ASSIGNMENT-6.5

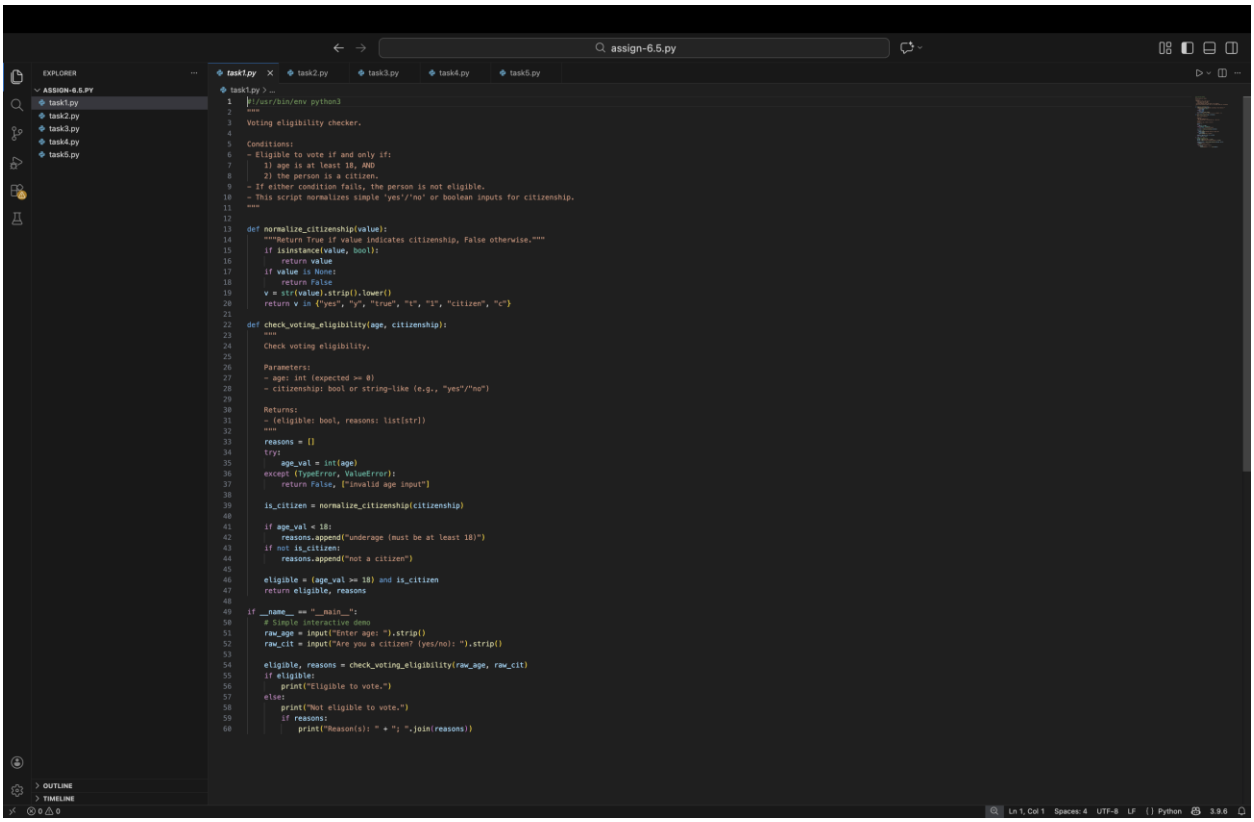
2303A51042

BATCH – 29

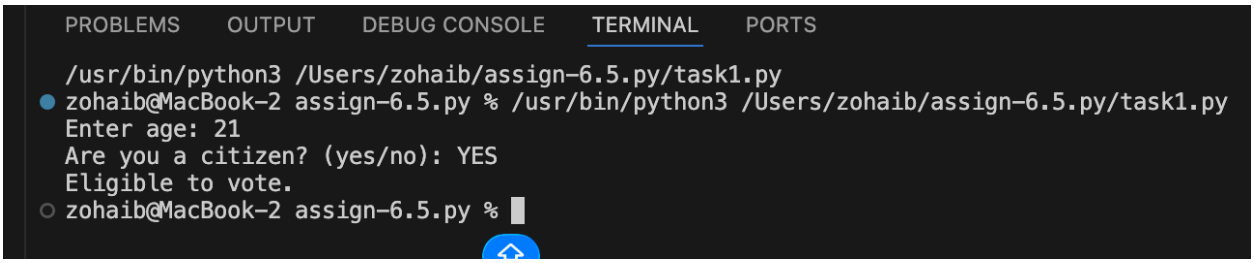
TASK-1:

Prompt : Generate Python code to check voting eligibility based on age and Citizenship.

CODE:



OUTPUT:



OBSERVATION:

The program accurately determines voting eligibility using age and citizenship conditions with proper logical validation. It handles different input formats and invalid values safely using normalization and exception handling. Clear reasons are displayed when eligibility requirements are not met.

TASK-2:

Prompt **: Generate Python code to count vowels and consonants in a string using a loop.**

CODE:

task2.py > ...

```python
from typing import Tuple

#!/usr/bin/env python3
"""
Count vowels and consonants in a string using a loop.
Includes simple output verification.
"""


VOWELS = set("aeiouAEIOU")


def count_vowels_consonants(s: str) -> Tuple[int, int]:
    vowels = 0
    consonants = 0
    for ch in s:
        if ch.isalpha():             # consider letters only
            if ch in VOWELS:
                vowels += 1
            else:
                consonants += 1
    return vowels, consonants


if __name__ == "__main__":
    examples = [
        "Hello, World!",
        "AI-based code completion 123",
        "aeiouAEIOU",
        "Python3.9",
        ""
    ]

    for text in examples:
        v, c = count_vowels_consonants(text)
        total_letters = sum(1 for ch in text if ch.isalpha())
        print(f"Input: {text!r}")
        print(f"Vowels: {v}, Consonants: {c}")
        # Output verification
        assert v + c == total_letters, "Verification failed: counts don't sum to total letters"
        print("Verification: passed\n")

    # Optional interactive test
    s = input("Enter a string to analyze (or press Enter to exit): ")
    if s:
        v, c = count_vowels_consonants(s)
        print(f"Result -> Vowels: {v}, Consonants: {c}")
```

OUTPUT:

```
zohaib@MacBook-2 assign-6.5.py % /usr/bin/python3 /Users/zohaib/assign-6.5.py/task2.py
Input: 'Hello, World!'
Vowels: 3, Consonants: 7
Verification: passed

Input: 'AI-based code completion 123'
Vowels: 10, Consonants: 11
Verification: passed

Input: 'aeiouAEIOU'
Vowels: 10, Consonants: 0
Verification: passed

Input: 'Python3.9'
Vowels: 1, Consonants: 5
Verification: passed

Input: ''
Vowels: 0, Consonants: 0
Verification: passed

Enter a string to analyze (or press Enter to exit):
```

OBSERVATION:

The program accurately counts vowels and consonants by iterating through each character and considering only alphabetic inputs. It ignores numbers and symbols, ensuring correct results. Output verification confirms that the total count matches the number of letters in the string.

TASK-3:

Prompt : Generate a Python program for a library management system using classes, loops, and conditional statements.

CODE:

task3.py > Book > free

```python
import json, os
from dataclasses import dataclass, asdict
FILE="library_catalog.json"
@dataclass
class Book:
    id:int; title:str; author:str; year:int; borrower:str=None
    def free(self): return self.borrower is None
class Library:
    def __init__(s): s.books=[]; s.n=1

    def add(s,t,a,y):
        b=Book(s.n,t.strip(),a.strip(),y); s.books.append(b); s.n+=1; return b

    def get(s,i): return next((b for b in s.books if b.id==i),None)

    def remove(s,i):
        b=s.get(i)
        if b: s.books.remove(b); return True
        return False

    def borrow(s,i,n):
        b=s.get(i)
        if b and b.free(): b.borrower=n.strip(); return True
        return False

    def ret(s,i):
        b=s.get(i)
        if b and not b.free(): b.borrower=None; return True
        return False

    def search(s,k):
        k=str(k).lower()
        return [b for b in s.books if k in b.title.lower() or k in b.author.lower() or k==str(b.year)]

    def save(s):
        with open(FILE,"w") as f:
            json.dump({"n":s.n,"b":[asdict(x) for x in s.books]},f)

    def load(s):
        if not os.path.exists(FILE): return
        d=json.load(open(FILE))
        s.n=d["n"]; s.books=[Book(**x) for x in d["b"]]

def pint(m):
    while 1:
        try: return int(input(m))
        except: print("Enter number")

def show(b):
    print(f"[{b.id}] {b.title} - {b.author} ({b.year}) - {'Available' if b.free() else 'Borrowed by '+b.borrower}")

def main():
    L=Library(); L.load()
    while 1:
        c=input("\n1 Add 2 Remove 3 All 4 Free 5 Borrow 6 Return 7 Search 8 Save 9 Exit: ")
        if c=="1": show(L.add(input("Title:"),input("Author:"),pint("Year:")))
        elif c=="2": print("Removed" if L.remove(pint("ID:")) else "Not found")
        elif c=="3": [show(b) for b in L.books] or print("Empty")
        elif c=="4": [show(b) for b in L.books if b.free()] or print("None free")
        elif c=="5": print("Done" if L.borrow(pint("ID:"),input("Name:")) else "Fail")
        elif c=="6": print("Returned" if L.ret(pint("ID:")) else "Fail")
        elif c=="7": [show(b) for b in L.search(input("Key:"))] or print("No match")
        elif c=="8": L.save(); print("Saved")
        elif c=="9": L.save(); break
        else: print("Wrong choice")

main()
```

OUTPUT:

```
zohaib@MacBook-2 assign-6.5.py % /usr/bin/python3 /Users/zohaib/assign-6.5.py/task3.py

Library Menu:
1) Add book
2) Remove book
3) List all books
4) List available books
5) Borrow book
6) Return book
7) Search books
8) Save catalog
9) Exit
Choose an option (1-9): 1
Title: ZZZ
Author: ZZZ
Year: 2002
Added book:
[1] ZZZ — ZZZ (2002) — Available

Library Menu:
1) Add book
2) Remove book
3) List all books
4) List available books
5) Borrow book
6) Return book
7) Search books
8) Save catalog
9) Exit
Choose an option (1-9): 3
[1] ZZZ — ZZZ (2002) — Available

Library Menu:
1) Add book
2) Remove book
3) List all books
4) List available books
5) Borrow book
6) Return book
7) Search books
8) Save catalog
9) Exit
Choose an option (1-9):
```

OBSERVATION:

The program efficiently manages book records using object-oriented concepts and stores data persistently in a JSON file. It supports adding, borrowing, returning, searching, and deleting books with proper validation. The menu-driven system allows continuous user interaction until exit.

TASK-4:

Prompt: Generate a Python class to mark and display student attendance using loops.

CODE:

task4.py > ...

```python
class AttendanceManager:
    def __init__(s, students):
        s.students=list(students)
        s.att={}

    def mark(s,d,present):
        P=set(present)
        s.att[d]={st:st in P for st in s.students}

    def mark_one(s,d,st,val=True):
        s.att.setdefault(d,{x:False for x in s.students})
        if st in s.students: s.att[d][st]=val

    def show(s,d):
        if d not in s.att: return print("No record for",d)
        print("Attendance for",d)
        for st in s.students:
            print(f" - {st}: {'Present' if s.att[d].get(st) else 'Absent'}")

    def get(s,d): return dict(s.att.get(d,{}))


if __name__=="__main__":
    names=["Alice","Bob","Charlie","Diana","Eve"]
    m=AttendanceManager(names)

    m.mark("2026-02-10",["Alice","Charlie","Eve"])
    m.show("2026-02-10"); print()

    m.mark("2026-02-11",[n for i,n in enumerate(names) if i%2==0])
    m.show("2026-02-11"); print()

    m.mark_one("2026-02-12","Bob")
    m.show("2026-02-12")

    r=m.get("2026-02-10")
    assert r["Alice"] and not r["Bob"]
    print("\nAll tests passed")

```

OUTPUT:

```
zohaib@MacBook-2 assign-6.5.py % /usr/bin/python3 /Users/zohaib/assign-6.5.py/task4.py
 - Bob: Absent
 - Charlie: Present
 - Diana: Absent
 - Eve: Present

Attendance for 2026-02-11
 - Alice: Present
 - Bob: Absent
 - Charlie: Present
 - Diana: Absent
 - Eve: Present

Attendance for 2026-02-12
 - Alice: Absent
 - Bob: Present
 - Charlie: Absent
 - Diana: Absent
 - Eve: Absent

All tests passed
```

OBSERVATION:

The program effectively records and displays student attendance using dictionaries and object-oriented design. It supports marking attendance for all students at once or individually by date. Built-in assertions verify the correctness of stored records.

TASK-5:

Prompt: Generate a Python program using loops and conditionals to simulate an ATM menu.

CODE:

task1.py | task2.py | task3.py | task4.py | task5.py

task5.py > ...

```python
import sys

class ATM:
    def __init__(s,b=1000,p="1234"):
        s.b=float(b); s.p=str(p)

    def amt(s,m):
        try:
            a=float(input(m))
            return a if a>0 else print("Amount must be positive.")
        except: print("Invalid amount.")

    def bal(s): print(f"Balance: ${s.b:.2f}")

    def dep(s):
        a=s.amt("Deposit: ")
        if a: s.b+=a; s.bal()

    def wd(s):
        a=s.amt("Withdraw: ")
        if a and a<=s.b: s.b-=a; s.bal()
        elif a: print("Insufficient funds")

    def trans(s):
        a=s.amt("Transfer: ")
        if not a or a>s.b: return print("Insufficient funds")
        if not input("Dest ID: ").strip(): return print("Invalid dest")
        s.b-=a; s.bal()

    def pin(s):
        if input("Old PIN: ")!=s.p: return print("Wrong PIN")
        n=input("New PIN: ")
        if n.isdigit() and len(n)==4: s.p=n; print("PIN changed")
        else: print("PIN must be 4 digits")

    def auth(s):
        for _ in range(3):
            if input("PIN: ")==s.p: return 1
            print("Wrong PIN")
        return print("Locked")

    def menu(s):
        if not s.auth(): return
        while 1:
            c=input("\n1 Bal 2 Dep 3 Wd 4 Trans 5 PIN 6 Exit: ")
            if c=="1": s.bal()
            elif c=="2": s.dep()
            elif c=="3": s.wd()
            elif c=="4": s.trans()
            elif c=="5": s.pin()
            elif c=="6": break
            else: print("Invalid")

def test():
    a=ATM(500,"0000")
    a.b+=200; a.b-=150; a.b-=50; a.p="1234"
    assert a.b==500 and a.p=="1234"
    print("Test OK — Balance:",a.b)

if __name__=="__main__":
    test() if len(sys.argv)>1 else ATM().menu()
```

OUTPUT :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

/usr/bin/python3 /Users/zohaib/assign-6.5.py/task5.py
zohaib@MacBook-2 assign-6.5.py % /usr/bin/python3 /Users/zohaib/assign-6.5.py/task5.py
PIN: 1234

1 Bal 2 Dep 3 Wd 4 Trans 5 PIN 6 Exit: 1
Balance: $1000.00

1 Bal 2 Dep 3 Wd 4 Trans 5 PIN 6 Exit: 2
Deposit: 12
Balance: $1012.00

1 Bal 2 Dep 3 Wd 4 Trans 5 PIN 6 Exit:
```

OBSERVATION :

The program simulates core ATM operations including balance check, deposit, withdrawal, fund transfer, and PIN change using a menu-driven approach. It validates user inputs and restricts access through PIN authentication with limited attempts. A built-in test confirms correct balance updates and PIN modification.