Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

Name: Tushar Panchal

En.No: 21162101014

Sub: FP(Functional Programming)

Branch: CBA

Batch:41

---------------------------PRACTICAL 8----------------------------

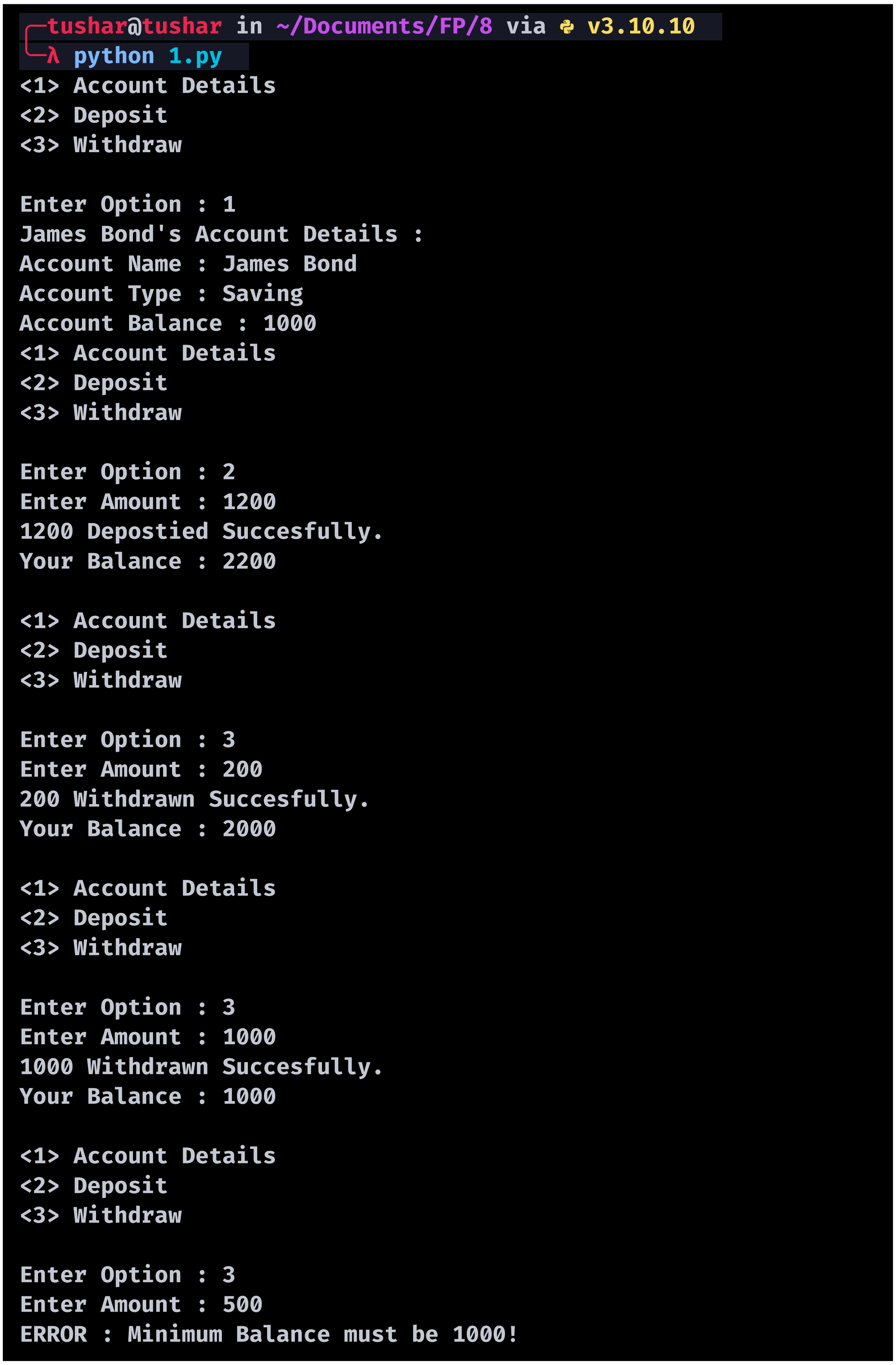
- **Question-1 :**

A bank application corresponding to the customer of ABC bank is being developed. It takes into consideration Name of customer, its account type (saving or current), balance corresponding to the account. Account type is by default fixed from the day when user creates and account in a bank. Withdrawal & deposit are other operations which any customer would do. Also implement function called setbalance() & getbalance(). Ensure that minimum amount to be maintained is 1000/- in both savings and current. An error should be raised if such scenario occurs.

✓ **Source Code :**

```python
class BankAccount:
    def __init__(self, name, acc_type, balance):
        self.name = name
        self.acc_type = 'Current' if acc_type == 0 else 'Saving'
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"{amount} Depostied Succesfully.")
        else:
            raise ValueError(
                f"ERROR : Invalid Amount , Try Again! [Amount:{amount}]")
        print(f"Your Balance : {self.balance}\n")
    def withdraw(self, amount):
        self.balance -= amount
        print(f"{amount} Withdrawn Succesfully.")
        print(f"Your Balance : {self.balance}\n")

    @property
    def balance(self):
        return self._balance

    @balance.setter
    def balance(self, bal):
        if bal >= 1000:
            self._balance = bal
        else:
            raise ValueError(f"ERROR : Minimum Balance must be 1000!")

    def show_details(self):
        print(f"{self.name}'s Account Details : ")
        print(f"Account Name : {self.name}")
        print(f"Account Type : {self.acc_type}")
        print(f"Account Balance : {self.balance}")


try:
    acc = BankAccount("James Bond", 1, 1000)
    while True:
        print("<1> Account Details")
        print("<2> Deposit")
        print("<3> Withdraw\n")

        choice = int(input("Enter Option : "))
        if choice == 1:
            acc.show_details()
        elif choice == 2:
            amt = int(input("Enter Amount : "))
            acc.deposit(amt)
        elif choice == 3:
            amt = int(input("Enter Amount : "))
            acc.withdraw(amt)
except Exception as e:
    print(e)
```

✓ **Output :**


```
┌─tushar@tushar in ~/Documents/FP/8 via 🐍 v3.10.10
└─λ python 1.py
<1> Account Details
<2> Deposit
<3> Withdraw

Enter Option : 1
James Bond's Account Details :
Account Name : James Bond
Account Type : Saving
Account Balance : 1000
<1> Account Details
<2> Deposit
<3> Withdraw

Enter Option : 2
Enter Amount : 1200
1200 Depostied Succesfully.
Your Balance : 2200

<1> Account Details
<2> Deposit
<3> Withdraw

Enter Option : 3
Enter Amount : 200
200 Withdrawn Succesfully.
Your Balance : 2000

<1> Account Details
<2> Deposit
<3> Withdraw

Enter Option : 3
Enter Amount : 1000
1000 Withdrawn Succesfully.
Your Balance : 1000

<1> Account Details
<2> Deposit
<3> Withdraw

Enter Option : 3
Enter Amount : 500
ERROR : Minimum Balance must be 1000!
```

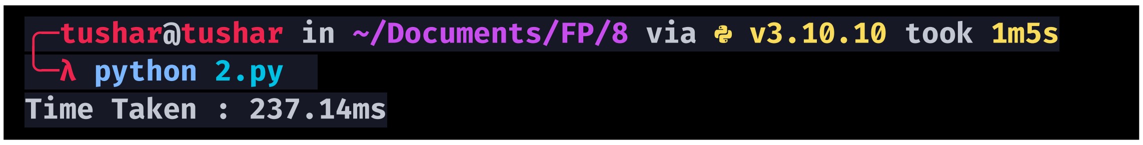
✓ Question-2 :

Design a class named StopWatch. The class contains:
■ The private data fields startTime and endTime with get methods.
■ A constructor that initializes startTime with the current time.
■ A method named start() that resets the startTime to the current time.
■ A method named stop() that sets the endTime to the current time.
■ A method named getElapsedTime() that returns the elapsed time for the stop watch in milliseconds.
Write a test program that measures the execution time of adding numbers from 1 to 1,000,000.

✓ Source Code :

```python
import time

class Stopwatch:
    def __init__(self):
        self.start_time = time.time()

    def start(self):
        self.start_time = time.time()

    def stop(self):
        self.end_time = time.time()

    def elapsed_time(self):
        return (self.end_time - self.start_time)*1000

sum = 0
sw = Stopwatch()
sw.start()
for i in range(1000001):
    sum += i
    sw.stop()
print(f"Time Taken : {round(sw.elapsed_time(),2)}ms")
```

✓ Output :

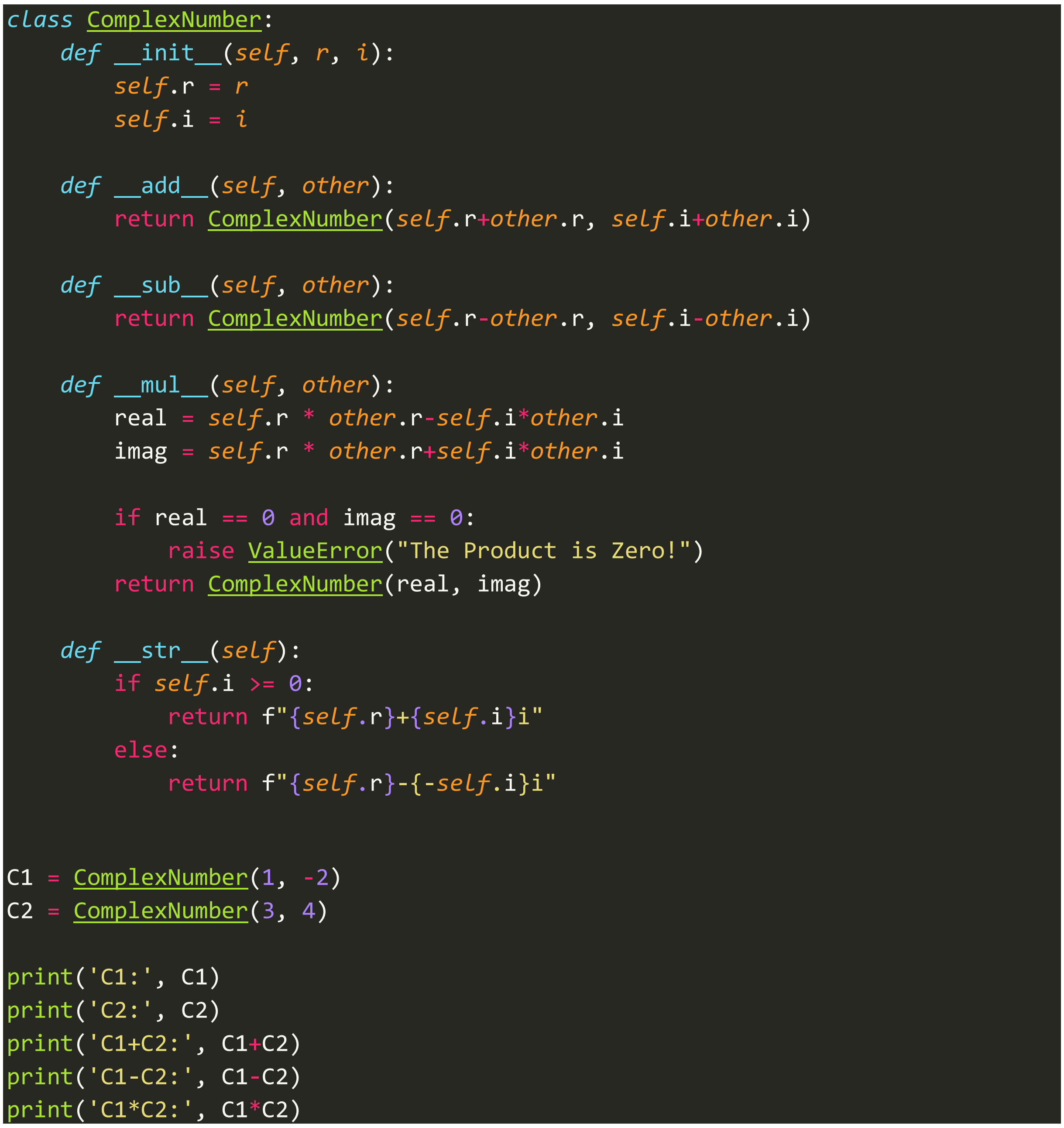


```
╭─tushar@tushar in ~/Documents/FP/8 via 🐍 v3.10.10 took 1m5s
╰─λ python 2.py
Time Taken : 237.14ms
```

✓ Question-3 :

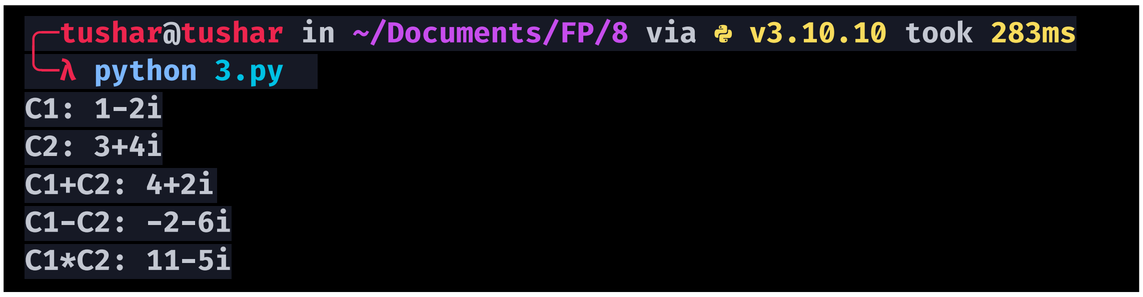
A small module for manipulation of complex numbers is being developed for ease of research related work. Implement add() for addition, mul() for multiplication , sub() for subtraction of two complex numbers. By default, a complex number will be assigned an imaginary value of 3. Also ensure that mul() cannot be ever 0 ; so program should raise an error.

✓ **Source Code :**



```python
class ComplexNumber:
    def __init__(self, r, i):
        self.r = r
        self.i = i

    def __add__(self, other):
        return ComplexNumber(self.r+other.r, self.i+other.i)

    def __sub__(self, other):
        return ComplexNumber(self.r-other.r, self.i-other.i)

    def __mul__(self, other):
        real = self.r * other.r-self.i*other.i
        imag = self.r * other.r+self.i*other.i

        if real == 0 and imag == 0:
            raise ValueError("The Product is Zero!")
        return ComplexNumber(real, imag)

    def __str__(self):
        if self.i >= 0:
            return f"{self.r}+{self.i}i"
        else:
            return f"{self.r}-{-self.i}i"


C1 = ComplexNumber(1, -2)
C2 = ComplexNumber(3, 4)

print('C1:', C1)
print('C2:', C2)
print('C1+C2:', C1+C2)
print('C1-C2:', C1-C2)
print('C1*C2:', C1*C2)
```

✓ **Output :**



```
╭─tushar@tushar in ~/Documents/FP/8 via 🐍 v3.10.10 took 283ms
╰─λ python 3.py
C1: 1-2i
C2: 3+4i
C1+C2: 4+2i
C1-C2: -2-6i
C1*C2: 11-5i
```