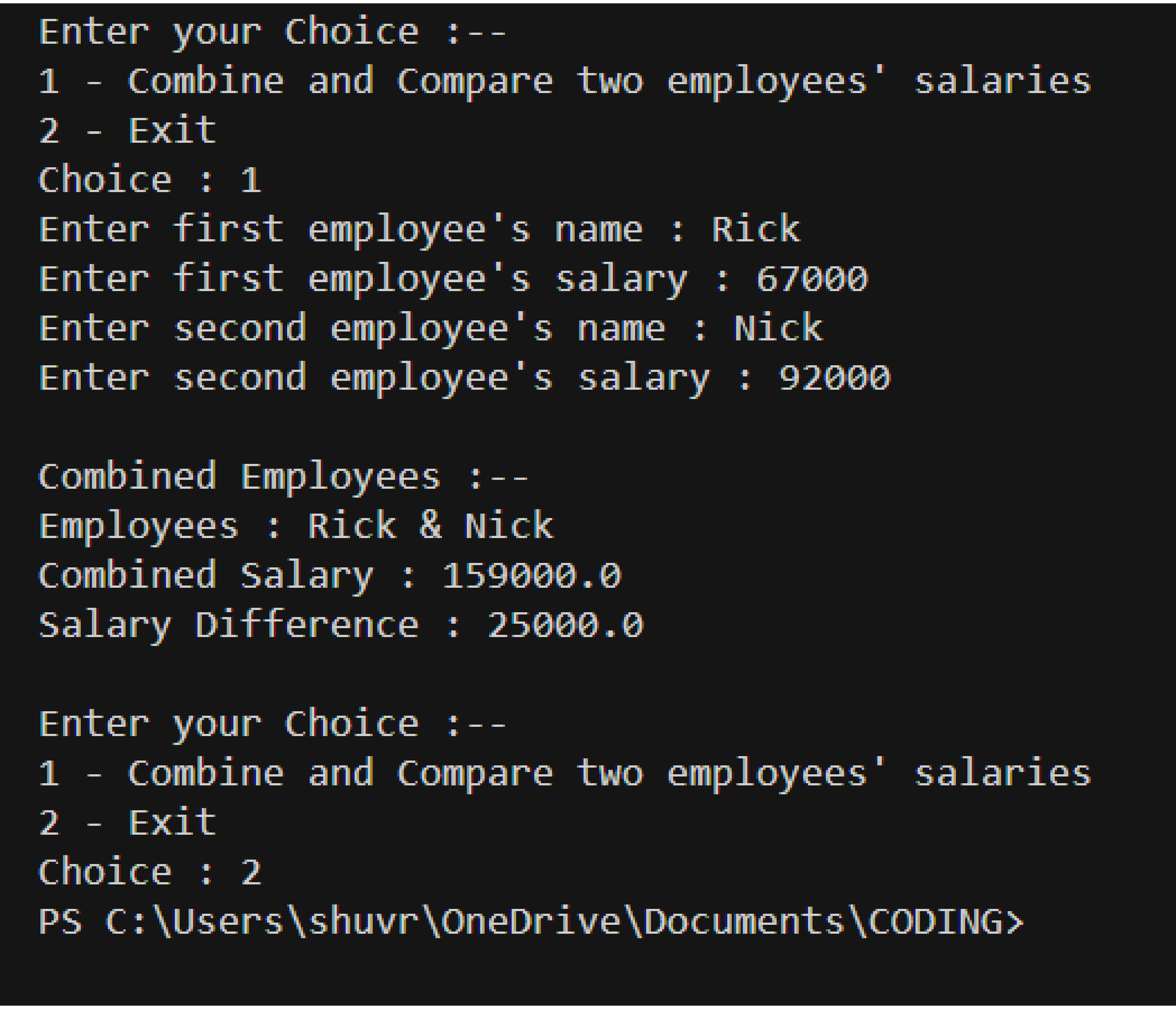
```
1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 1
Enter Account Number : 123
Enter Name : Rick
Enter Initial Balance : 600000

Account created successfully.

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 2
Enter Account Number : 2000
Enter Amount to Deposit : 600

Account not found!

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 2
Enter Account Number : 123
Enter Amount to Deposit : 90000

Deposited 90000.0 successfully.
New Balance : 690000.0
```



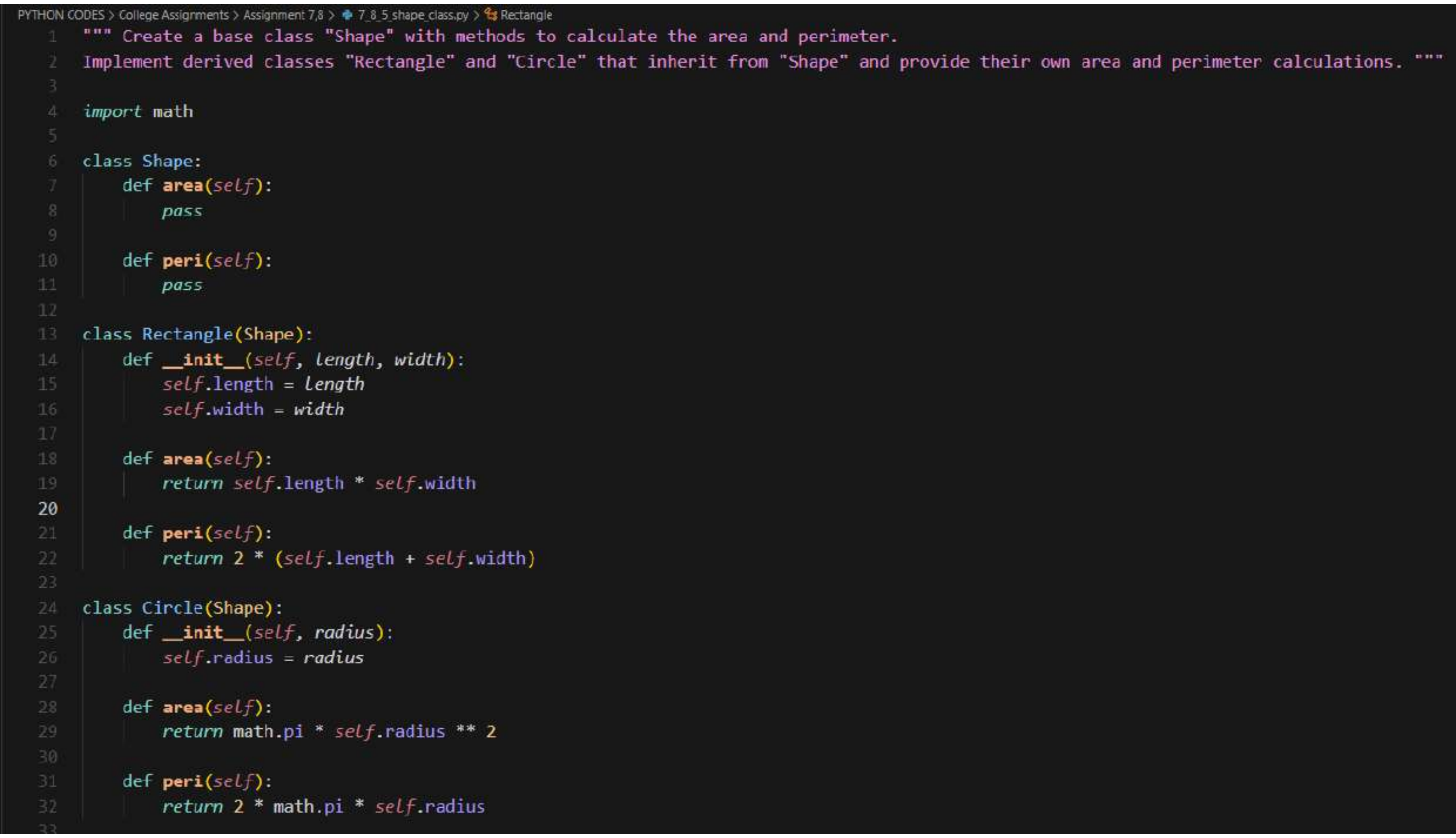
```
1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 3
Enter Account Number : 123
Enter Amount to Withdraw : 70000

Withdrew 70000.0 successfully.
New Balance : 620000.0

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 4
Enter Account Number : 123

Account Number : 123
Name : Rick
Balance : 620000.0

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 5
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_4_emp_class.py > ...

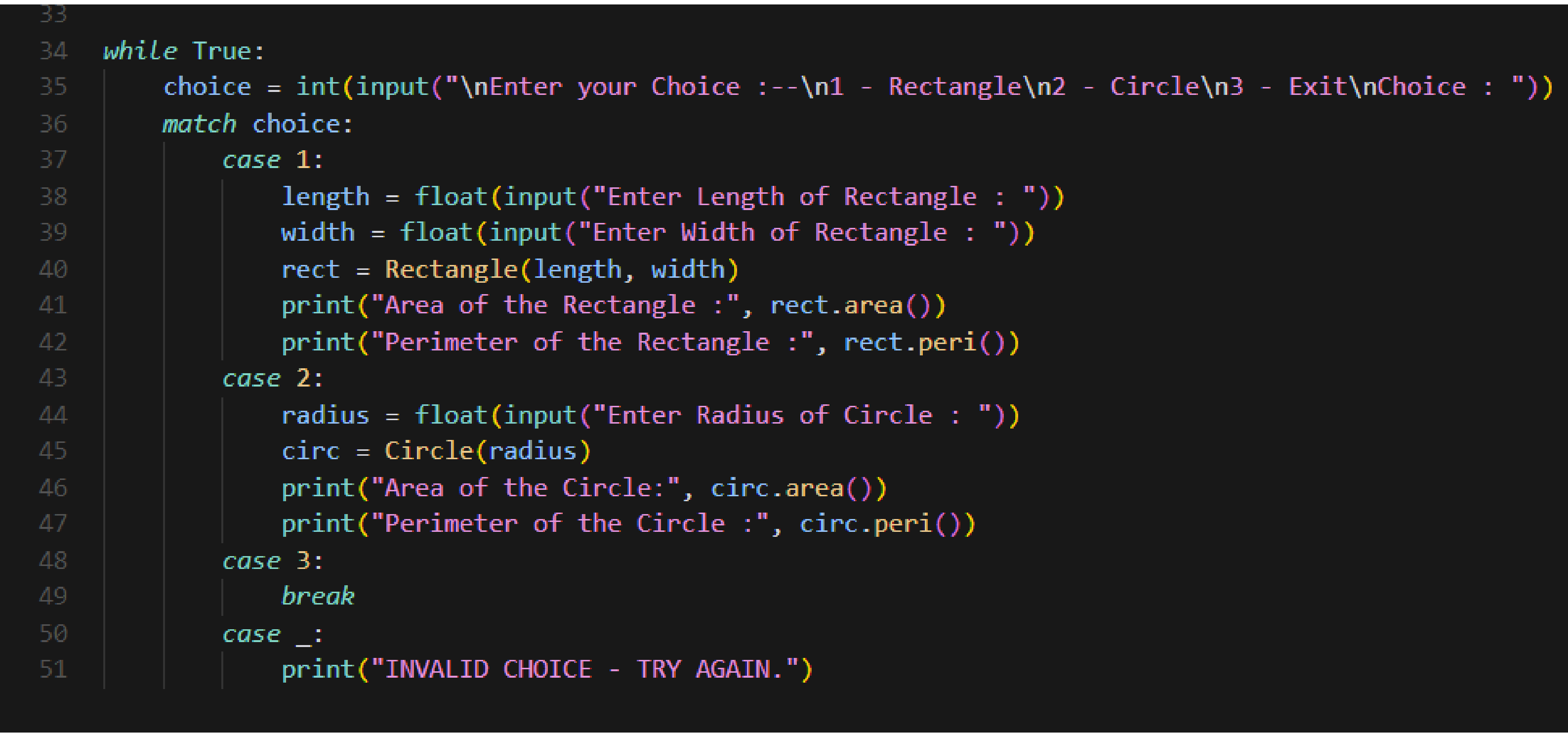
```python
""" Create a class "Employee" with attributes name and salary.
Implement overloaded operators + and - to combine and compare employees based on their salaries. """

class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def __add__(self, second):
        return Employee(f"{self.name} & {second.name}", self.salary + second.salary)

    def __sub__(self, second):
        return abs(self.salary - second.salary)

    def __str__(self):
        return f"\nEmployees : {self.name}\nCombined Salary : {self.salary}"

while True:
    choice = int(input("\nEnter your Choice :--\n1 - Combine and Compare two employees' salaries\n2 - Exit\nChoice : "))
    match choice:
        case 1:
            name1 = input("Enter first employee's name : ")
            salary1 = float(input("Enter first employee's salary : "))
            name2 = input("Enter second employee's name : ")
            salary2 = float(input("Enter second employee's salary : "))
            emp1 = Employee(name1, salary1)
            emp2 = Employee(name2, salary2)
            print("\nCombined Employees :--", emp1 + emp2)
            print("Salary Difference :", emp1 - emp2)
        case 2:
            break
        case _:
            print("INVALID INPUT - TRY AGAIN")
```



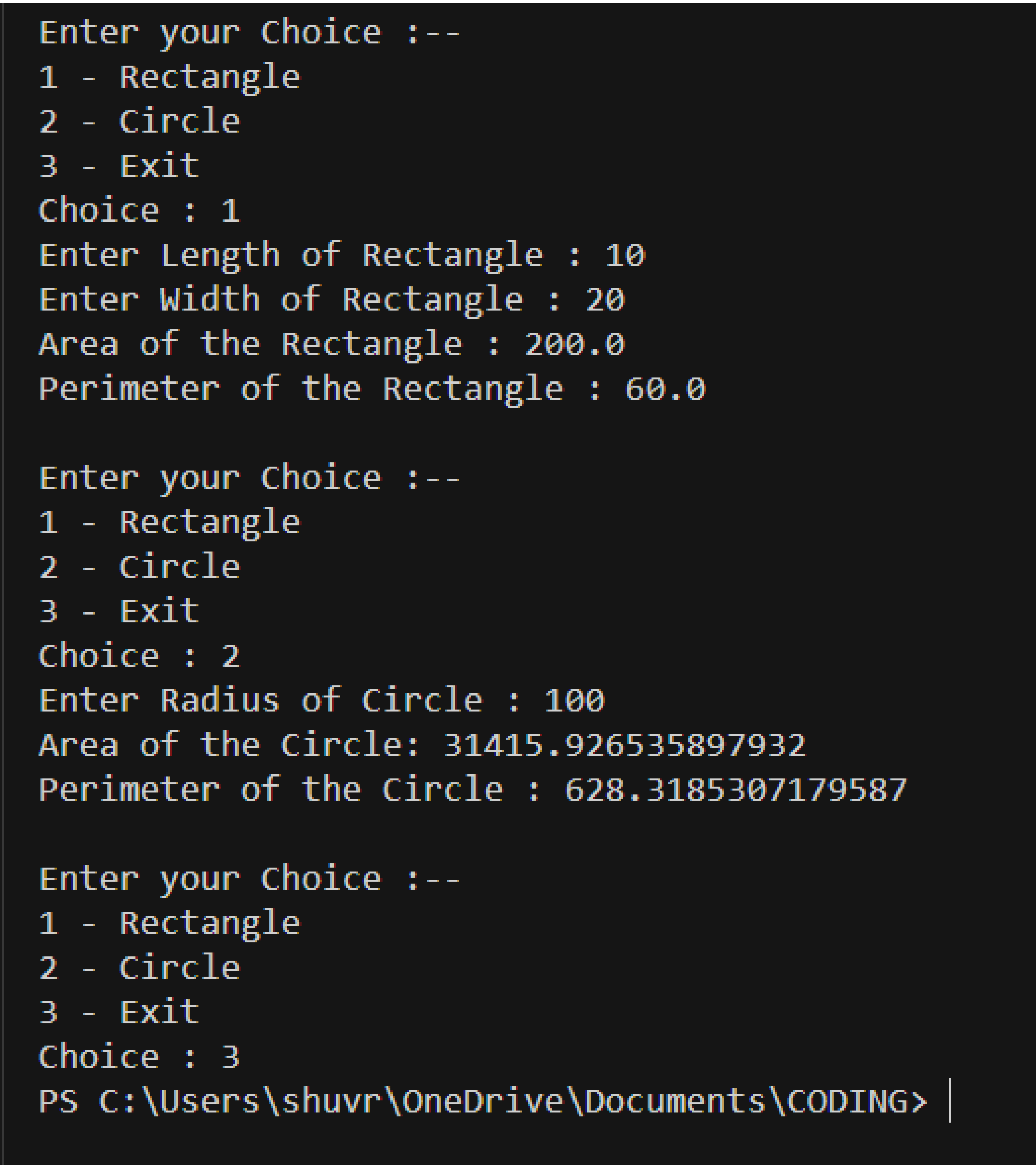
```
Enter your Choice :--
1 - Combine and Compare two employees' salaries
2 - Exit
Choice : 1
Enter first employee's name : Rick
Enter first employee's salary : 67000
Enter second employee's name : Nick
Enter second employee's salary : 92000

Combined Employees :--
Employees : Rick & Nick
Combined Salary : 159000.0
Salary Difference : 25000.0

Enter your Choice :--
1 - Combine and Compare two employees' salaries
2 - Exit
Choice : 2
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_5_shape_class.py > Rectangle

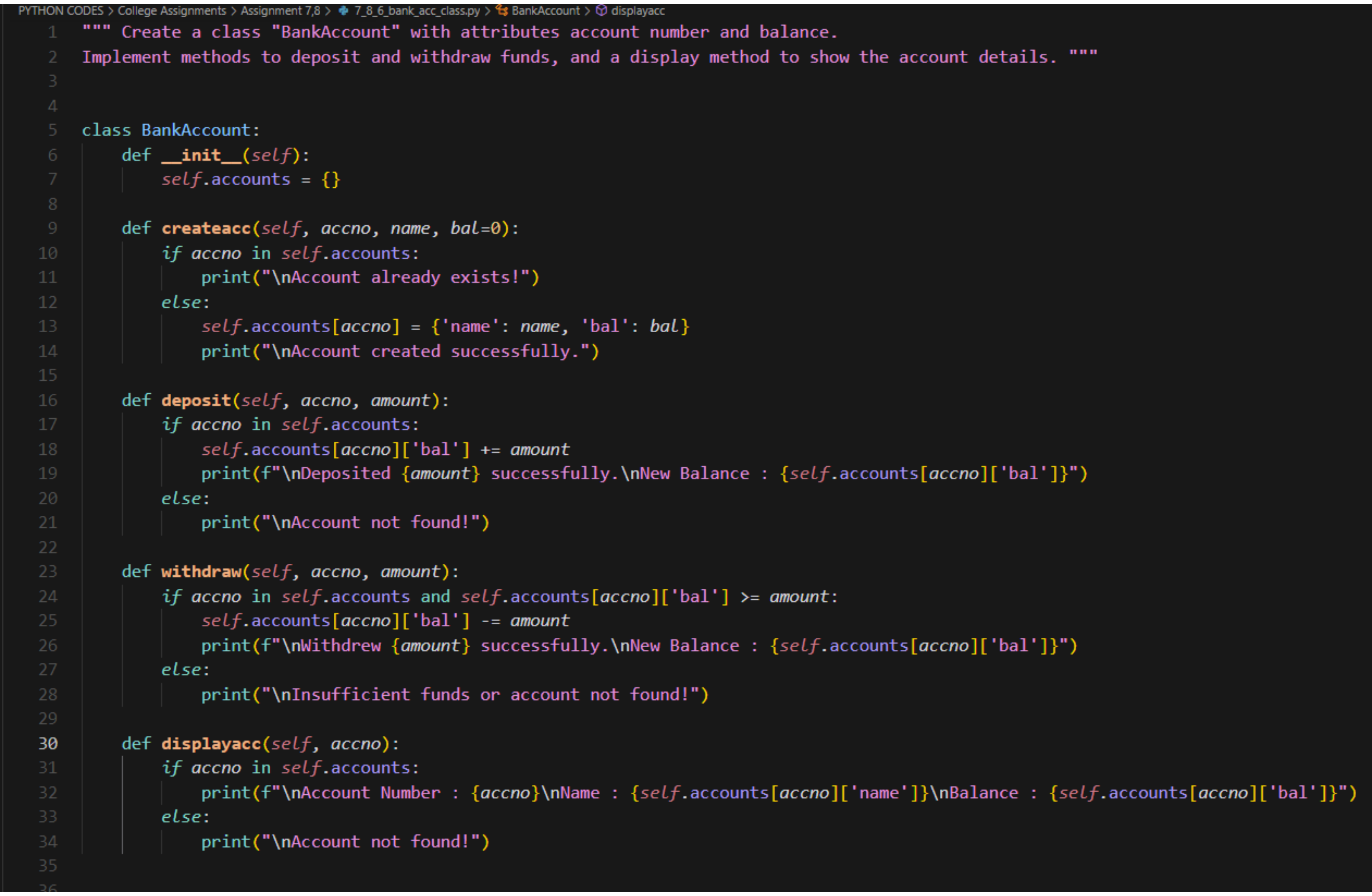
```python
""" Create a base class "Shape" with methods to calculate the area and perimeter.
Implement derived classes "Rectangle" and "Circle" that inherit from "Shape" and provide their own area and perimeter calculations. """

import math

class Shape:
    def area(self):
        pass

    def peri(self):
        pass

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def peri(self):
        return 2 * (self.length + self.width)

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

    def peri(self):
        return 2 * math.pi * self.radius
```



```
while True:
    choice = int(input("\nEnter your Choice :--\n1 - Rectangle\n2 - Circle\n3 - Exit\nChoice : "))
    match choice:
        case 1:
            length = float(input("Enter Length of Rectangle : "))
            width = float(input("Enter Width of Rectangle : "))
            rect = Rectangle(length, width)
            print("Area of the Rectangle :", rect.area())
            print("Perimeter of the Rectangle :", rect.peri())
        case 2:
            radius = float(input("Enter Radius of Circle : "))
            circ = Circle(radius)
            print("Area of the Circle:", circ.area())
            print("Perimeter of the Circle :", circ.peri())
        case 3:
            break
        case _:
            print("INVALID CHOICE - TRY AGAIN.")
```

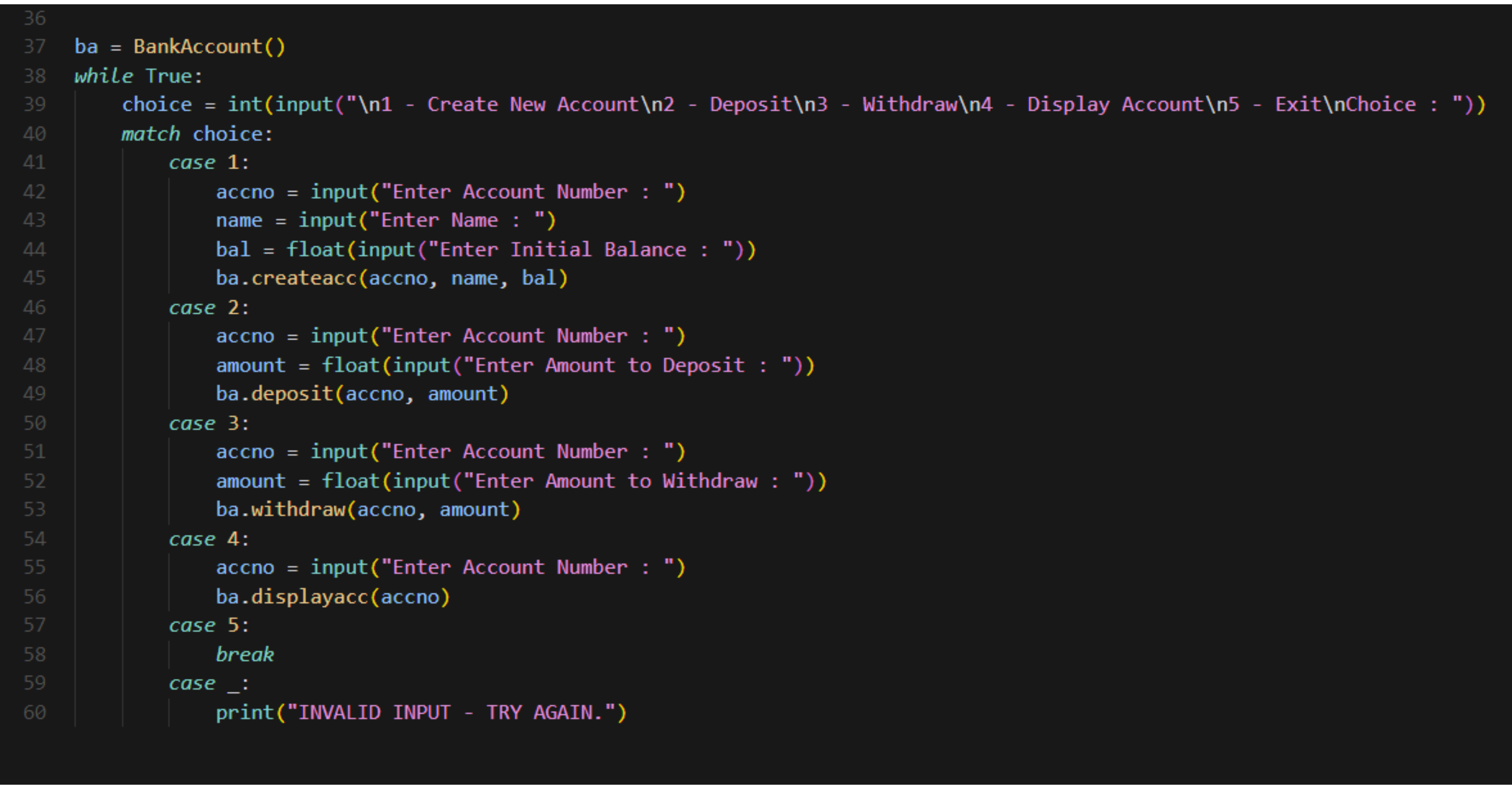


```
Enter your Choice :--
1 - Rectangle
2 - Circle
3 - Exit
Choice : 1
Enter Length of Rectangle : 10
Enter Width of Rectangle : 20
Area of the Rectangle : 200.0
Perimeter of the Rectangle : 60.0

Enter your Choice :--
1 - Rectangle
2 - Circle
3 - Exit
Choice : 2
Enter Radius of Circle : 100
Area of the Circle: 31415.926535897932
Perimeter of the Circle : 628.3185307179587

Enter your Choice :--
1 - Rectangle
2 - Circle
3 - Exit
Choice : 3
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```

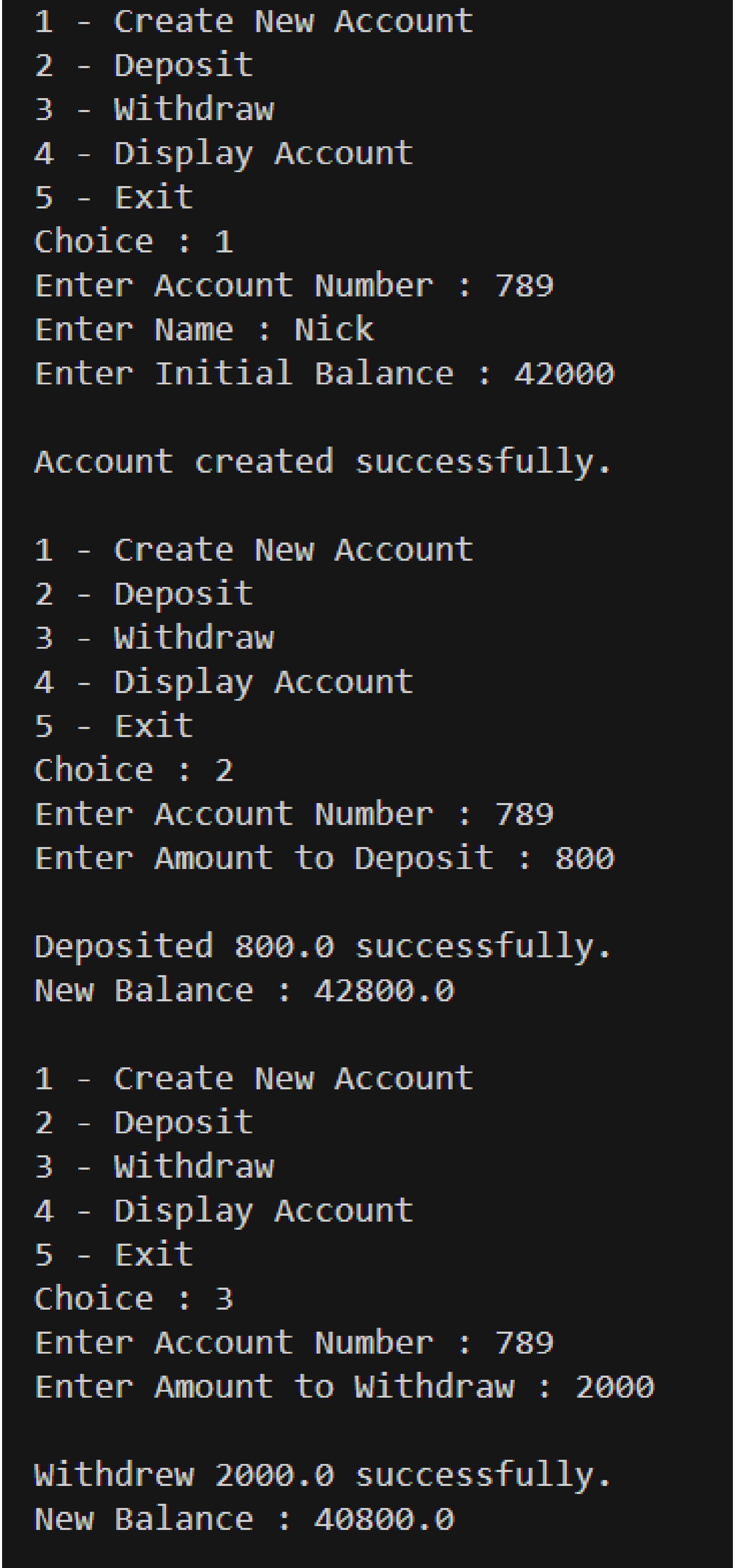


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_6_bank_acc_class.py > BankAccount > displayacc

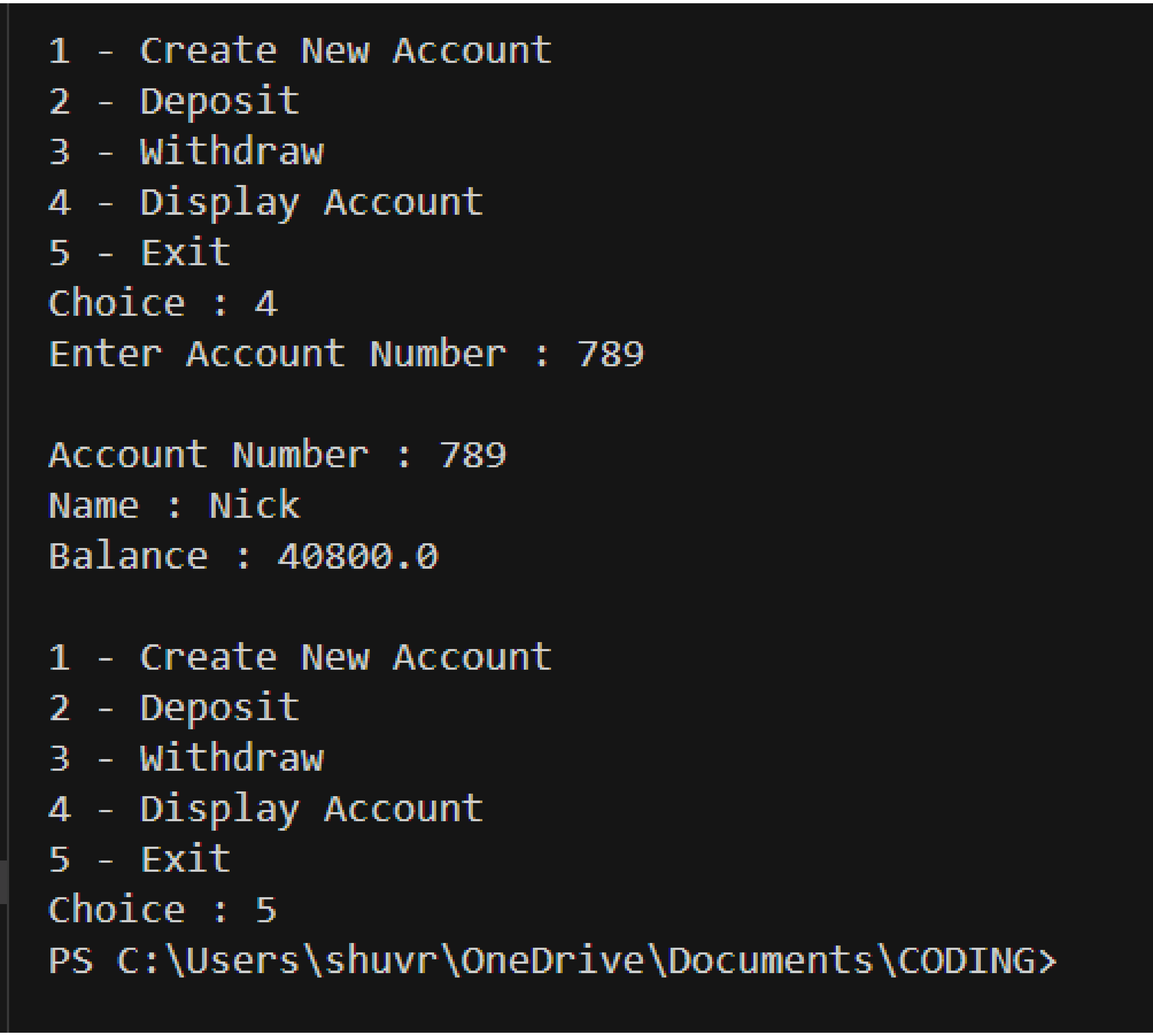
```python
""" Create a class "BankAccount" with attributes account number and balance.
Implement methods to deposit and withdraw funds, and a display method to show the account details. """


class BankAccount:
    def __init__(self):
        self.accounts = {}

    def createacc(self, accno, name, bal=0):
        if accno in self.accounts:
            print("\nAccount already exists!")
        else:
            self.accounts[accno] = {'name': name, 'bal': bal}
            print("\nAccount created successfully.")

    def deposit(self, accno, amount):
        if accno in self.accounts:
            self.accounts[accno]['bal'] += amount
            print(f"\nDeposited {amount} successfully.\nNew Balance : {self.accounts[accno]['bal']}")
        else:
            print("\nAccount not found!")

    def withdraw(self, accno, amount):
        if accno in self.accounts and self.accounts[accno]['bal'] >= amount:
            self.accounts[accno]['bal'] -= amount
            print(f"\nWithdrew {amount} successfully.\nNew Balance : {self.accounts[accno]['bal']}")
        else:
            print("\nInsufficient funds or account not found!")

    def displayacc(self, accno):
        if accno in self.accounts:
            print(f"\nAccount Number : {accno}\nName : {self.accounts[accno]['name']}\nBalance : {self.accounts[accno]['bal']}")
        else:
            print("\nAccount not found!")
```



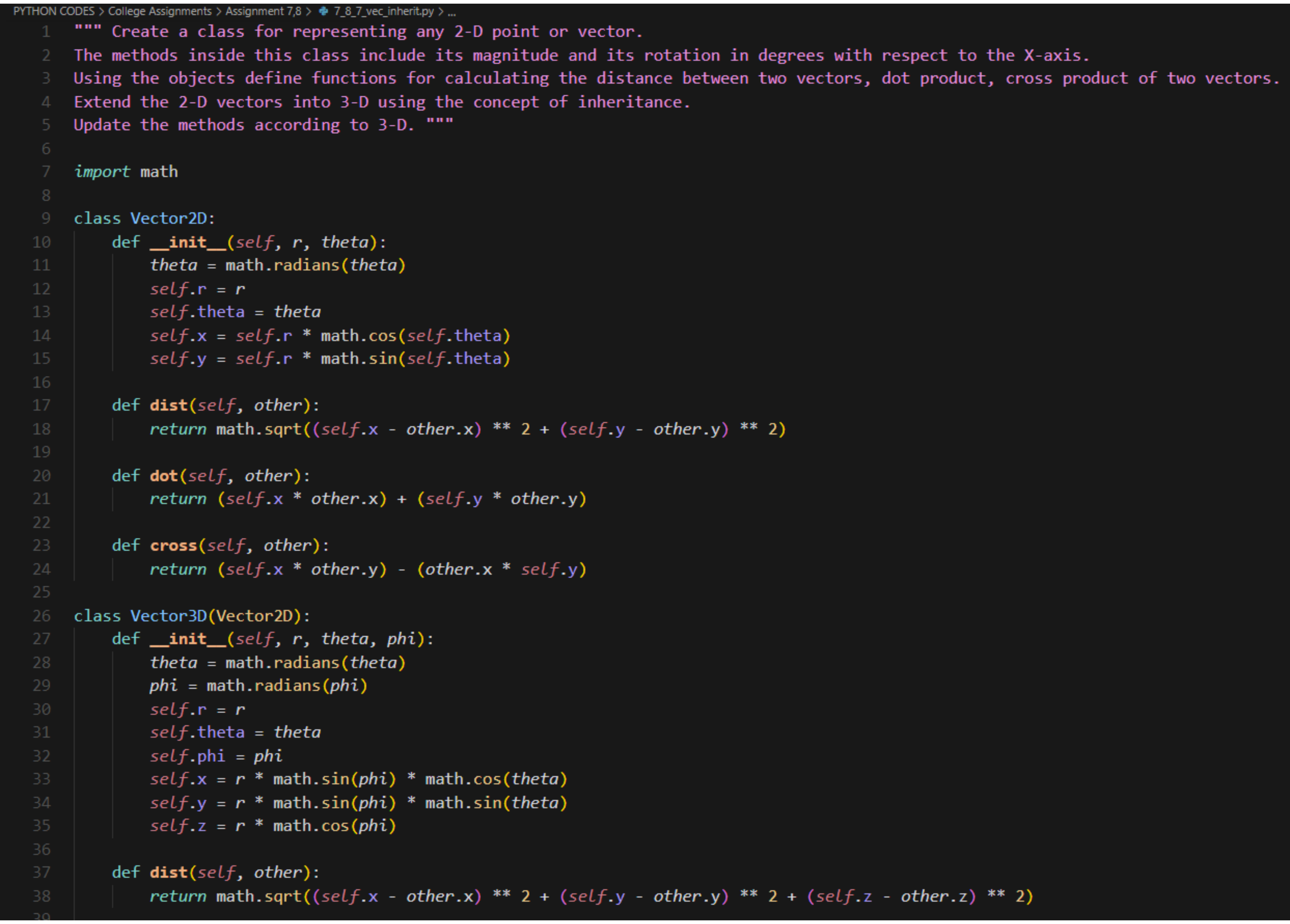
```python
ba = BankAccount()
while True:
    choice = int(input("\n1 - Create New Account\n2 - Deposit\n3 - Withdraw\n4 - Display Account\n5 - Exit\nChoice : "))
    match choice:
        case 1:
            accno = input("Enter Account Number : ")
            name = input("Enter Name : ")
            bal = float(input("Enter Initial Balance : "))
            ba.createacc(accno, name, bal)
        case 2:
            accno = input("Enter Account Number : ")
            amount = float(input("Enter Amount to Deposit : "))
            ba.deposit(accno, amount)
        case 3:
            accno = input("Enter Account Number : ")
            amount = float(input("Enter Amount to Withdraw : "))
            ba.withdraw(accno, amount)
        case 4:
            accno = input("Enter Account Number : ")
            ba.displayacc(accno)
        case 5:
            break
        case _:
            print("INVALID INPUT - TRY AGAIN.")
```



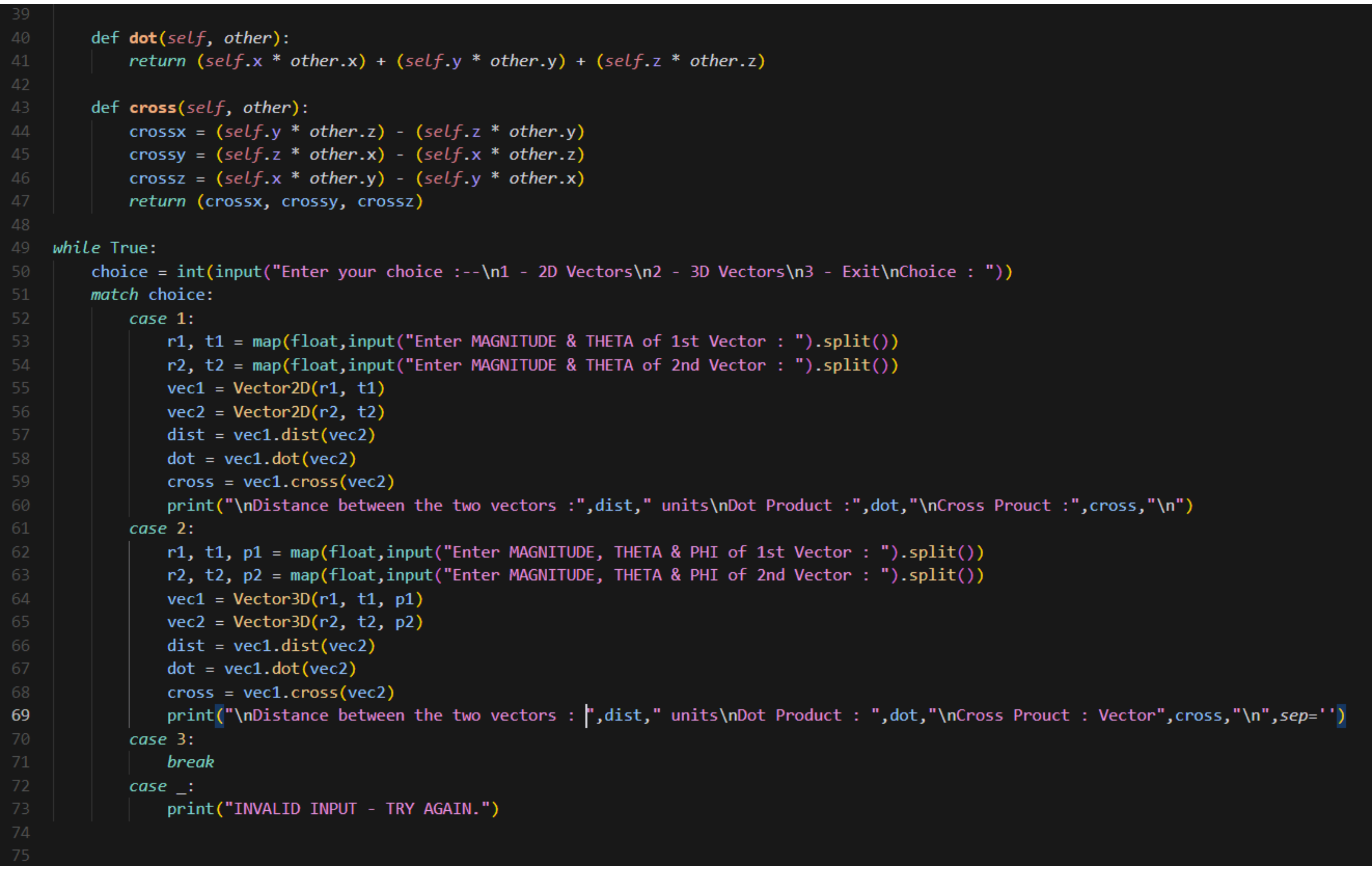
```
1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 1
Enter Account Number : 789
Enter Name : Nick
Enter Initial Balance : 42000

Account created successfully.

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 2
Enter Account Number : 789
Enter Amount to Deposit : 800

Deposited 800.0 successfully.
New Balance : 42800.0

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 3
Enter Account Number : 789
Enter Amount to Withdraw : 2000

Withdrew 2000.0 successfully.
New Balance : 40800.0
```



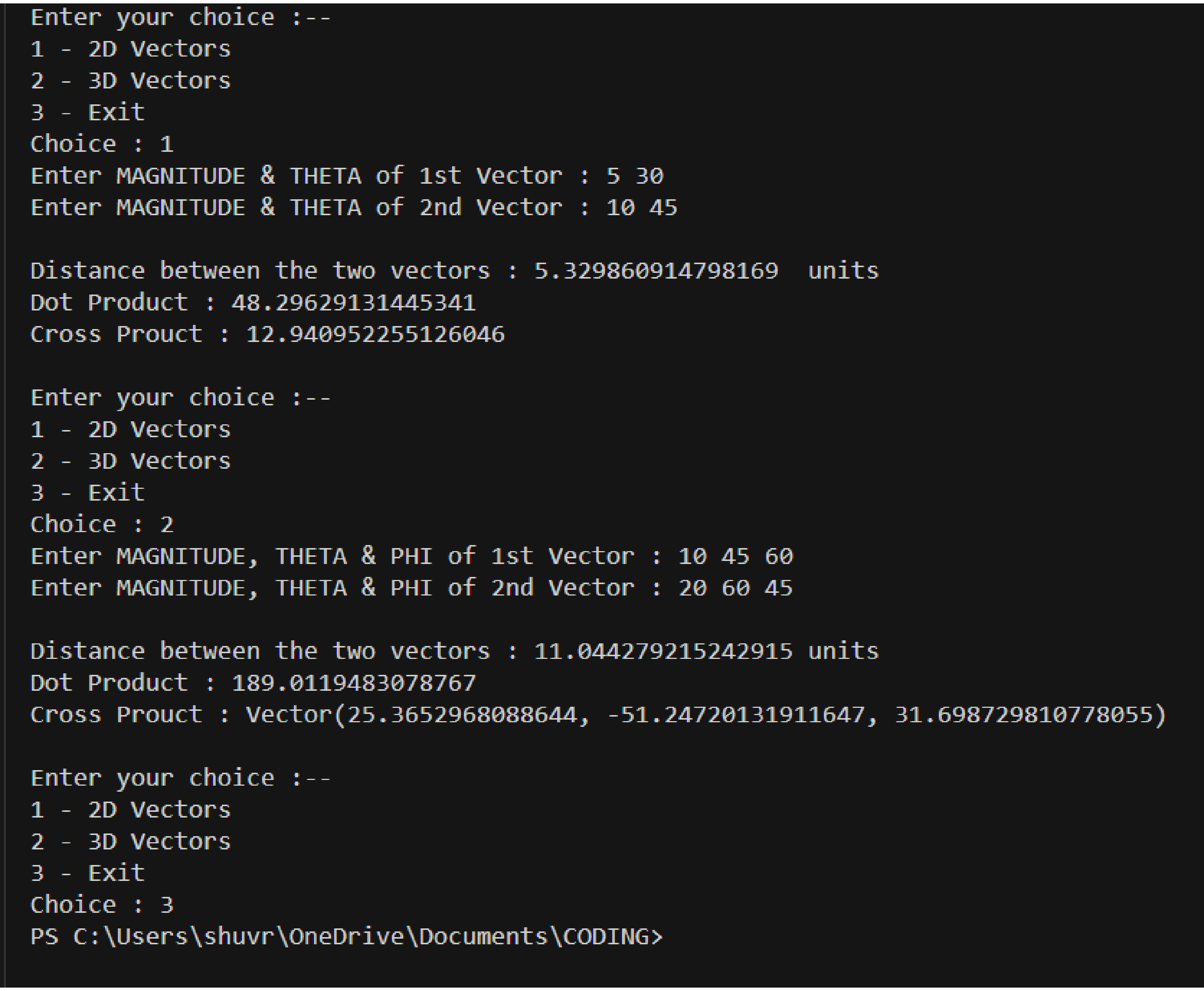
```
1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 4
Enter Account Number : 789

Account Number : 789
Name : Nick
Balance : 40800.0

1 - Create New Account
2 - Deposit
3 - Withdraw
4 - Display Account
5 - Exit
Choice : 5
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_7_vec_inherit.py > ...

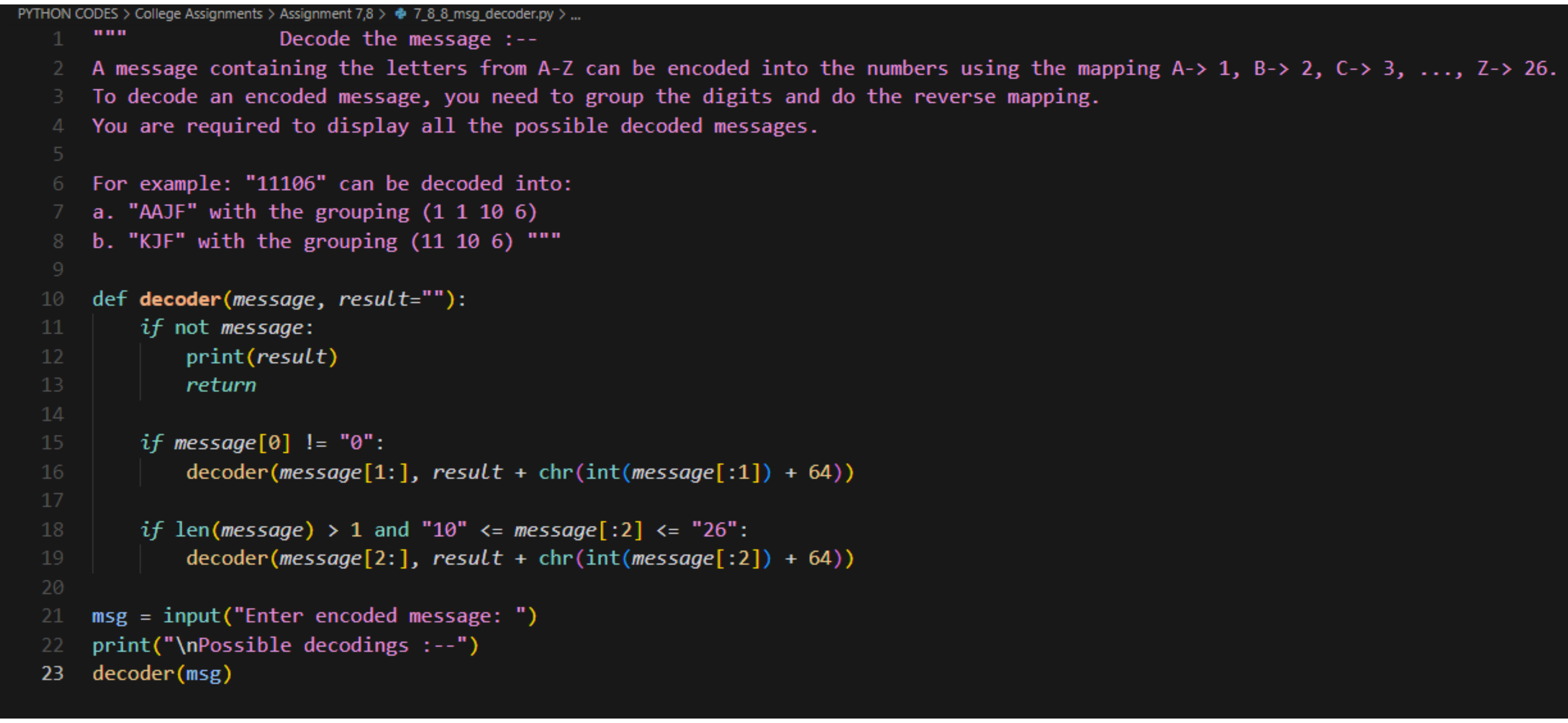
```python
""" Create a class for representing any 2-D point or vector.
The methods inside this class include its magnitude and its rotation in degrees with respect to the X-axis.
Using the objects define functions for calculating the distance between two vectors, dot product, cross product of two vectors.
Extend the 2-D vectors into 3-D using the concept of inheritance.
Update the methods according to 3-D. """

import math

class Vector2D:
    def __init__(self, r, theta):
        theta = math.radians(theta)
        self.r = r
        self.theta = theta
        self.x = self.r * math.cos(self.theta)
        self.y = self.r * math.sin(self.theta)

    def dist(self, other):
        return math.sqrt((self.x - other.x) ** 2 + (self.y - other.y) ** 2)

    def dot(self, other):
        return (self.x * other.x) + (self.y * other.y)

    def cross(self, other):
        return (self.x * other.y) - (other.x * self.y)

class Vector3D(Vector2D):
    def __init__(self, r, theta, phi):
        theta = math.radians(theta)
        phi = math.radians(phi)
        self.r = r
        self.theta = theta
        self.phi = phi
        self.x = r * math.sin(phi) * math.cos(theta)
        self.y = r * math.sin(phi) * math.sin(theta)
        self.z = r * math.cos(phi)

    def dist(self, other):
        return math.sqrt((self.x - other.x) ** 2 + (self.y - other.y) ** 2 + (self.z - other.z) ** 2)
```



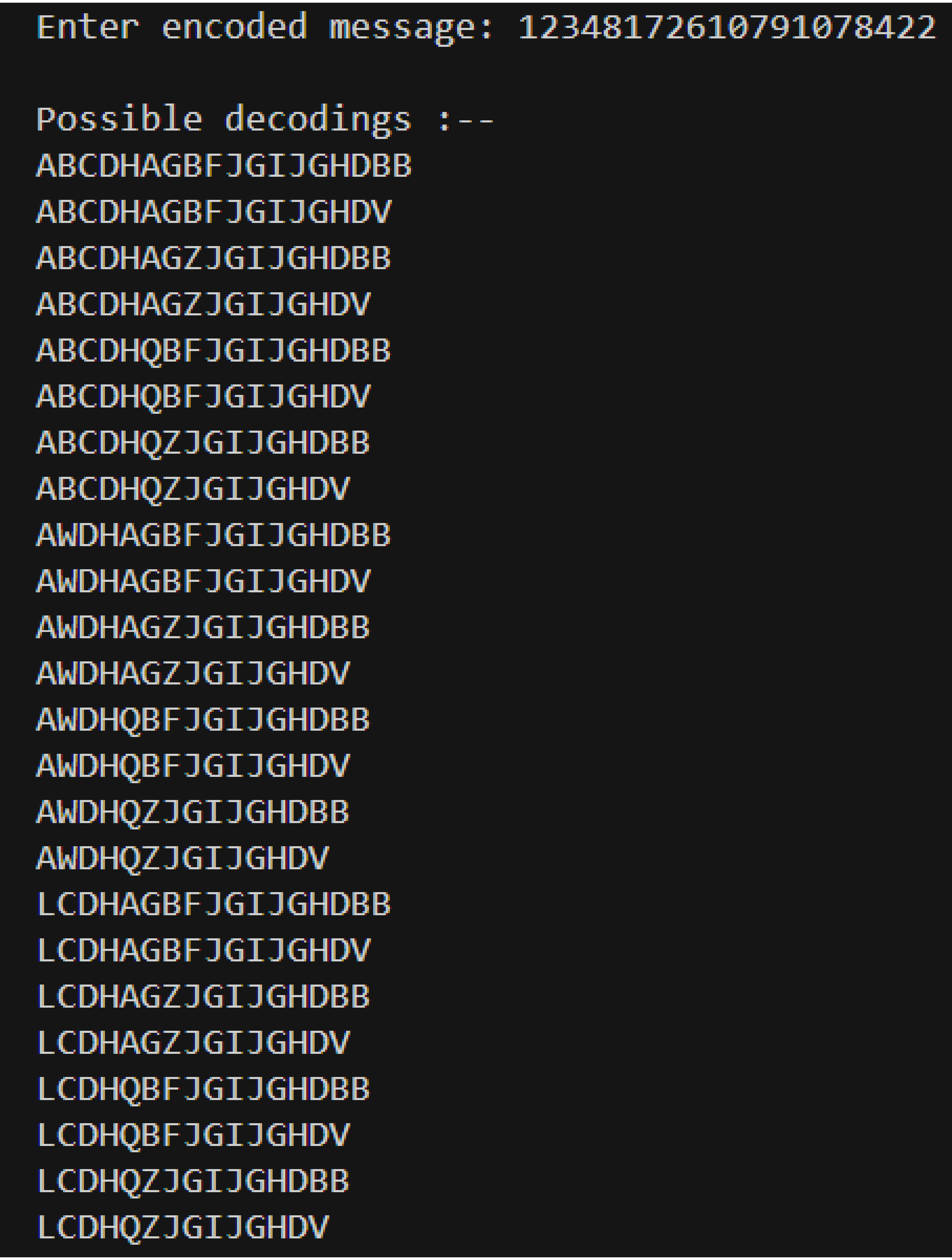
```python

    def dot(self, other):
        return (self.x * other.x) + (self.y * other.y) + (self.z * other.z)

    def cross(self, other):
        crossx = (self.y * other.z) - (self.z * other.y)
        crossy = (self.z * other.x) - (self.x * other.z)
        crossz = (self.x * other.y) - (self.y * other.x)
        return (crossx, crossy, crossz)

while True:
    choice = int(input("Enter your choice :--\n1 - 2D Vectors\n2 - 3D Vectors\n3 - Exit\nChoice : "))
    match choice:
        case 1:
            r1, t1 = map(float,input("Enter MAGNITUDE & THETA of 1st Vector : ").split())
            r2, t2 = map(float,input("Enter MAGNITUDE & THETA of 2nd Vector : ").split())
            vec1 = Vector2D(r1, t1)
            vec2 = Vector2D(r2, t2)
            dist = vec1.dist(vec2)
            dot = vec1.dot(vec2)
            cross = vec1.cross(vec2)
            print("\nDistance between the two vectors :",dist," units\nDot Product :",dot,"\nCross Prouct :",cross,"\n")
        case 2:
            r1, t1, p1 = map(float,input("Enter MAGNITUDE, THETA & PHI of 1st Vector : ").split())
            r2, t2, p2 = map(float,input("Enter MAGNITUDE, THETA & PHI of 2nd Vector : ").split())
            vec1 = Vector3D(r1, t1, p1)
            vec2 = Vector3D(r2, t2, p2)
            dist = vec1.dist(vec2)
            dot = vec1.dot(vec2)
            cross = vec1.cross(vec2)
            print("\nDistance between the two vectors : ",dist," units\nDot Product : ",dot,"\nCross Prouct : Vector",cross,"\n",sep='')
        case 3:
            break
        case _:
            print("INVALID INPUT - TRY AGAIN.")


```



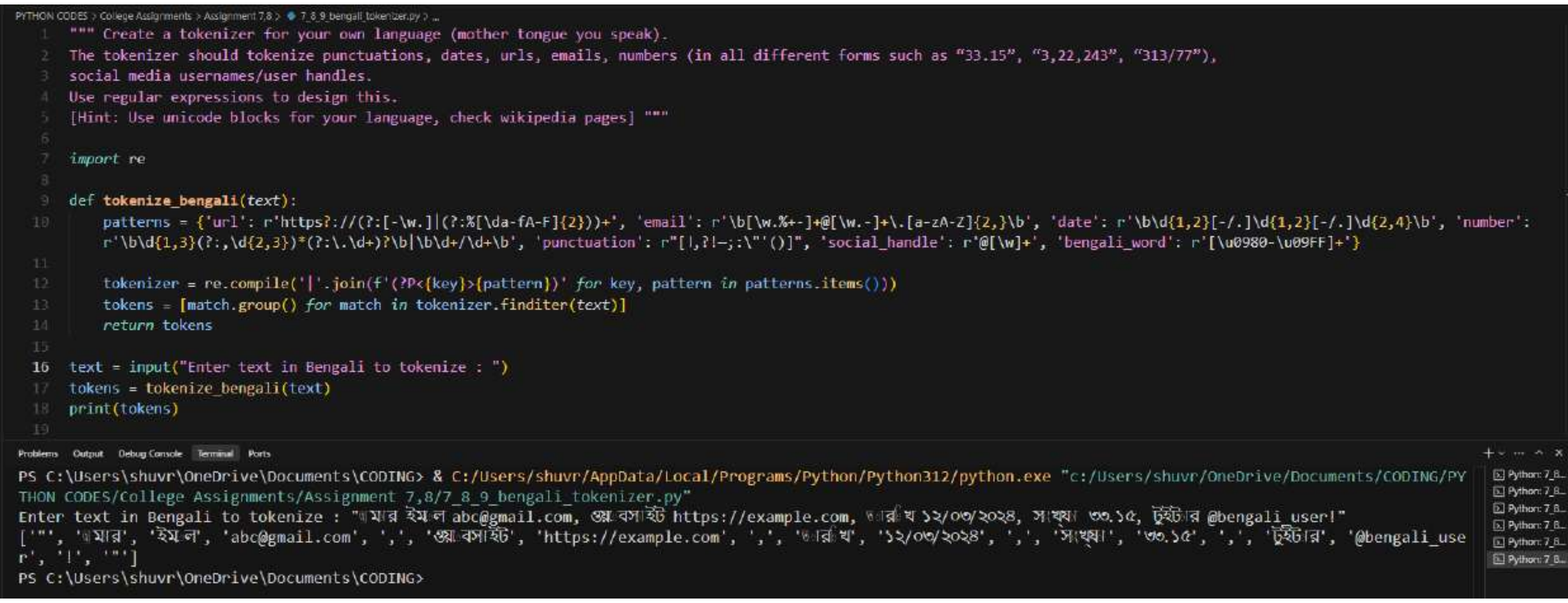
```
Enter your choice :--
1 - 2D Vectors
2 - 3D Vectors
3 - Exit
Choice : 1
Enter MAGNITUDE & THETA of 1st Vector : 5 30
Enter MAGNITUDE & THETA of 2nd Vector : 10 45

Distance between the two vectors : 5.329860914798169  units
Dot Product : 48.29629131445341
Cross Prouct : 12.940952255126046

Enter your choice :--
1 - 2D Vectors
2 - 3D Vectors
3 - Exit
Choice : 2
Enter MAGNITUDE, THETA & PHI of 1st Vector : 10 45 60
Enter MAGNITUDE, THETA & PHI of 2nd Vector : 20 60 45

Distance between the two vectors : 11.044279215242915 units
Dot Product : 189.0119483078767
Cross Prouct : Vector(25.3652968088644, -51.24720131911647, 31.698729810778055)

Enter your choice :--
1 - 2D Vectors
2 - 3D Vectors
3 - Exit
Choice : 3
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```


PYTHON CODES > College Assignments > Assignment 7,8 > 7_8_8_msg_decoder.py > ...

```python
"""             Decode the message :--
A message containing the letters from A-Z can be encoded into the numbers using the mapping A-> 1, B-> 2, C-> 3, ..., Z-> 26.
To decode an encoded message, you need to group the digits and do the reverse mapping.
You are required to display all the possible decoded messages.

For example: "11106" can be decoded into:
a. "AAJF" with the grouping (1 1 10 6)
b. "KJF" with the grouping (11 10 6) """

def decoder(message, result=""):
    if not message:
        print(result)
        return

    if message[0] != "0":
        decoder(message[1:], result + chr(int(message[:1]) + 64))

    if len(message) > 1 and "10" <= message[:2] <= "26":
        decoder(message[2:], result + chr(int(message[:2]) + 64))

msg = input("Enter encoded message: ")
print("\nPossible decodings :--")
decoder(msg)
```



```
Enter encoded message: 12348172610791078422

Possible decodings :--
ABCDHAGBFJGIJGHDBB
ABCDHAGBFJGIJGHDV
ABCDHAGZJGIJGHDBB
ABCDHAGZJGIJGHDV
ABCDHQBFJGIJGHDBB
ABCDHQBFJGIJGHDV
ABCDHQZJGIJGHDBB
ABCDHQZJGIJGHDV
AWDHAGBFJGIJGHDBB
AWDHAGBFJGIJGHDV
AWDHAGZJGIJGHDBB
AWDHAGZJGIJGHDV
AWDHQBFJGIJGHDBB
AWDHQBFJGIJGHDV
AWDHQZJGIJGHDBB
AWDHQZJGIJGHDV
LCDHAGBFJGIJGHDBB
LCDHAGBFJGIJGHDV
LCDHAGZJGIJGHDBB
LCDHAGZJGIJGHDV
LCDHQBFJGIJGHDBB
LCDHQBFJGIJGHDV
LCDHQZJGIJGHDBB
LCDHQZJGIJGHDV
```



```python
""" Create a tokenizer for your own language (mother tongue you speak).
The tokenizer should tokenize punctuations, dates, urls, emails, numbers (in all different forms such as “33.15”, “3,22,243”, “313/77”),
social media usernames/user handles.
Use regular expressions to design this.
[Hint: Use unicode blocks for your language, check wikipedia pages] """

import re

def tokenize_bengali(text):
    patterns = {'url': r'https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+', 'email': r'\b[\w.%+-]+@[\w.-]+\.[a-zA-Z]{2,}\b', 'date': r'\b\d{1,2}[-/.]\d{1,2}[-/.]\d{2,4}\b', 'number': 
    r'\b\d{1,3}(?:,\d{2,3})*(?:\.\d+)?\b|\b\d+/\d+\b', 'punctuation': r"[।,?!—;:\"'()]", 'social_handle': r'@[\w]+', 'bengali_word': r'[\u0980-\u09FF]+'}

    tokenizer = re.compile('|'.join(f'(?P<{key}>{pattern})' for key, pattern in patterns.items()))
    tokens = [match.group() for match in tokenizer.finditer(text)]
    return tokens

text = input("Enter text in Bengali to tokenize : ")
tokens = tokenize_bengali(text)
print(tokens)
```

```
PS C:\Users\shuvr\OneDrive\Documents\CODING> & C:/Users/shuvr/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/shuvr/OneDrive/Documents/CODING/PY
THON CODES/College Assignments/Assignment 7,8/7_8_9_bengali_tokenizer.py"
Enter text in Bengali to tokenize : "আমার ইমেল abc@gmail.com, ওয়েবসাইট https://example.com, তারিখ ১২/০৩/২০২৪, সংখ্যা ৩৩.১৫, টুইটার @bengali_user!"
['"', 'আমার', 'ইমেল', 'abc@gmail.com', ',', 'ওয়েবসাইট', 'https://example.com', ',', 'তারিখ', '১২/০৩/২০২৪', ',', 'সংখ্যা', '৩৩.১৫', ',', 'টুইটার', '@bengali_use
r', '!', '"']
PS C:\Users\shuvr\OneDrive\Documents\CODING>
```