Misha Akram CS-18118
Iqra Irfan CS-18123

Mujtaba Khan CS-18136
Firdous Riaz CS-18141

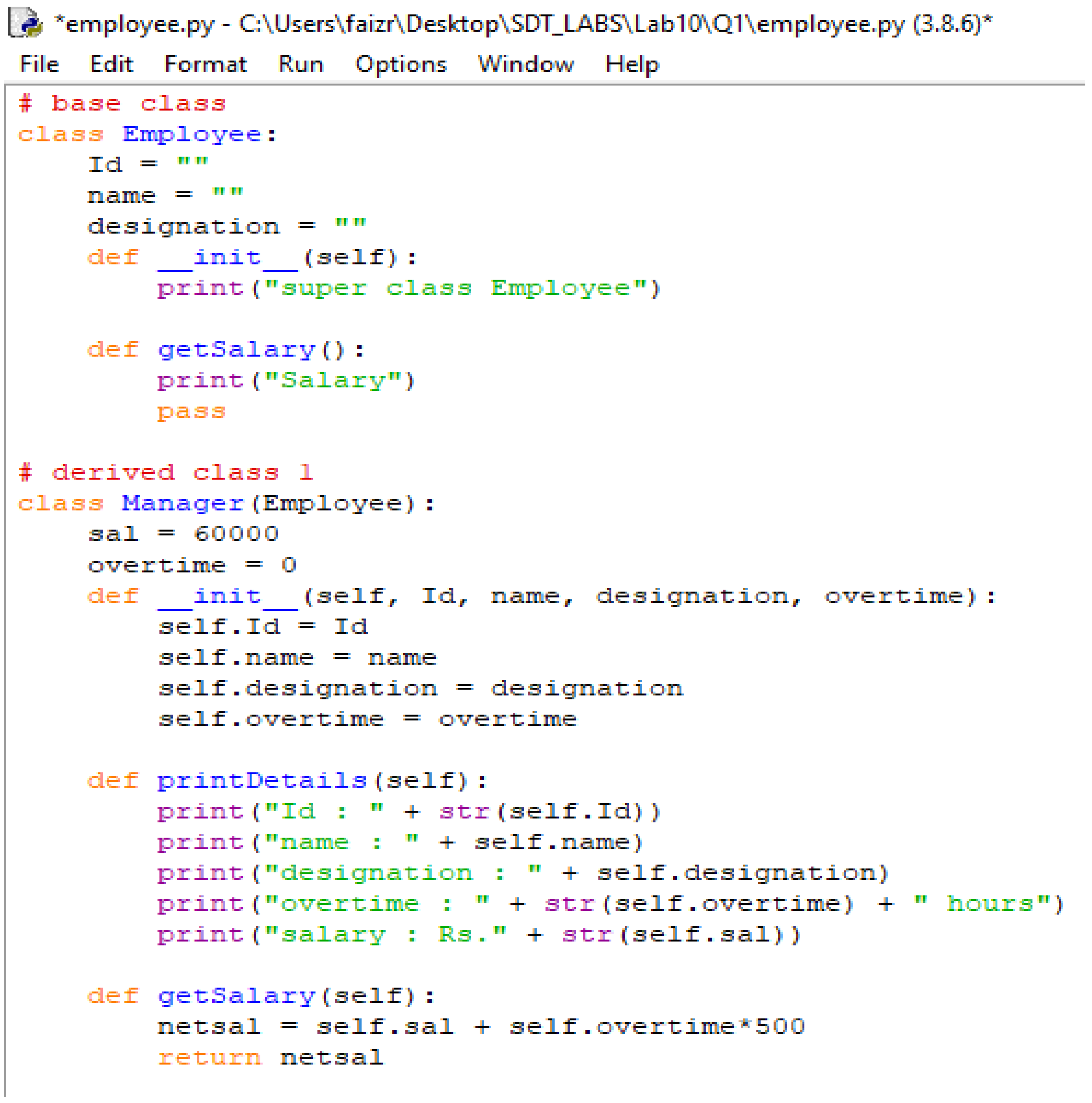
Lab Session 10

Practice testing of object-oriented applications

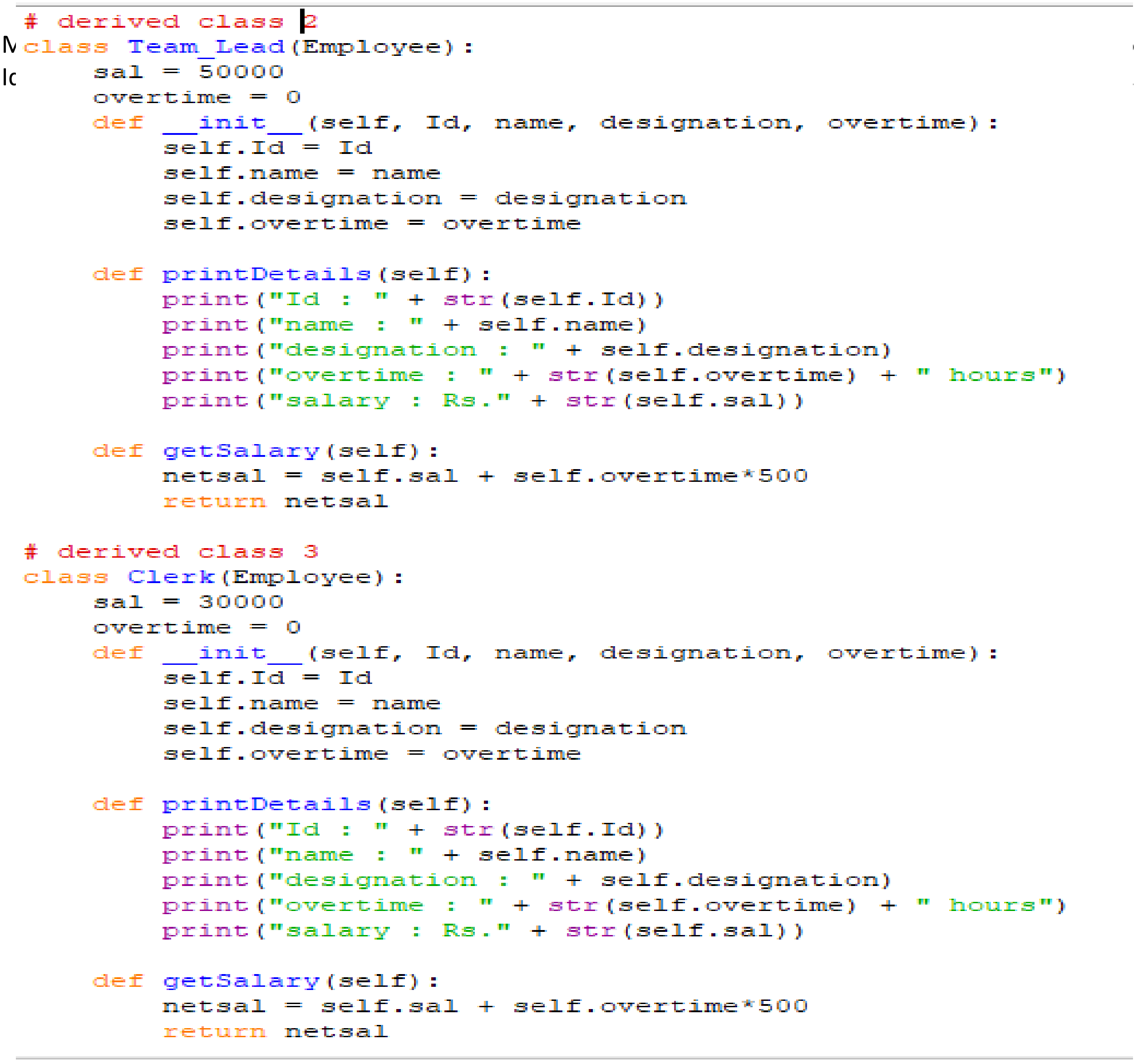
Exercises:

1. For the parent class 'Employee' having attributes of employee_id, employee_name and designation make functions to print the details and calculate the salary of the employee. Also create three child classes of Manager, Team_Lead and Clerk with suitable attributes and same functions. Create another script having the test functions to test the code. Test the functions using pytest.

employee.py:


employee.py - C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q1\employee.py (3.8.6)
File Edit Format Run Options Window Help

```
# base class
class Employee:
    Id = ""
    name = ""
    designation = ""
    def __init__(self):
        print("super class Employee")

    def getSalary():
        print("Salary")
        pass

# derived class 1
class Manager(Employee):
    sal = 60000
    overtime = 0
    def __init__(self, Id, name, designation, overtime):
        self.Id = Id
        self.name = name
        self.designation = designation
        self.overtime = overtime

    def printDetails(self):
        print("Id : " + str(self.Id))
        print("name : " + self.name)
        print("designation : " + self.designation)
        print("overtime : " + str(self.overtime) + " hours")
        print("salary : Rs." + str(self.sal))

    def getSalary(self):
        netsal = self.sal + self.overtime*500
        return netsal
```



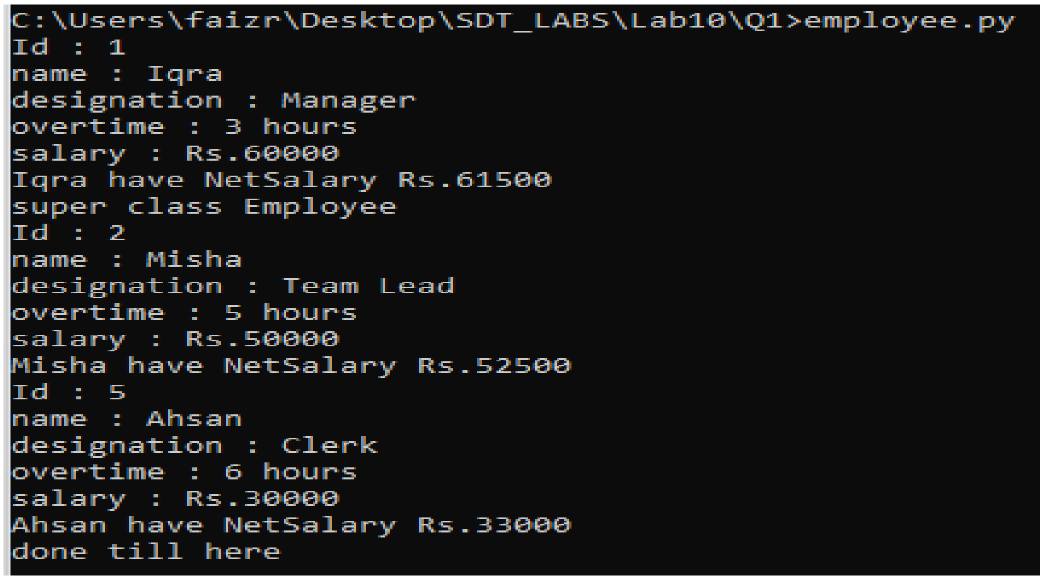
```
# derived class 2
class Team_Lead(Employee):
    sal = 50000
    overtime = 0
    def __init__(self, Id, name, designation, overtime):
        self.Id = Id
        self.name = name
        self.designation = designation
        self.overtime = overtime

    def printDetails(self):
        print("Id : " + str(self.Id))
        print("name : " + self.name)
        print("designation : " + self.designation)
        print("overtime : " + str(self.overtime) + " hours")
        print("salary : Rs." + str(self.sal))

    def getSalary(self):
        netsal = self.sal + self.overtime*500
        return netsal

# derived class 3
class Clerk(Employee):
    sal = 30000
    overtime = 0
    def __init__(self, Id, name, designation, overtime):
        self.Id = Id
        self.name = name
        self.designation = designation
        self.overtime = overtime

    def printDetails(self):
        print("Id : " + str(self.Id))
        print("name : " + self.name)
        print("designation : " + self.designation)
        print("overtime : " + str(self.overtime) + " hours")
        print("salary : Rs." + str(self.sal))

    def getSalary(self):
        netsal = self.sal + self.overtime*500
        return netsal
```

```
def main():
    m = Manager(1, 'Iqra', 'Manager', 3)
    m.printDetails()
    print(m.name + " have NetSalary Rs." + str(m.getSalary()))

    e = Employee()
    t = Team_Lead(2, 'Misha', 'Team Lead', 5)
    t.printDetails()
    print(t.name + " have NetSalary Rs." + str(t.getSalary()))
    c = Clerk(5, 'Ahsan', 'Clerk', 6)
    c.printDetails()
    print(c.name + " have NetSalary Rs." + str(c.getSalary()))
    print("done till here")

if __name__ == '__main__':
    main()
```

Misha Akram CS-18118
Iqra Irfan CS-18123

Mujtaba Khan CS-18136
Firdous Riaz CS-18141

Output:



```
C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q1>employee.py
Id : 1
name : Iqra
designation : Manager
overtime : 3 hours
salary : Rs.60000
Iqra have NetSalary Rs.61500
super class Employee
Id : 2
name : Misha
designation : Team Lead
overtime : 5 hours
salary : Rs.50000
Misha have NetSalary Rs.52500
Id : 5
name : Ahsan
designation : Clerk
overtime : 6 hours
salary : Rs.30000
Ahsan have NetSalary Rs.33000
done till here
```

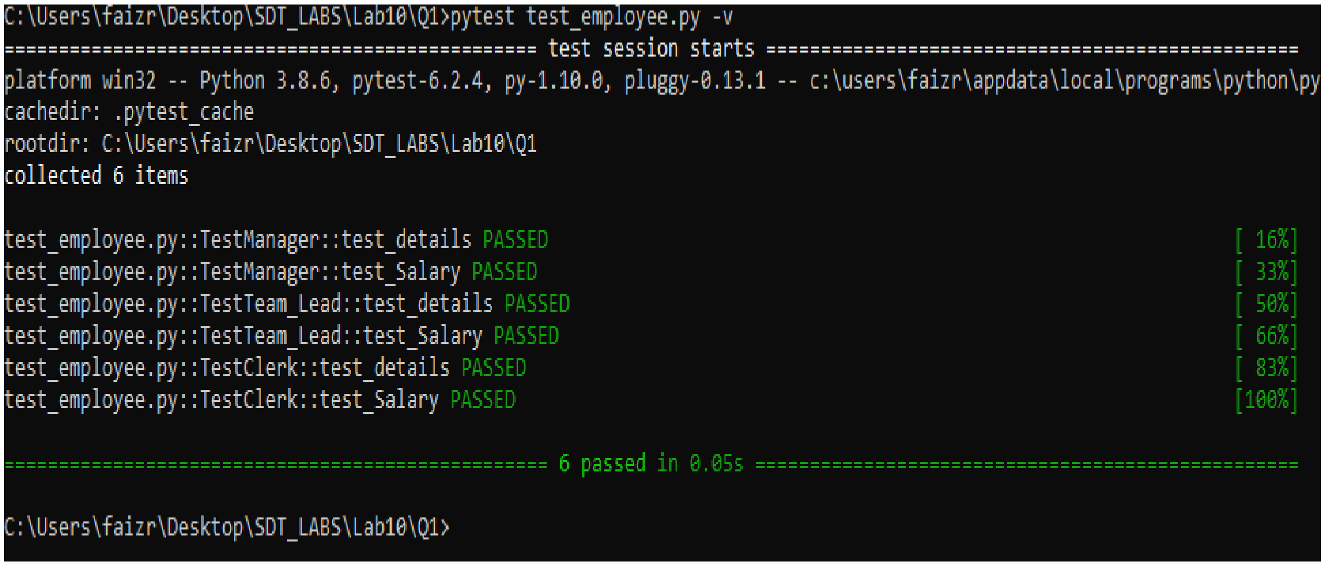
test_employee.py:

```python
import pytest
from employee import Manager, Team_Lead, Clerk

class TestManager:
# test case for printing details of Manager
    def test_details(self):
        m = Manager(1, 'Iqra', 'manager', 3)
        print("testing details : manager")
        assert (1, 'Iqra' , 'manager', 3, 60000) == (m.Id, m.name, m.designation, m.overtime, m.sal)
# test case for calculating Salary of manager
    def test_Salary(self):
        m = Manager(2,'Firdous', 'Manager', 14)
        print("testing salary : manager")
        netsal = m.getSalary()
        assert netsal == 67000

class TestTeam_Lead:
    def test_details(self):
        t = Team_Lead(2, 'Misha', 'team_lead', 5)
        print("testing details : team_lead")
        assert (2, 'Misha' , 'team_lead', 5, 50000) == (t.Id, t.name, t.designation, t.overtime, t.sal)
    def test_Salary(self):
        t = Team_Lead(5,'Mujtuba', 'team_lead', 0)
        print("testing salary : team_lead")
        netsal = t.getSalary()
        assert netsal == 50000

class TestClerk:
    def test_details(self):
        c = Clerk(6, 'Ahsan', 'clerk', 6)
        print("testing details : clerk")
        assert (6, 'Ahsan' , 'clerk', 6, 30000) == (c.Id, c.name, c.designation, c.overtime, c.sal)
    def test_Salary(self):
        c = Clerk(3,'Sajid', 'clerk', 16)
        print("testing salary : clerk")
        netsal = c.getSalary()
        assert netsal == 38000
```

Misha Akram CS-18118
Iqra Irfan CS-18123

Mujtaba Khan CS-18136
Firdous Riaz CS-18141

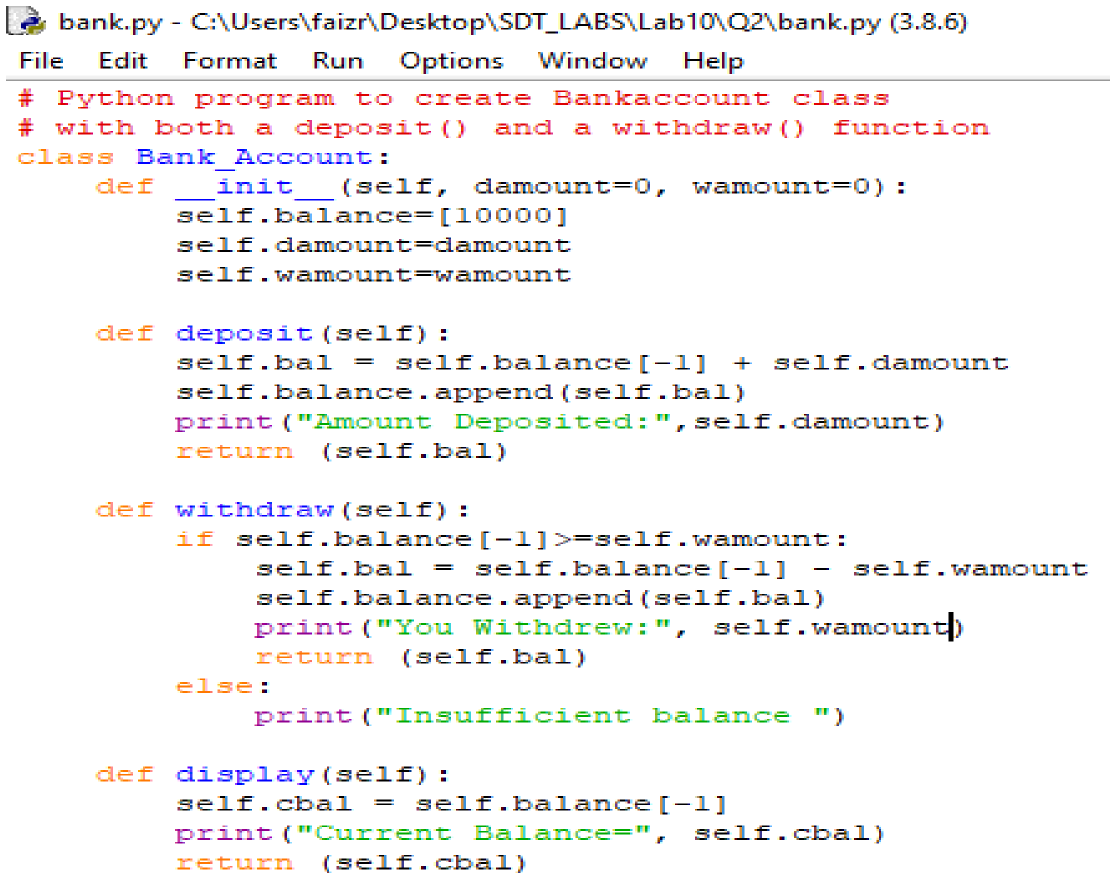
Output:



```
C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q1>pytest test_employee.py -v
============================================== test session starts ===============================================
platform win32 -- Python 3.8.6, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- c:\users\faizr\appdata\local\programs\python\py
cachedir: .pytest_cache
rootdir: C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q1
collected 6 items

test_employee.py::TestManager::test_details PASSED                                                          [ 16%]
test_employee.py::TestManager::test_Salary PASSED                                                           [ 33%]
test_employee.py::TestTeam_Lead::test_details PASSED                                                        [ 50%]
test_employee.py::TestTeam_Lead::test_Salary PASSED                                                         [ 66%]
test_employee.py::TestClerk::test_details PASSED                                                            [ 83%]
test_employee.py::TestClerk::test_Salary PASSED                                                             [100%]

================================================ 6 passed in 0.05s ================================================

C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q1>
```

2. For the class 'Bank_Account' having the attribute of account_number, define two child classes ('Saving_Account' and 'Current_Account'). For the Saving_Account, calculate the minimum_ balance and interest_amount. For the Current_Account, specify the withdrawal_limit and calculate it after every transaction. Test the functions developed using pytest.

bank.py:

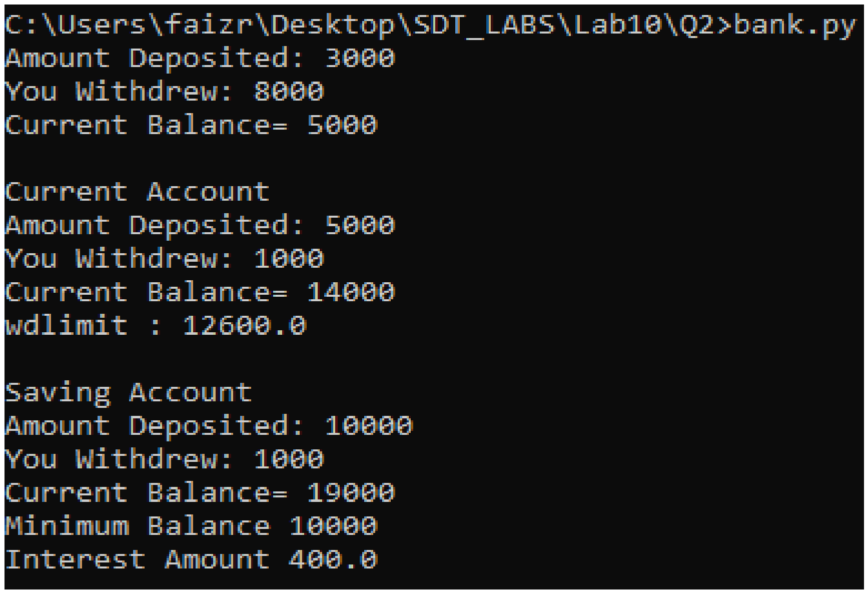


bank.py - C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q2\bank.py (3.8.6)
File Edit Format Run Options Window Help

```python
# Python program to create Bankaccount class
# with both a deposit() and a withdraw() function
class Bank_Account:
    def __init__(self, damount=0, wamount=0):
        self.balance=[10000]
        self.damount=damount
        self.wamount=wamount

    def deposit(self):
        self.bal = self.balance[-1] + self.damount
        self.balance.append(self.bal)
        print("Amount Deposited:",self.damount)
        return (self.bal)

    def withdraw(self):
        if self.balance[-1]>=self.wamount:
            self.bal = self.balance[-1] - self.wamount
            self.balance.append(self.bal)
            print("You Withdrew:", self.wamount)
            return (self.bal)
        else:
            print("Insufficient balance ")

    def display(self):
        self.cbal = self.balance[-1]
        print("Current Balance=", self.cbal)
        return (self.cbal)
```

```
class Current_Account(Bank_Account):
    withdrawl_limit=90
    def wdlimit(self):
        self.limit=(self.balance[-1]*self.withdrawl_limit)/100
        print("wdlimit : " + str(self.limit))
        return (self.limit)

class Saving_Account(Bank_Account):
    intRate = 4
    def min_bal(self):
        return min(self.balance)

    def interest(self):
        return (min(self.balance)*4)/100

def main():
    s = Bank_Account(3000,8000)
    s.deposit()
    s.withdraw()
    s.display()

    print("\nCurrent Account")
    c = Current_Account(5000,1000)
    c.deposit()
    c.withdraw()
    c.display()
    c.wdlimit()

    print("\nSaving Account")
    s = Saving_Account(10000,1000)
    s.deposit()
    s.withdraw()
    s.display()
    print("Minimum Balance " + str(s.min_bal()))
    print("Interest Amount " + str(s.interest()))

if __name__ == '__main__':
    main()
```

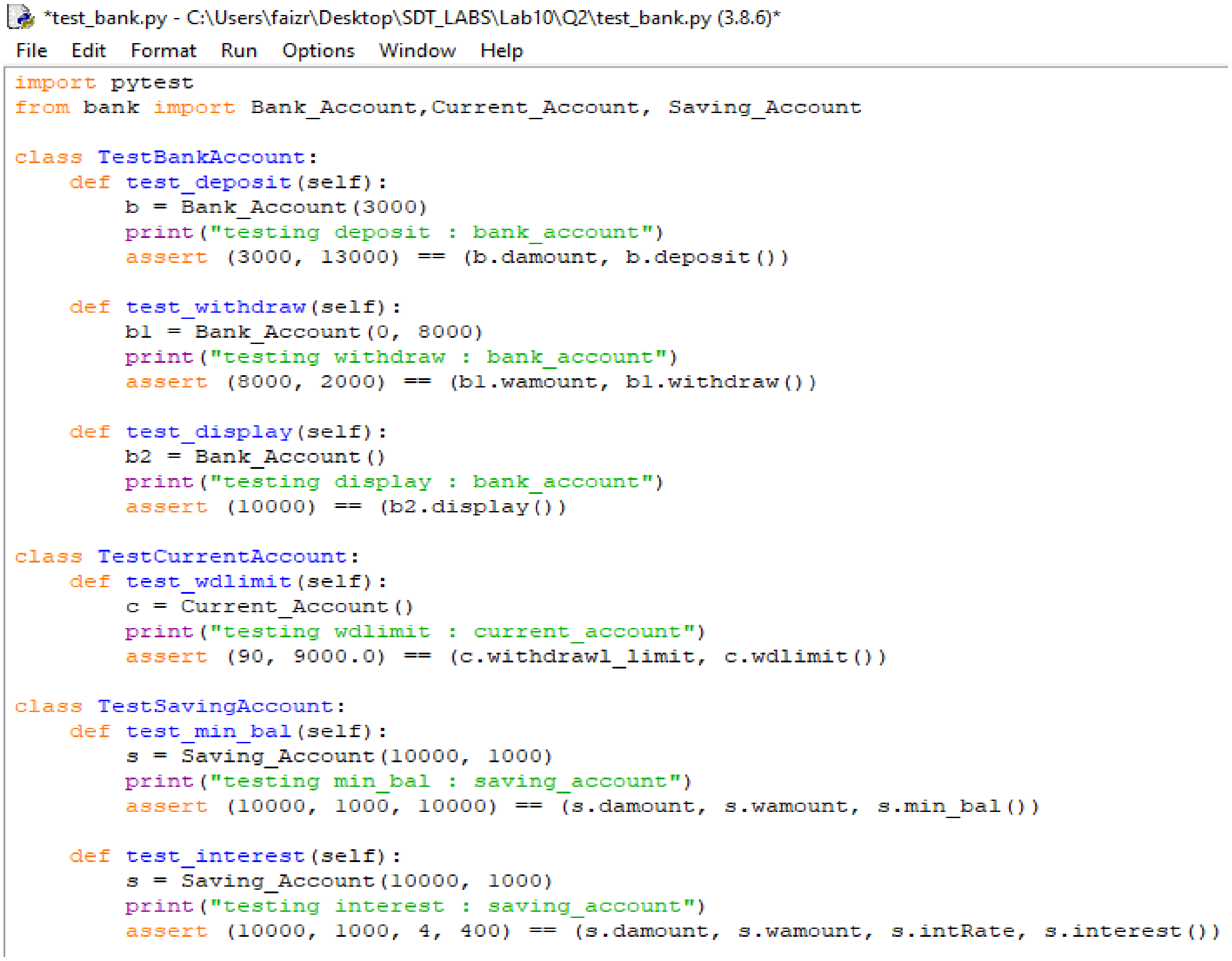
Output:



```
C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q2>bank.py
Amount Deposited: 3000
You Withdrew: 8000
Current Balance= 5000

Current Account
Amount Deposited: 5000
You Withdrew: 1000
Current Balance= 14000
wdlimit : 12600.0

Saving Account
Amount Deposited: 10000
You Withdrew: 1000
Current Balance= 19000
Minimum Balance 10000
Interest Amount 400.0
```

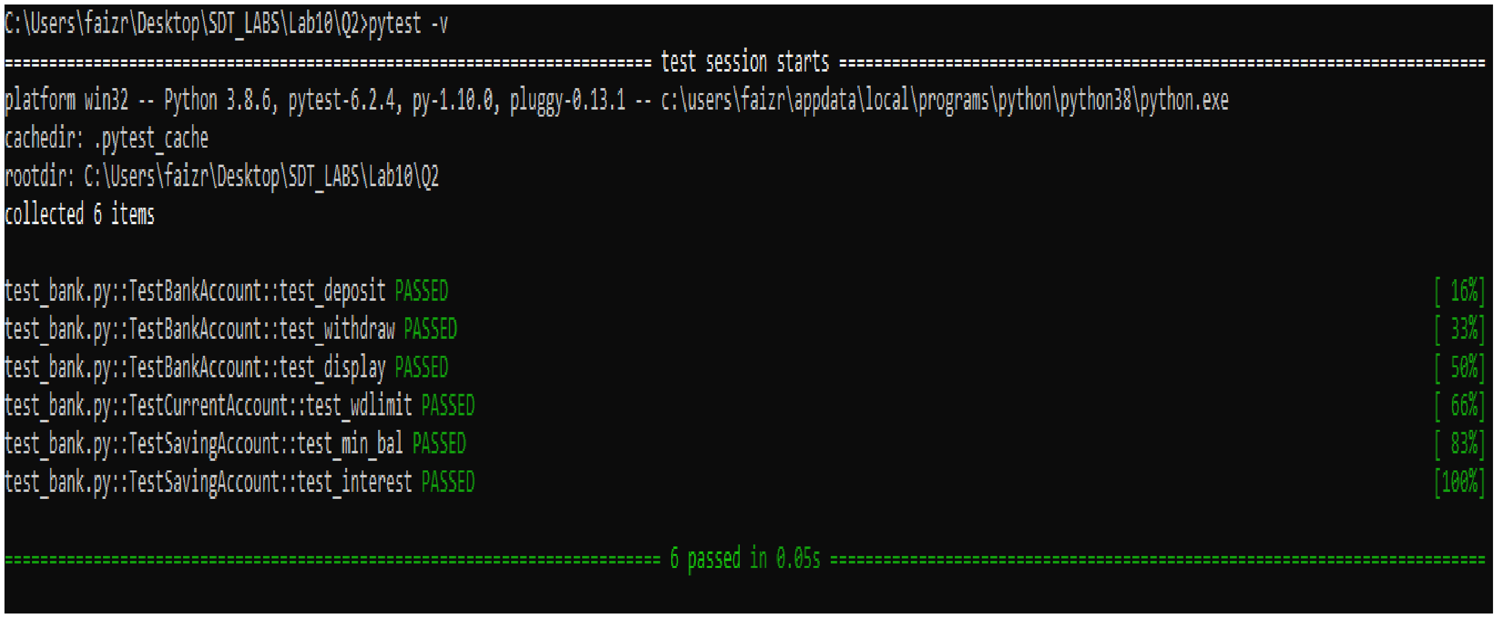
Misha Akram CS-18118 Mujtaba Khan CS-18136
Iqra Irfan CS-18123 Firdous Riaz CS-18141

test_bank.py:


test_bank.py - C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q2\test_bank.py (3.8.6)
File Edit Format Run Options Window Help

```python
import pytest
from bank import Bank_Account,Current_Account, Saving_Account

class TestBankAccount:
    def test_deposit(self):
        b = Bank_Account(3000)
        print("testing deposit : bank_account")
        assert (3000, 13000) == (b.damount, b.deposit())

    def test_withdraw(self):
        b1 = Bank_Account(0, 8000)
        print("testing withdraw : bank_account")
        assert (8000, 2000) == (b1.wamount, b1.withdraw())

    def test_display(self):
        b2 = Bank_Account()
        print("testing display : bank_account")
        assert (10000) == (b2.display())

class TestCurrentAccount:
    def test_wdlimit(self):
        c = Current_Account()
        print("testing wdlimit : current_account")
        assert (90, 9000.0) == (c.withdrawl_limit, c.wdlimit())

class TestSavingAccount:
    def test_min_bal(self):
        s = Saving_Account(10000, 1000)
        print("testing min_bal : saving_account")
        assert (10000, 1000, 10000) == (s.damount, s.wamount, s.min_bal())

    def test_interest(self):
        s = Saving_Account(10000, 1000)
        print("testing interest : saving_account")
        assert (10000, 1000, 4, 400) == (s.damount, s.wamount, s.intRate, s.interest())
```

Output:



```
C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q2>pytest -v
=================================================================== test session starts ===================================================================
platform win32 -- Python 3.8.6, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 -- c:\users\faizr\appdata\local\programs\python\python38\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\faizr\Desktop\SDT_LABS\Lab10\Q2
collected 6 items

test_bank.py::TestBankAccount::test_deposit PASSED                                                                                                  [ 16%]
test_bank.py::TestBankAccount::test_withdraw PASSED                                                                                                 [ 33%]
test_bank.py::TestBankAccount::test_display PASSED                                                                                                  [ 50%]
test_bank.py::TestCurrentAccount::test_wdlimit PASSED                                                                                               [ 66%]
test_bank.py::TestSavingAccount::test_min_bal PASSED                                                                                                [ 83%]
test_bank.py::TestSavingAccount::test_interest PASSED                                                                                               [100%]

==================================================================== 6 passed in 0.05s ====================================================================
```