THE INDIAN HIGH SCHOOL - DUBAI

Comp. Sci Journal
2025-26

Name: Abyaz Javid
Roll no.: 4

CERTIFICATE

This is to certify that the work in this journal is the bonafide work of

Master ______________________________

*Class*__________ *Div*________

recorded in the school lab during the academic year

20xx-20xx

Date: ____________________

Teacher in charge: ____________________

Proud winner of the Sheikh Hamdan bin Rashid Al Maktoum Award for Distinguished School & School Administration, 2002 & 2005

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the Central Board of Secondary Education (CBSE) and The Indian High School-Dubai, for granting me the opportunity to deepen my knowledge in my favorite subject, Computer Science

I would also like to thank my teacher Mrs. Swapnil Verma for guiding me and sharing her wide variety of knowledge

INDEX

FUNCTIONS

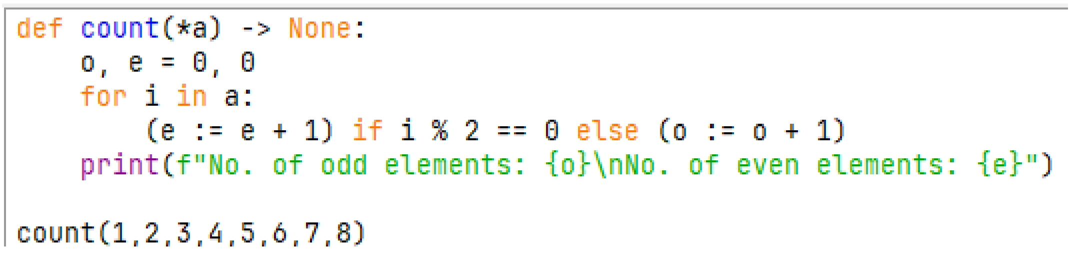
Program 1

Aim: To write a function that takes 'n' number of integers and count the number of odd and even numbers.

Modules used: N/A

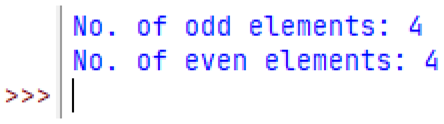
Data types used: Int

Script:



```
def count(*a) -> None:
    o, e = 0, 0
    for i in a:
        (e := e + 1) if i % 2 == 0 else (o := o + 1)
    print(f"No. of odd elements: {o}\nNo. of even elements: {e}")

count(1,2,3,4,5,6,7,8)
```

Output:



```
No. of odd elements: 4
No. of even elements: 4
>>> |
```

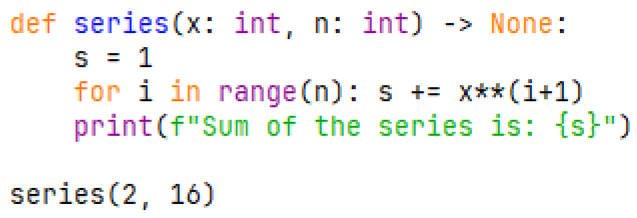
Program 2

Aim: To write a function to find the sum of the series: $1 + x^2 + x^3 + \cdots + x^n$

Modules used: N/A

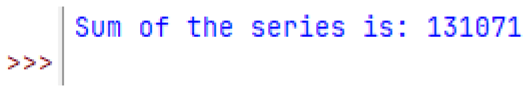
Data types used: Int

Script:



```python
def series(x: int, n: int) -> None:
    s = 1
    for i in range(n): s += x**(i+1)
    print(f"Sum of the series is: {s}")

series(2, 16)
```

Output:



```
    Sum of the series is: 131071
>>>
```

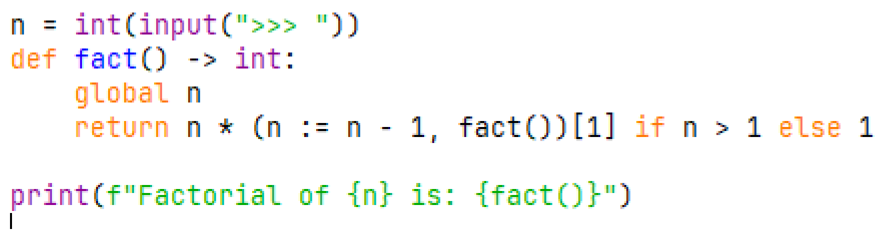
Program 3

Aim: To write a function to find the factorial of a number without taking an argument

Modules used: N/A

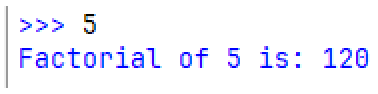
Data types used: Int

Script:



```python
n = int(input(">>> "))
def fact() -> int:
    global n
    return n * (n := n - 1, fact())[1] if n > 1 else 1

print(f"Factorial of {n} is: {fact()}")
```

Output:



```
>>> 5
Factorial of 5 is: 120
```

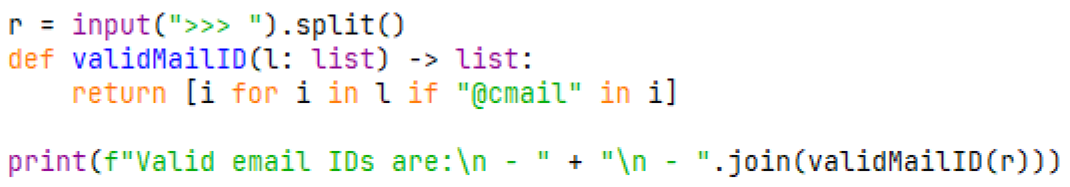
Program 4

Aim: To write a function that takes a list of strings and return the emails that contain the substring “@cmail”

Modules used: N/A

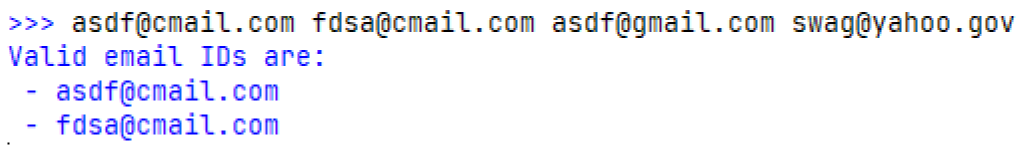
Data types used: String, List

Script:



```python
r = input(">>> ").split()
def validMailID(l: list) -> list:
    return [i for i in l if "@cmail" in i]

print(f"Valid email IDs are:\n - " + "\n - ".join(validMailID(r)))
```

Output:



```
>>> asdf@cmail.com fdsa@cmail.com asdf@gmail.com swag@yahoo.gov
Valid email IDs are:
 - asdf@cmail.com
 - fdsa@cmail.com
```

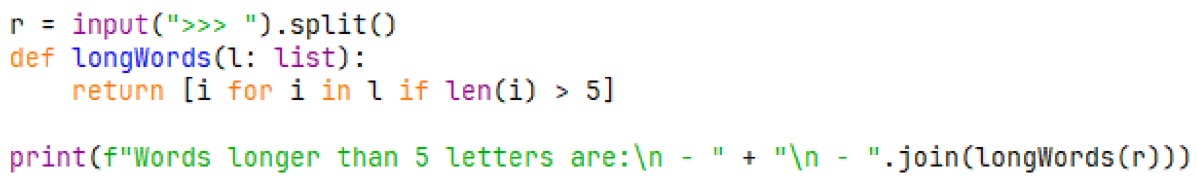
Program 5

Aim: To write a function that takes a list of strings and returns the strings that are longer than 5 characters.

Modules used: N/A

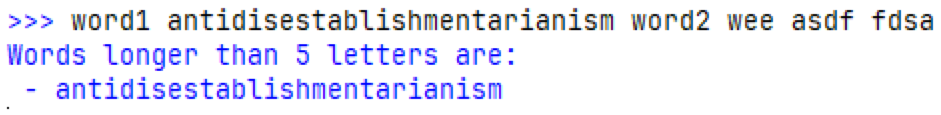
Data types used: String, List

Script:



```
r = input(">>> ").split()
def longWords(l: list):
    return [i for i in l if len(i) > 5]

print(f"Words longer than 5 letters are:\n - " + "\n - ".join(longWords(r)))
```

Output:



```
>>> word1 antidisestablishmentarianism word2 wee asdf fdsa
Words longer than 5 letters are:
 - antidisestablishmentarianism
```

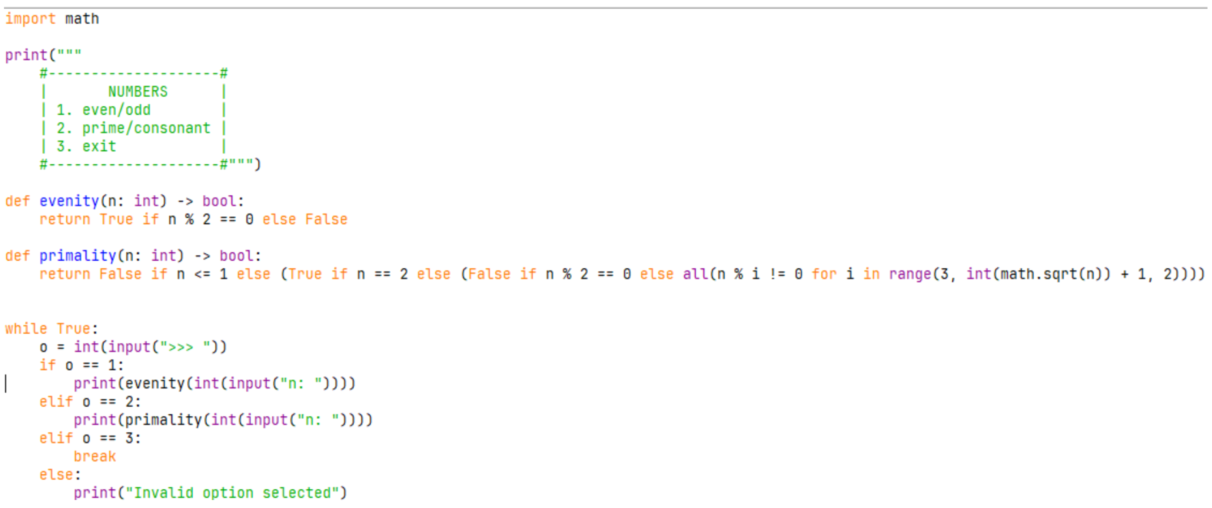
Program 6

Aim: To write a menu driven program to find odd/even numbers and prime numbers.

Modules used: math

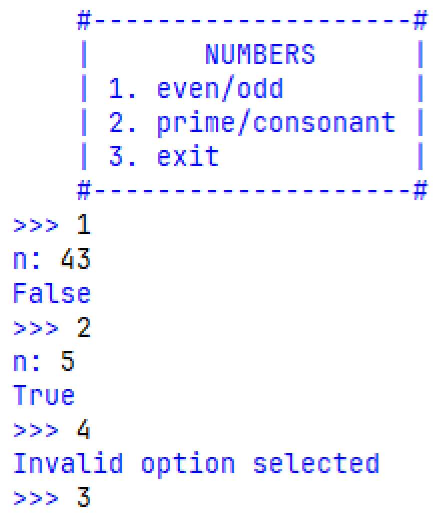
Data types used: Int, Bool

Script:



```
import math

print("""
    #--------------------#
    |       NUMBERS      |
    | 1. even/odd        |
    | 2. prime/consonant |
    | 3. exit            |
    #--------------------#""")

def evenity(n: int) -> bool:
    return True if n % 2 == 0 else False

def primality(n: int) -> bool:
    return False if n <= 1 else (True if n == 2 else (False if n % 2 == 0 else all(n % i != 0 for i in range(3, int(math.sqrt(n)) + 1, 2))))


while True:
    o = int(input(">>> "))
    if o == 1:
        print(evenity(int(input("n: "))))
    elif o == 2:
        print(primality(int(input("n: "))))
    elif o == 3:
        break
    else:
        print("Invalid option selected")
```

Output:



```
    #--------------------#
    |       NUMBERS      |
    | 1. even/odd        |
    | 2. prime/consonant |
    | 3. exit            |
    #--------------------#
>>> 1
n: 43
False
>>> 2
n: 5
True
>>> 4
Invalid option selected
>>> 3
```

Program 7

Aim: To write a function that returns the greater of two numbers.

Modules used: N/A

Data types used: Tuple, Float

Script:

```
a, b = tuple([*map(float, input(">>> ").split())])
def findBig() -> float:
    global a, b
    return a if a > b else b

print(f"Bigger number: {findBig()}")
```

Output:

```
>>> 1 3
Bigger number: 3.0
```

Program 8

Aim: To write a function that takes a list and moves all the elements divisible by 5 to the end of the list.

Modules used: N/A

Data types used: List, Int

Script:

```
x = [*map(int, input(">>> ").split())]
def move(l: list) -> None:
    l[:] = [x for x in l if x % 5] + [x for x in l if x % 5 == 0]
    print(f"Ordered list is: {l}")

move(x)
```

Output:

```
>>> 1 2 5 4 6 58 65 2350 15
Ordered list is: [1, 2, 4, 6, 58, 5, 65, 2350, 15]
```

Program 9

Aim: Given a dictionary containing information about vehicles, display the vehicles that were released in 2020 and order the dict in alphabetical order by brand name.

Modules used: N/A

Data types used: Dict, Int, List, String

Script:

```
vehicle = {
    "car1": ["Toyota", "Camry", 2020, 25_000],
    "car2": ["Ford", "Explorer", 2019, 32_000],
    "car3": ["Chevy", "Silverado", 2021, 40_000],
    "car4.5": ["Honda", "Civic", 2020, 22_000],
    "car5": ["anotherRealCarBrand", "Model nine", 2023, 45_000]
}

def _2020(D: dict):
    print(f"No. of vehicles released in 2020: {len([c for c in D if c[2] == 2020])}")

def sort(cars_dict):
    sorted_cars = sorted(cars_dict.items(), key=lambda x: x[1][0].lower())
    for key, value in sorted_cars:
        print(f"{key}: {value}")

_2020(vehicle)
print("\nSorted dict:")
sort(vehicle)
```

Output:

```
No. of vehicles released in 2020: 0

Sorted dict:
car5: ['anotherRealCarBrand', 'Model nine', 2023, 45000]
car3: ['Chevy', 'Silverado', 2021, 40000]
car2: ['Ford', 'Explorer', 2019, 32000]
car4.5: ['Honda', 'Civic', 2020, 22000]
car1: ['Toyota', 'Camry', 2020, 25000]
```

Program 10

Aim: To write a function that takes a tuple and returns the indices of the non-zero elements.

Modules used: N/A

Data types used: Int, Tuple, List

Script:

```python
t = tuple([*map(int, input(">>> ").split())])
def indexTuple(t: tuple) -> list:
    return [i for i, v in enumerate(t) if v != 0]

print(f"Non zero indices are: {indexTuple(t)}")
```

Output:

```
>>> 1 0 25 03 64 00 5
Non zero indices are: [0, 2, 3, 4, 6]
```

Program 11

Aim: To write a function to count the number of vowels in user input.

Modules used: N/A

Data types used: String

Script:

```
r = input(">>> ")
def vowelCount() -> None:
    global r
    print(f"No. of vowels: {sum(i in "AEIOUaeiou" for i in r)}")

vowelCount()
```

Output:

```
>>> hello my name is star.stalker9160
No. of vowels: 8
```

FILE HANDLING – 1

TEXT FILES

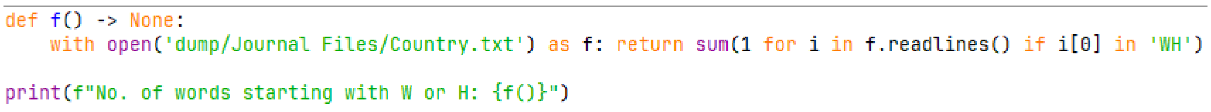
Program 1

Aim: Write a function to count the number of lines that start with the alphabet ‘W’ or ‘H’

Modules used: N/A

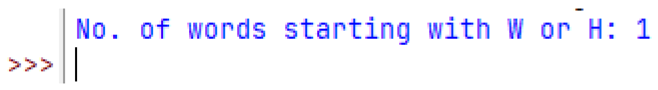
Data types used: Int, Str

Script:



```python
def f() -> None:
    with open('dump/Journal Files/Country.txt') as f: return sum(1 for i in f.readlines() if i[0] in 'WH')

print(f"No. of words starting with W or H: {f()}")
```

Output:



```
No. of words starting with W or H: 1
>>> 
```

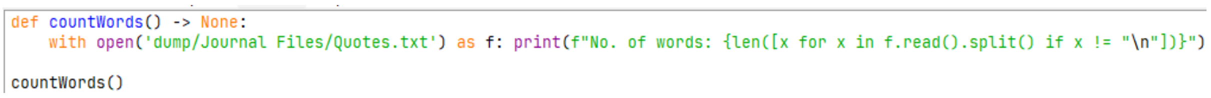
Program 2

Aim: Write a function countWords() to display total number of words in a file

Modules used: N/A

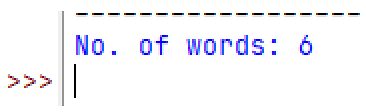
Data types used: Int, Str

Script:



```python
def countWords() -> None:
    with open('dump/Journal Files/Quotes.txt') as f: print(f"No. of words: {len([x for x in f.read().split() if x != "\n"])}")

countWords()
```

Output:



```
    -------------------
    No. of words: 6
>>> 
```

Program 3

Aim: Write a function filter(oldfile, newfile) that copies all lines from oldfile into newfile that don’t start with ‘@’

Modules used: N/A

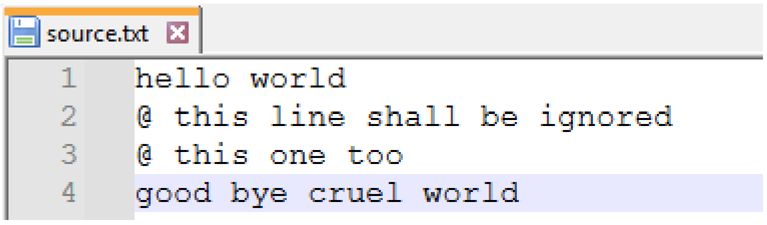
Data types used: Int, Str

Script:

```python
def filter(oldfile: str, newfile: str) -> None:
    with open(oldfile) as o, open(newfile, "w") as n:
        n.writelines([l for l in o if not l.startswith("@")])

filter("dump/Journal Files/source.txt", "dump/Journal Files/target.txt")
```

Output:

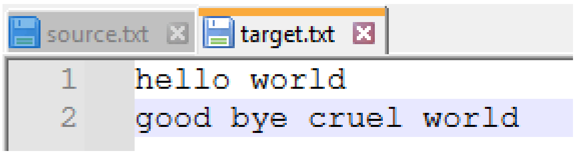
source.txt:



source.txt

```
hello world
@ this line shall be ignored
@ this one too
good bye cruel world
```

target.txt



source.txt target.txt

```
hello world
good bye cruel world
```

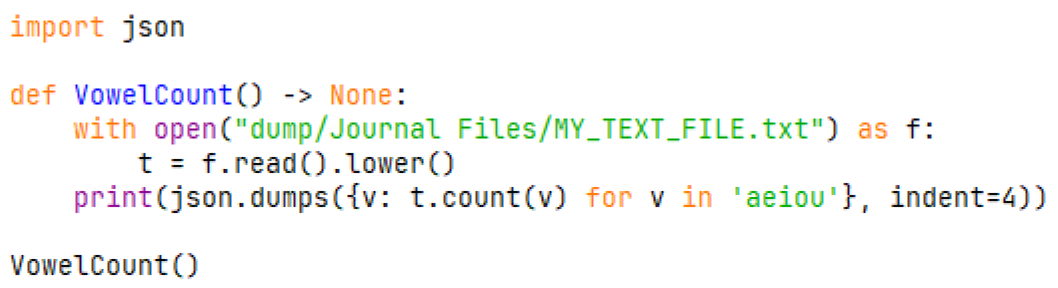
Program 4

Aim: Write a function VowelCount that displays the occurrence of vowels in a file

Modules used: json

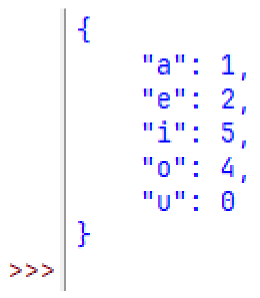
Data types used: Int, Str, Dict

Script:



```python
import json

def VowelCount() -> None:
    with open("dump/Journal Files/MY_TEXT_FILE.txt") as f:
        t = f.read().lower()
    print(json.dumps({v: t.count(v) for v in 'aeiou'}, indent=4))

VowelCount()
```

Output:



```
{
    "a": 1,
    "e": 2,
    "i": 5,
    "o": 4,
    "u": 0
}
>>>
```

Program 5

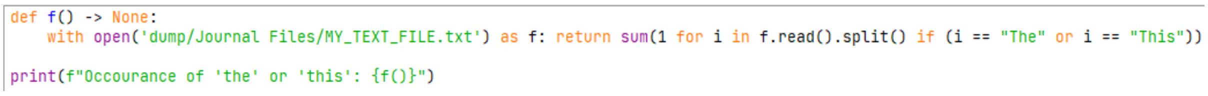
Aim: Write a function to count the occurrence of 'The' and 'This'

Modules used: N/A

Data types used: Int

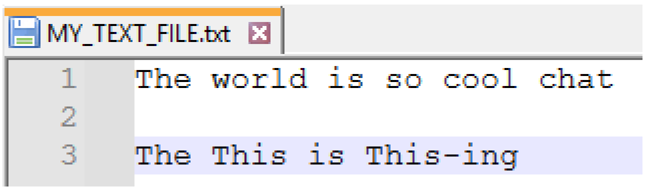
Script:

python:



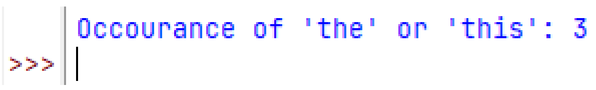
```python
def f() -> None:
    with open('dump/Journal Files/MY_TEXT_FILE.txt') as f: return sum(1 for i in f.read().split() if (i == "The" or i == "This"))

print(f"Occourance of 'the' or 'this': {f()}")
```

MY_TEXT_FILE.txt:



```
The world is so cool chat

The This is This-ing
```

Output:



```
Occourance of 'the' or 'this': 3
>>>
```

Program 6

Aim: Write a function ISTOUPCOUNT to count the occurrence of ‘IS’, ‘TO’ and ‘UP’ in a file

Modules used: json

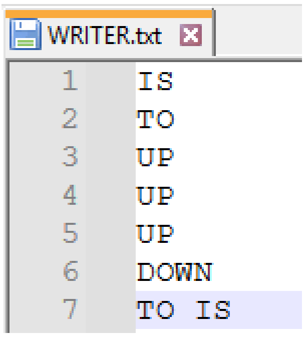
Data types used: Str, Dict, Int

Script:

python:

```python
import json

def ISTOUPCOUNT() -> None:
    with open("dump/Journal Files/WRITER.txt") as f:
        w = f.read().split()
        print(json.dumps({"IS": w.count("IS"), "TO": w.count("TO"), "UP": w.count("UP")}, indent=4))

ISTOUPCOUNT()
```

WRITER.txt:



```
IS
TO
UP
UP
UP
DOWN
TO IS
```

Output:

```
{
    "IS": 2,
    "TO": 2,
    "UP": 3
}
>>>
```

Program 7

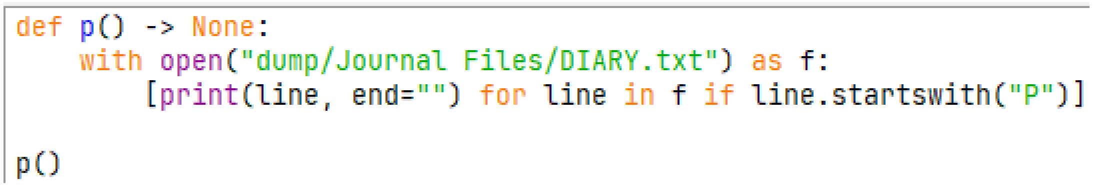
Aim: Write a function to print out the lines from a file that start with 'P'

Modules used: N/A

Data types used: Str, List

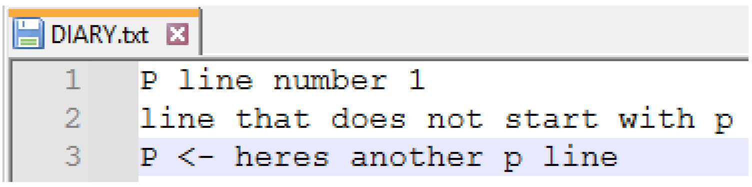
Script:

python:



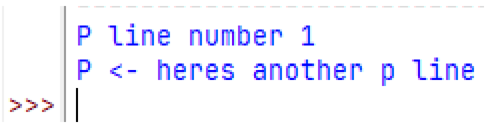
```python
def p() -> None:
    with open("dump/Journal Files/DIARY.txt") as f:
        [print(line, end="") for line in f if line.startswith("P")]

p()
```

DIARY.txt:



```
P line number 1
line that does not start with p
P <- heres another p line
```

Output:



```
P line number 1
P <- heres another p line
>>> 
```

Program 8

Aim: Write a function to display the number of lines starting with ‘H’

Modules used: N/A

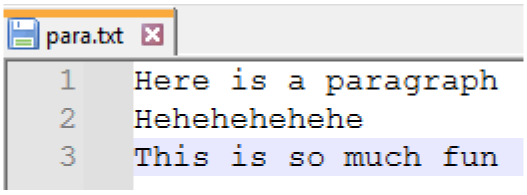
Data types used: Int

Script:

python:

```python
def h() -> None:
    with open("dump/Journal Files/para.txt") as f:
        print(sum(1 for l in f if l.startswith("H")))

h()
```

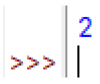
para.txt:



para.txt

```
Here is a paragraph
Hehehehehehe
This is so much fun
```

Output:



```
2
>>> 
```

Program 9

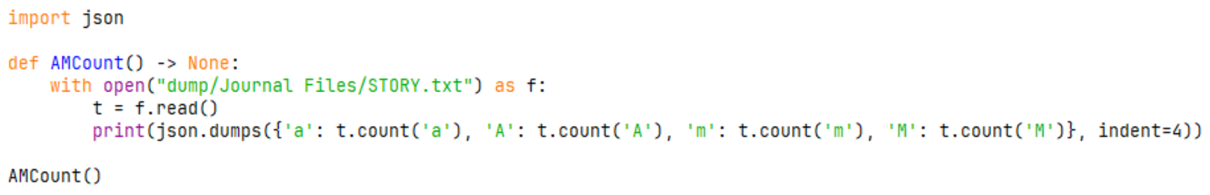
Aim: Write a function AMCount to count the occurrences of ‘a’ and ‘m’ both upper and lower case

Modules used: json

Data types used: Str, Int, Dict

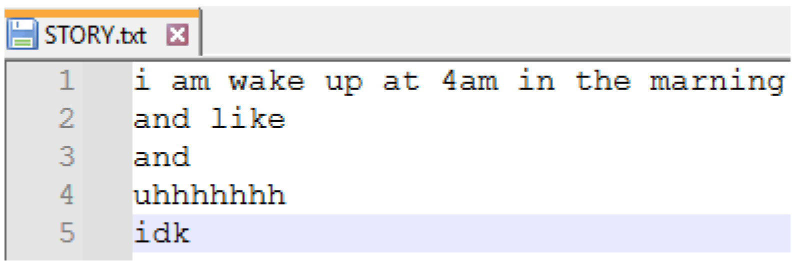
Script:

python:



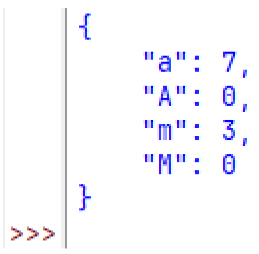
```python
import json

def AMCount() -> None:
    with open("dump/Journal Files/STORY.txt") as f:
        t = f.read()
        print(json.dumps({'a': t.count('a'), 'A': t.count('A'), 'm': t.count('m'), 'M': t.count('M')}, indent=4))

AMCount()
```

STORY.txt:



```
i am wake up at 4am in the marning
and like
and
uhhhhhhh
idk
```

Output:



```
{
    "a": 7,
    "A": 0,
    "m": 3,
    "M": 0
}
>>>
```

Program 10

Aim: Write a function COUNT to count the occurrence of 'Catholic' and 'mother'

Modules used: json

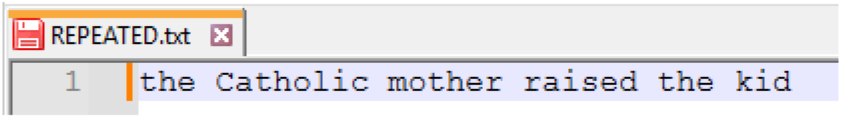
Data types used: Int, Str, Dict

Script:

python:

```python
import json

def COUNT():
    with open("dump/Journal Files/REPEATED.txt") as f:
        t = f.read().split()
        print(json.dumps({"Catholic": t.count("Catholic"), "mother": t.count("mother")}, indent=4))

COUNT()
```

REPEATED.txt:



REPEATED.txt

```
the Catholic mother raised the kid
```

Output:

```
{
    "Catholic": 1,
    "mother": 1
}
>>>
```

Program 11

Aim: Write a function to print out the lines that have only 2 chars

Modules used: N/A

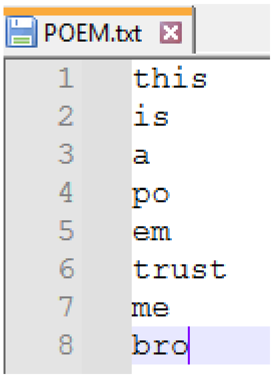
Data types used: Int, List, Str

Script:

python:

```python
def _2chars():
    with open("dump/Journal Files/POEM.txt") as f:
        [print(w) for l in f for w in l.split() if len(w) == 2]

_2chars()
```

POEM.txt:



POEM.txt

```
this
is
a
po
em
trust
me
bro
```

Output:

```
is
po
em
me
>>>
```

Program 12

Aim: Write a function COUNT_AND to count the occurrence of ‘and’ (case insensitive)

Modules used: N/A

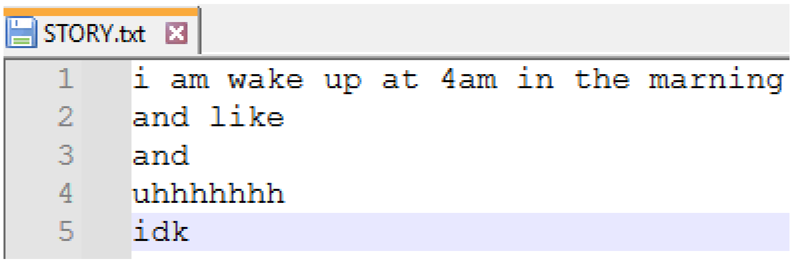
Data types used: Int, Str

Script:

python:

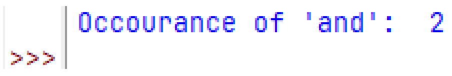
```python
def COUNT_AND():
    with open("dump/Journal Files/STORY.txt") as f:
        text = f.read().lower()
        print("Occourance of 'and': ", text.count("and"))

COUNT_AND()
```

STORY.txt:



```
i am wake up at 4am in the marning
and like
and
uhhhhhhh
idk
```

Output:



```
Occourance of 'and':  2
>>>
```

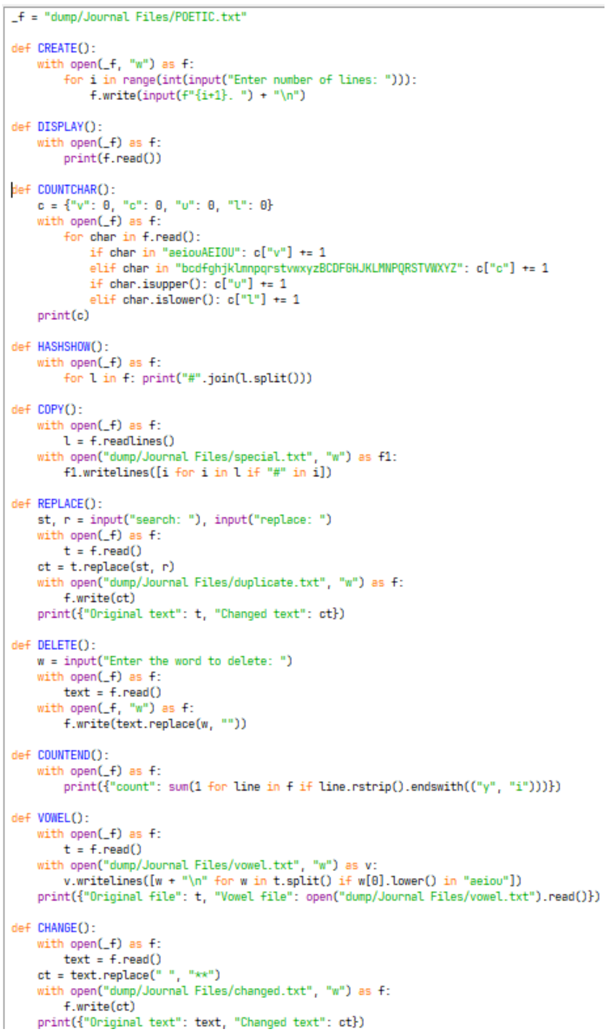
Program 13

Aim: Write a menu driven program to perform operations on a file

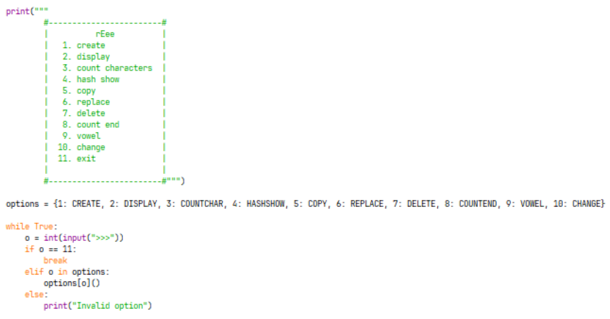
Modules used: N/A

Data types used: Int, List, Str

Script:

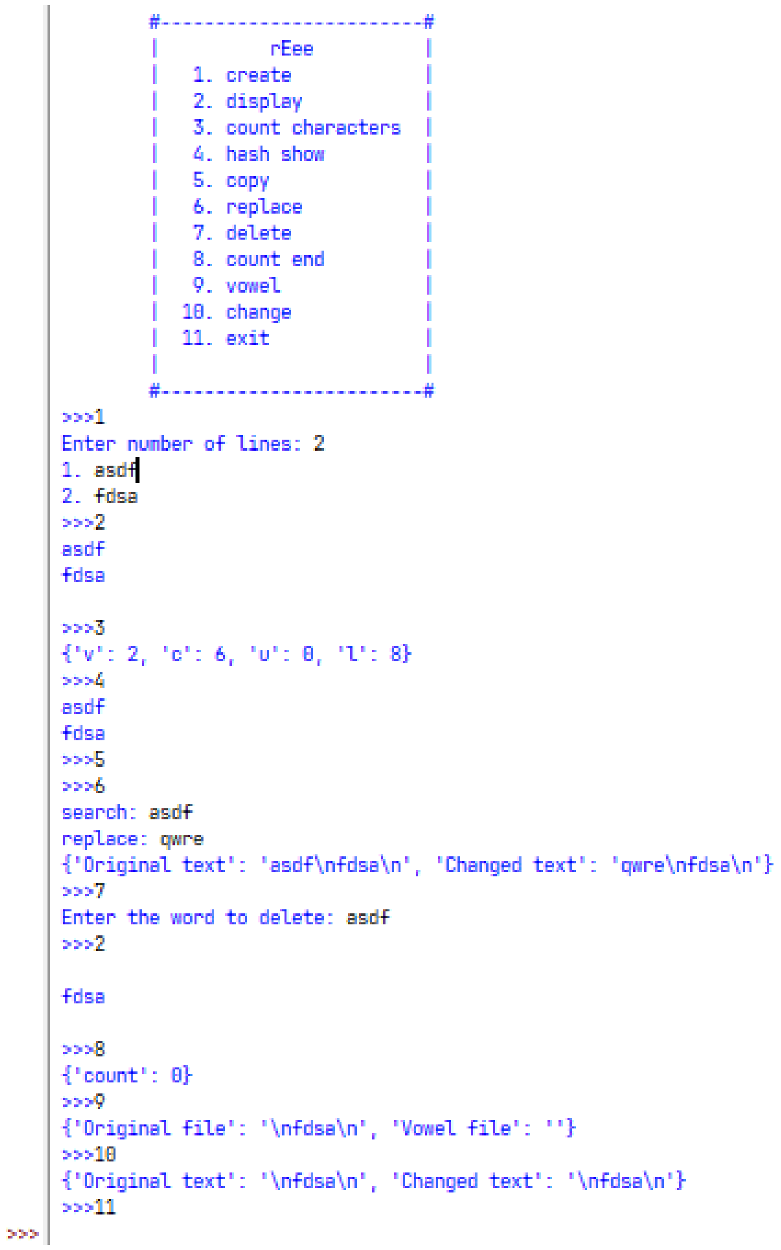


```python
_f = "dump/Journal Files/POETIC.txt"

def CREATE():
    with open(_f, "w") as f:
        for i in range(int(input("Enter number of lines: "))):
            f.write(input(f"{i+1}. ") + "\n")

def DISPLAY():
    with open(_f) as f:
        print(f.read())

def COUNTCHAR():
    c = {"v": 0, "c": 0, "u": 0, "l": 0}
    with open(_f) as f:
        for char in f.read():
            if char in "aeiouAEIOU": c["v"] += 1
            elif char in "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ": c["c"] += 1
            if char.isupper(): c["u"] += 1
            elif char.islower(): c["l"] += 1
    print(c)

def HASHSHOW():
    with open(_f) as f:
        for l in f: print("#".join(l.split()))

def COPY():
    with open(_f) as f:
        l = f.readlines()
    with open("dump/Journal Files/special.txt", "w") as f1:
        f1.writelines([i for i in l if "#" in i])

def REPLACE():
    st, r = input("search: "), input("replace: ")
    with open(_f) as f:
        t = f.read()
    ct = t.replace(st, r)
    with open("dump/Journal Files/duplicate.txt", "w") as f:
        f.write(ct)
    print({"Original text": t, "Changed text": ct})

def DELETE():
    w = input("Enter the word to delete: ")
    with open(_f) as f:
        text = f.read()
    with open(_f, "w") as f:
        f.write(text.replace(w, ""))

def COUNTEND():
    with open(_f) as f:
        print({"count": sum(1 for line in f if line.rstrip().endswith(("y", "i")))})

def VOWEL():
    with open(_f) as f:
        t = f.read()
    with open("dump/Journal Files/vowel.txt", "w") as v:
        v.writelines([w + "\n" for w in t.split() if w[0].lower() in "aeiou"])
    print({"Original file": t, "Vowel file": open("dump/Journal Files/vowel.txt").read()})

def CHANGE():
    with open(_f) as f:
        text = f.read()
    ct = text.replace(" ", "**")
    with open("dump/Journal Files/changed.txt", "w") as f:
        f.write(ct)
    print({"Original text": text, "Changed text": ct})
```



```python
print("""
        #----------------------#
        |         rEee         |
        |   1. create          |
        |   2. display         |
        |   3. count characters|
        |   4. hash show       |
        |   5. copy            |
        |   6. replace         |
        |   7. delete          |
        |   8. count end       |
        |   9. vowel           |
        |  10. change          |
        |  11. exit            |
        |                      |
        #----------------------#""")

options = {1: CREATE, 2: DISPLAY, 3: COUNTCHAR, 4: HASHSHOW, 5: COPY, 6: REPLACE, 7: DELETE, 8: COUNTEND, 9: VOWEL, 10: CHANGE}

while True:
    o = int(input(">>>"))
    if o == 11:
        break
    elif o in options:
        options[o]()
    else:
        print("Invalid option")
```

Output:


```
         #------------------------#
         |          rEee          |
         |   1. create            |
         |   2. display           |
         |   3. count characters  |
         |   4. hash show         |
         |   5. copy              |
         |   6. replace           |
         |   7. delete            |
         |   8. count end         |
         |   9. vowel             |
         |  10. change            |
         |  11. exit              |
         |                        |
         #------------------------#
>>>1
Enter number of lines: 2
1. asdf
2. fdsa
>>>2
asdf
fdsa

>>>3
{'v': 2, 'c': 6, 'u': 0, 'l': 8}
>>>4
asdf
fdsa
>>>5
>>>6
search: asdf
replace: qwre
{'Original text': 'asdf\nfdsa\n', 'Changed text': 'qwre\nfdsa\n'}
>>>7
Enter the word to delete: asdf
>>>2

fdsa

>>>8
{'count': 0}
>>>9
{'Original file': '\nfdsa\n', 'Vowel file': ''}
>>>10
{'Original text': '\nfdsa\n', 'Changed text': '\nfdsa\n'}
>>>11
>>>
```

FILE HANDLING – 2

BINARY FILES

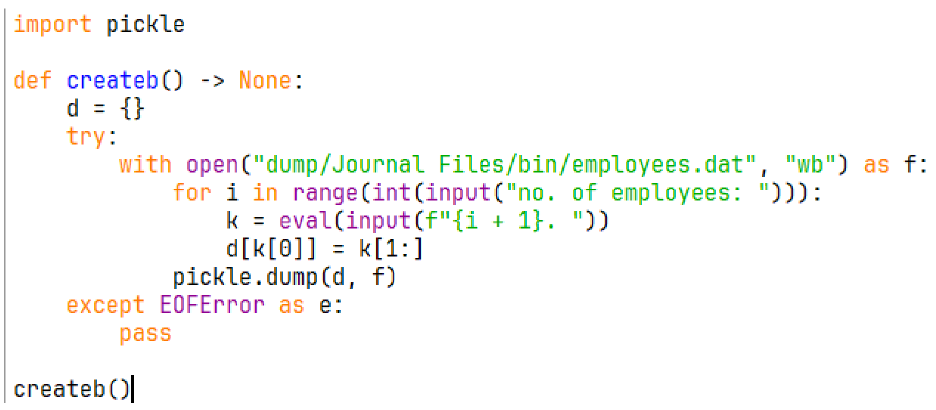
Program 1

Aim: Write a function ‘createb()’ to create a binary file with entered details and ‘Search()’ to look for a specific employee

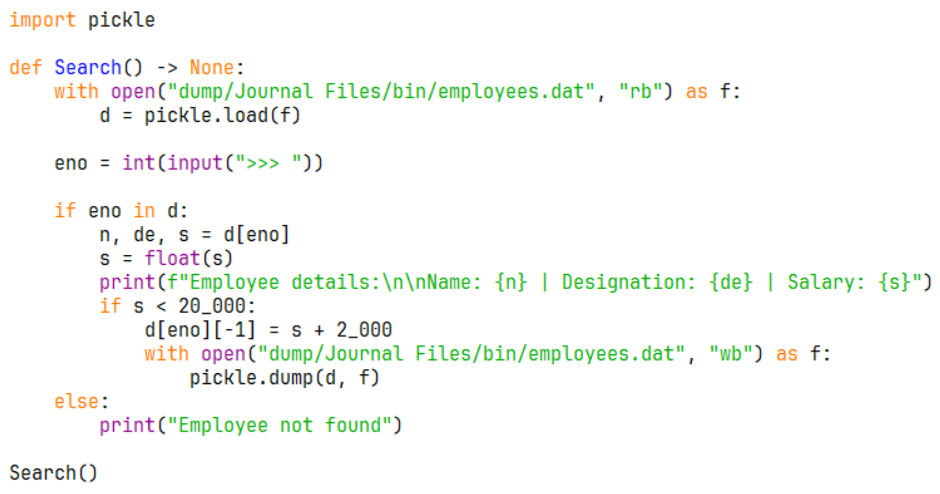
Modules used: pickle

Data types used: Int, Str, Dict, List

Script:



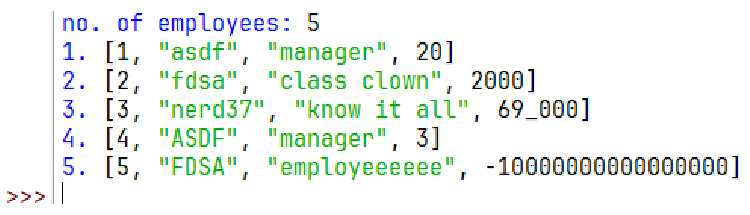
```
import pickle

def createb() -> None:
    d = {}
    try:
        with open("dump/Journal Files/bin/employees.dat", "wb") as f:
            for i in range(int(input("no. of employees: "))):
                k = eval(input(f"{i + 1}. "))
                d[k[0]] = k[1:]
            pickle.dump(d, f)
    except EOFError as e:
        pass

createb()
```



```
import pickle

def Search() -> None:
    with open("dump/Journal Files/bin/employees.dat", "rb") as f:
        d = pickle.load(f)

    eno = int(input(">>> "))

    if eno in d:
        n, de, s = d[eno]
        s = float(s)
        print(f"Employee details:\n\nName: {n} | Designation: {de} | Salary: {s}")
        if s < 20_000:
            d[eno][-1] = s + 2_000
            with open("dump/Journal Files/bin/employees.dat", "wb") as f:
                pickle.dump(d, f)
    else:
        print("Employee not found")

Search()
```

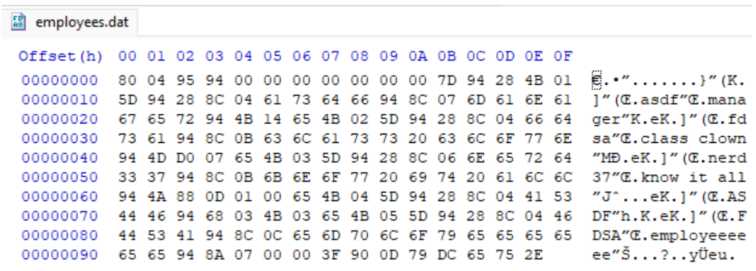
Output:

python (function 1):



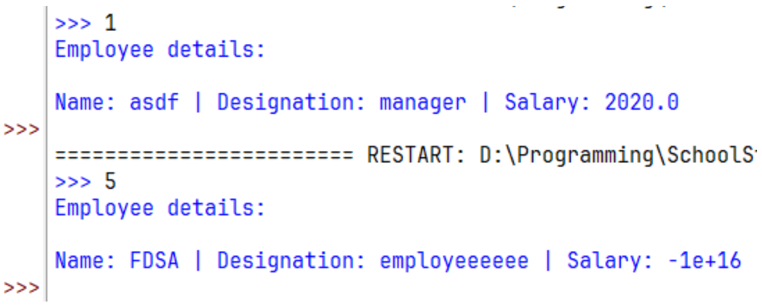
```
    no. of employees: 5
    1. [1, "asdf", "manager", 20]
    2. [2, "fdsa", "class clown", 2000]
    3. [3, "nerd37", "know it all", 69_000]
    4. [4, "ASDF", "manager", 3]
    5. [5, "FDSA", "employeeeeee", -10000000000000000]
>>> 
```

employee.dat (hex editor, function 1):



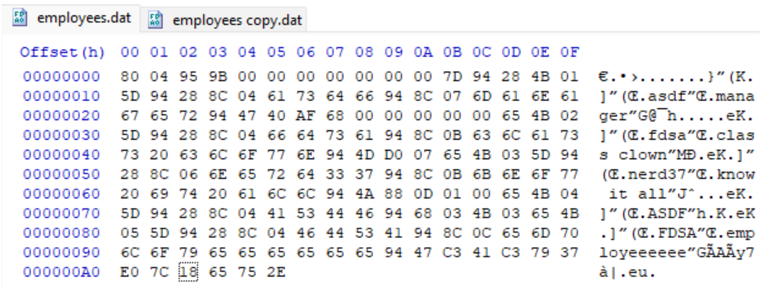
```
employees.dat

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  80 04 95 94 00 00 00 00 00 00 00 7D 94 28 4B 01  €.•”.......}”(K.
00000010  5D 94 28 8C 04 61 73 64 66 94 8C 07 6D 61 6E 61  ]”(Œ.asdf”Œ.mana
00000020  67 65 72 94 4B 14 65 4B 02 5D 94 28 8C 04 66 64  ger”K.eK.]”(Œ.fd
00000030  73 61 94 8C 0B 63 6C 61 73 73 20 63 6C 6F 77 6E  sa”Œ.class clown
00000040  94 4D D0 07 65 4B 03 5D 94 28 8C 06 6E 65 72 64  ”MÐ.eK.]”(Œ.nerd
00000050  33 37 94 8C 0B 6B 6E 6F 77 20 69 74 20 61 6C 6C  37”Œ.know it all
00000060  94 4A 88 0D 01 00 65 4B 04 5D 94 28 8C 04 41 53  ”Jˆ...eK.]”(Œ.AS
00000070  44 46 94 68 03 4B 03 65 4B 05 5D 94 28 8C 04 46  DF”h.K.eK.]”(Œ.F
00000080  44 53 41 94 8C 0C 65 6D 70 6C 6F 79 65 65 65 65  DSA”Œ.employeeee
00000090  65 65 94 8A 07 00 00 3F 90 0D 79 DC 65 75 2E     ee”Š...?..yÜeu.
```

python (function 2):



```
>>> 1
Employee details:

Name: asdf | Designation: manager | Salary: 2020.0
>>>
======================== RESTART: D:\Programming\SchoolS
>>> 5
Employee details:

Name: FDSA | Designation: employeeeeee | Salary: -1e+16
>>>
```

employee.dat (hex editor, function 2):



```
employees.dat    employees copy.dat

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  80 04 95 9B 00 00 00 00 00 00 00 7D 94 28 4B 01  €.•›.......}”(K.
00000010  5D 94 28 8C 04 61 73 64 66 94 8C 07 6D 61 6E 61  ]”(Œ.asdf”Œ.mana
00000020  67 65 72 94 47 40 AF 68 00 00 00 00 00 65 4B 02  ger”G@¯h.....eK.
00000030  5D 94 28 8C 04 66 64 73 61 94 8C 0B 63 6C 61 73  ]”(Œ.fdsa”Œ.clas
00000040  73 20 63 6C 6F 77 6E 94 4D D0 07 65 4B 03 5D 94  s clown”MÐ.eK.]”
00000050  28 8C 06 6E 65 72 64 33 37 94 8C 0B 6B 6E 6F 77  (Œ.nerd37”Œ.know
00000060  20 69 74 20 61 6C 6C 94 4A 88 0D 01 00 65 4B 04   it all”Jˆ...eK.
00000070  5D 94 28 8C 04 41 53 44 46 94 68 03 4B 03 65 4B  ]”(Œ.ASDF”h.K.eK
00000080  05 5D 94 28 8C 04 46 44 53 41 94 8C 0C 65 6D 70  .]”(Œ.FDSA”Œ.emp
00000090  6C 6F 79 65 65 65 65 65 65 94 47 C3 41 C3 79 37  loyeeeeee”GÃAÃy7
000000A0  E0 7C 18 65 75 2E                                à|.eu.
```

Program 2

Aim: Write a function to insert data at the end of the file

Modules used: pickle

Data types used: Int, Str, Dict, List

Script:

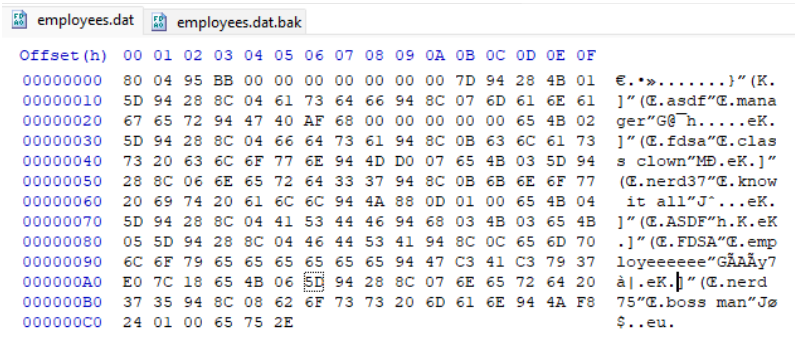
```python
import pickle

def f() -> None:
    with open("dump/Journal Files/bin/employees.dat", "rb") as f:
        d = pickle.load(f)

    for i in range(int(input("no. of new employees: "))):
        k = eval(input(f"{i + 1}. "))
        d[k[0]] = k[1:]

    with open("dump/Journal Files/bin/employees.dat", "wb") as f:
        pickle.dump(d, f)

f()
```

Output:

python:

```
no. of new employees: 1
1. [6, "nerd 75", "boss man", 75_000]
>>>
```

employee.dat (hex editor):



employees.dat | employees.dat.bak

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  80 04 95 BB 00 00 00 00 00 00 00 7D 94 28 4B 01  €.•».......}”(K.
00000010  5D 94 28 8C 04 61 73 64 66 94 8C 07 6D 61 6E 61  ]”(Œ.asdf”Œ.mana
00000020  67 65 72 94 47 40 AF 68 00 00 00 00 00 65 4B 02  ger”G@¯h.....eK.
00000030  5D 94 28 8C 04 66 64 73 61 94 8C 0B 63 6C 61 73  ]”(Œ.fdsa”Œ.clas
00000040  73 20 63 6C 6F 77 6E 94 4D D0 07 65 4B 03 5D 94  s clown”MÐ.eK.]”
00000050  28 8C 06 6E 65 72 64 33 37 94 8C 0B 6B 6E 6F 77  (Œ.nerd37”Œ.know
00000060  20 69 74 20 61 6C 6C 94 4A 88 0D 01 00 65 4B 04   it all”Jˆ...eK.
00000070  5D 94 28 8C 04 41 53 44 46 94 68 03 4B 03 65 4B  ]”(Œ.ASDF”h.K.eK
00000080  05 5D 94 28 8C 04 46 44 53 41 94 8C 0C 65 6D 70  .]”(Œ.FDSA”Œ.emp
00000090  6C 6F 79 65 65 65 65 65 65 94 47 C3 41 C3 79 37  loyeeeeee”GÃAÃy7
000000A0  E0 7C 18 65 4B 06 5D 94 28 8C 07 6E 65 72 64 20  à|.eK.]”(Œ.nerd 
000000B0  37 35 94 8C 08 62 6F 73 73 20 6D 61 6E 94 4A F8  75”Œ.boss man”Jø
000000C0  24 01 00 65 75 2E                                $..eu.
```

Program 3

Aim: Split `employee.dat` into two files, one with managers & the other with employees

Modules used: pickle

Data types used: Int, Str, Dict, List

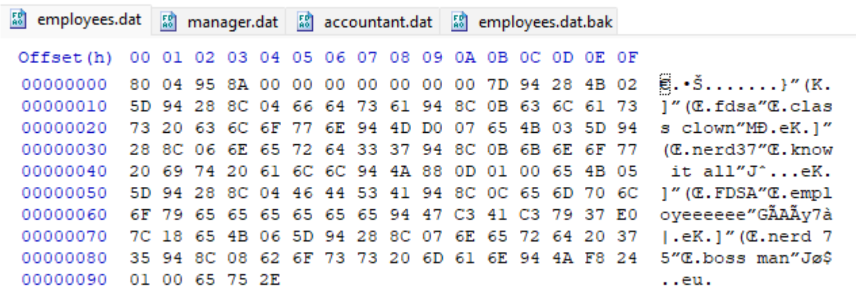
Script:

```
import pickle

def split() -> None:
    with open("dump/Journal Files/bin/employees.dat", "rb") as f1, \
         open("dump/Journal Files/bin/manager.dat", "ab") as f2, \
         open("dump/Journal Files/bin/accountant.dat", "ab") as f3:

        d = pickle.load(f1)
        x = {}

        for k, v in d.items():
            de = v[1].lower()
            if de == "manager":
                pickle.dump({k: v}, f2)
            elif de == "accountant":
                pickle.dump({k: v}, f3)
            else:
                x[k] = v

    with open("dump/Journal Files/bin/employees.dat", "wb") as f1:
        pickle.dump(x, f1)

split()
```

Output:

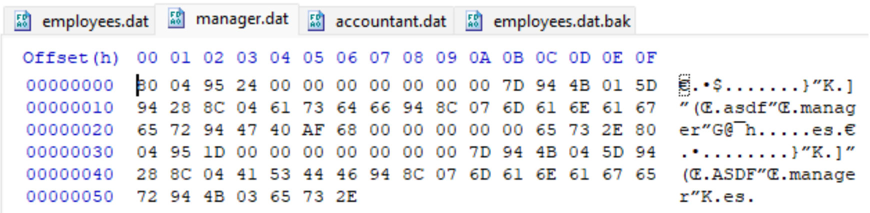
python – no output

employee.dat (hex editor):



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  80 04 95 8A 00 00 00 00 00 00 00 7D 94 28 4B 02  €.•Š.......}”(K.
00000010  5D 94 28 8C 04 66 64 73 61 94 8C 0B 63 6C 61 73  ]”(Œ.fdsa”Œ.clas
00000020  73 20 63 6C 6F 77 6E 94 4D D0 07 65 4B 03 5D 94  s clown”MÐ.eK.]”
00000030  28 8C 06 6E 65 72 64 33 37 94 8C 0B 6B 6E 6F 77  (Œ.nerd37”Œ.know
00000040  20 69 74 20 61 6C 6C 94 4A 88 0D 01 00 65 4B 05   it all”Jˆ...eK.
00000050  5D 94 28 8C 04 46 44 53 41 94 8C 0C 65 6D 70 6C  ]”(Œ.FDSA”Œ.empl
00000060  6F 79 65 65 65 65 65 65 94 47 C3 41 C3 79 37 E0  oyeeeeee”GÃAÃy7à
00000070  7C 18 65 4B 06 5D 94 28 8C 07 6E 65 72 64 20 37  |.eK.]”(Œ.nerd 7
00000080  35 94 8C 08 62 6F 73 73 20 6D 61 6E 94 4A F8 24  5”Œ.boss man”Jø$
00000090  01 00 65 75 2E                                   ..eu.
```

manager.dat (hex editor):



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  80 04 95 24 00 00 00 00 00 00 00 7D 94 4B 01 5D  €.•$.......}”K.]
00000010  94 28 8C 04 61 73 64 66 94 8C 07 6D 61 6E 61 67  ”(Œ.asdf”Œ.manag
00000020  65 72 94 47 40 AF 68 00 00 00 00 00 65 73 2E 80  er”G@¯h.....es.€
00000030  04 95 1D 00 00 00 00 00 00 00 7D 94 4B 04 5D 94  .•........}”K.]”
00000040  28 8C 04 41 53 44 46 94 8C 07 6D 61 6E 61 67 65  (Œ.ASDF”Œ.manage
00000050  72 94 4B 03 65 73 2E                             r”K.es.
```

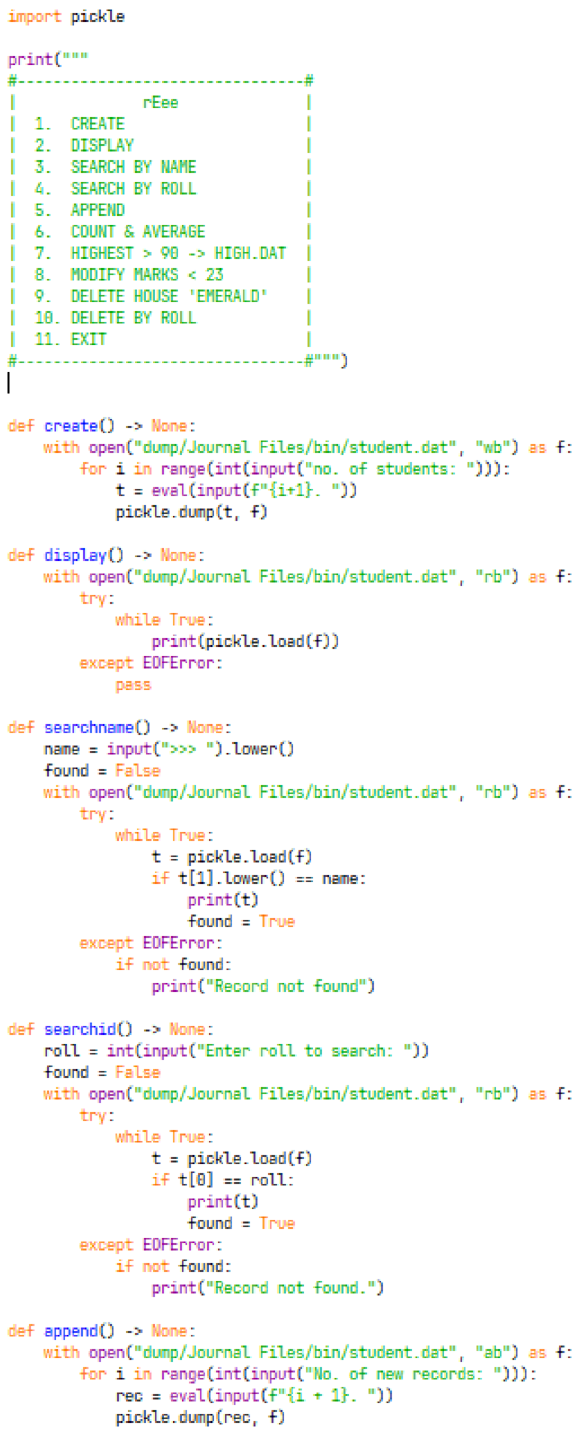
Program 4

Aim: Write a menu driven program to perform operations on a file

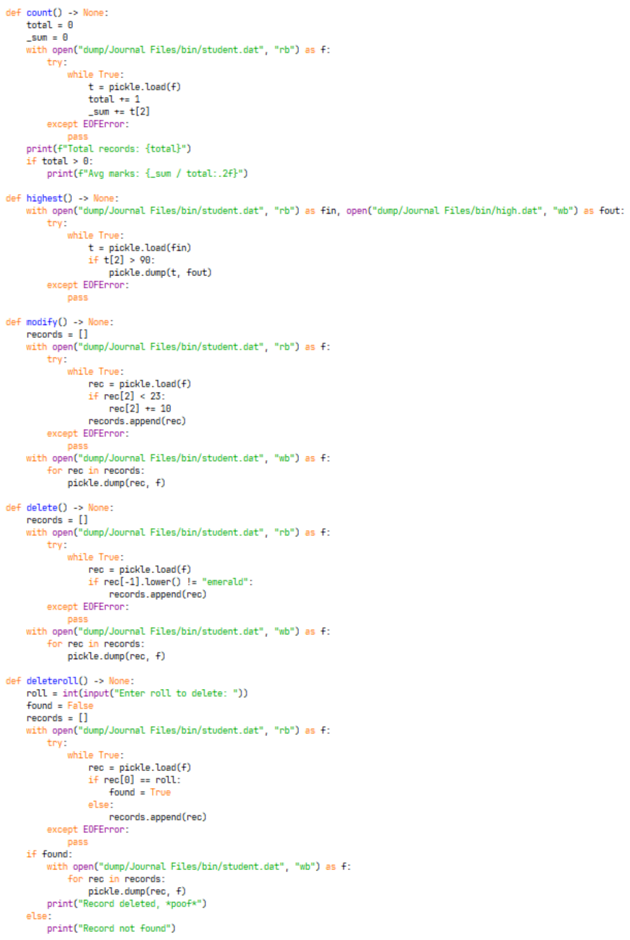
Modules used: pickle

Data types used: Int, Str, Dict, List

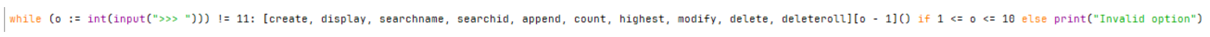
Script:



```python
import pickle

print("""
#--------------------------------#
|             rEee               |
|  1.  CREATE                    |
|  2.  DISPLAY                   |
|  3.  SEARCH BY NAME            |
|  4.  SEARCH BY ROLL            |
|  5.  APPEND                    |
|  6.  COUNT & AVERAGE           |
|  7.  HIGHEST > 90 -> HIGH.DAT  |
|  8.  MODIFY MARKS < 23         |
|  9.  DELETE HOUSE 'EMERALD'    |
|  10. DELETE BY ROLL            |
|  11. EXIT                      |
#--------------------------------#""")


def create() -> None:
    with open("dump/Journal Files/bin/student.dat", "wb") as f:
        for i in range(int(input("no. of students: "))):
            t = eval(input(f"{i+1}. "))
            pickle.dump(t, f)

def display() -> None:
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except EOFError:
            pass

def searchname() -> None:
    name = input(">>> ").lower()
    found = False
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                t = pickle.load(f)
                if t[1].lower() == name:
                    print(t)
                    found = True
        except EOFError:
            if not found:
                print("Record not found")

def searchid() -> None:
    roll = int(input("Enter roll to search: "))
    found = False
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                t = pickle.load(f)
                if t[0] == roll:
                    print(t)
                    found = True
        except EOFError:
            if not found:
                print("Record not found.")

def append() -> None:
    with open("dump/Journal Files/bin/student.dat", "ab") as f:
        for i in range(int(input("No. of new records: "))):
            rec = eval(input(f"{i + 1}. "))
            pickle.dump(rec, f)
```

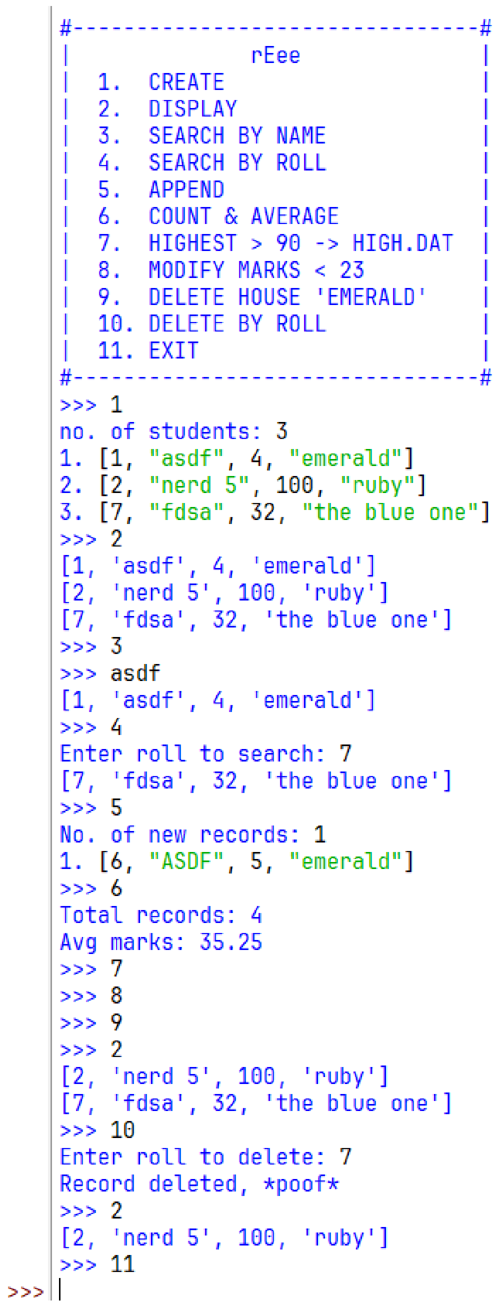


```python
def count() -> None:
    total = 0
    _sum = 0
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                t = pickle.load(f)
                total += 1
                _sum += t[2]
        except EOFError:
            pass
    print(f"Total records: {total}")
    if total > 0:
        print(f"Avg marks: {_sum / total:.2f}")

def highest() -> None:
    with open("dump/Journal Files/bin/student.dat", "rb") as fin, open("dump/Journal Files/bin/high.dat", "wb") as fout:
        try:
            while True:
                t = pickle.load(fin)
                if t[2] > 90:
                    pickle.dump(t, fout)
        except EOFError:
            pass

def modify() -> None:
    records = []
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                rec = pickle.load(f)
                if rec[2] < 23:
                    rec[2] += 10
                records.append(rec)
        except EOFError:
            pass
    with open("dump/Journal Files/bin/student.dat", "wb") as f:
        for rec in records:
            pickle.dump(rec, f)

def delete() -> None:
    records = []
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                rec = pickle.load(f)
                if rec[-1].lower() != "emerald":
                    records.append(rec)
        except EOFError:
            pass
    with open("dump/Journal Files/bin/student.dat", "wb") as f:
        for rec in records:
            pickle.dump(rec, f)

def deleteroll() -> None:
    roll = int(input("Enter roll to delete: "))
    found = False
    records = []
    with open("dump/Journal Files/bin/student.dat", "rb") as f:
        try:
            while True:
                rec = pickle.load(f)
                if rec[0] == roll:
                    found = True
                else:
                    records.append(rec)
        except EOFError:
            pass
    if found:
        with open("dump/Journal Files/bin/student.dat", "wb") as f:
            for rec in records:
                pickle.dump(rec, f)
        print("Record deleted, *poof*")
    else:
        print("Record not found")
```



```python
while (o := int(input(">>> "))) != 11: [create, display, searchname, searchid, append, count, highest, modify, delete, deleteroll][o - 1]() if 1 <= o <= 10 else print("Invalid option")
```

Output:



```
#---------------------------------#
|               rEee              |
|  1.  CREATE                     |
|  2.  DISPLAY                    |
|  3.  SEARCH BY NAME             |
|  4.  SEARCH BY ROLL             |
|  5.  APPEND                     |
|  6.  COUNT & AVERAGE            |
|  7.  HIGHEST > 90 -> HIGH.DAT   |
|  8.  MODIFY MARKS < 23          |
|  9.  DELETE HOUSE 'EMERALD'     |
|  10. DELETE BY ROLL             |
|  11. EXIT                       |
#---------------------------------#
>>> 1
no. of students: 3
1. [1, "asdf", 4, "emerald"]
2. [2, "nerd 5", 100, "ruby"]
3. [7, "fdsa", 32, "the blue one"]
>>> 2
[1, 'asdf', 4, 'emerald']
[2, 'nerd 5', 100, 'ruby']
[7, 'fdsa', 32, 'the blue one']
>>> 3
>>> asdf
[1, 'asdf', 4, 'emerald']
>>> 4
Enter roll to search: 7
[7, 'fdsa', 32, 'the blue one']
>>> 5
No. of new records: 1
1. [6, "ASDF", 5, "emerald"]
>>> 6
Total records: 4
Avg marks: 35.25
>>> 7
>>> 8
>>> 9
>>> 2
[2, 'nerd 5', 100, 'ruby']
[7, 'fdsa', 32, 'the blue one']
>>> 10
Enter roll to delete: 7
Record deleted, *poof*
>>> 2
[2, 'nerd 5', 100, 'ruby']
>>> 11
>>> |
```

FILE HANDLING – 3

CSV FILES

Program 1

Aim: Write a menu driven program to perform operations on a file

Modules used: csv

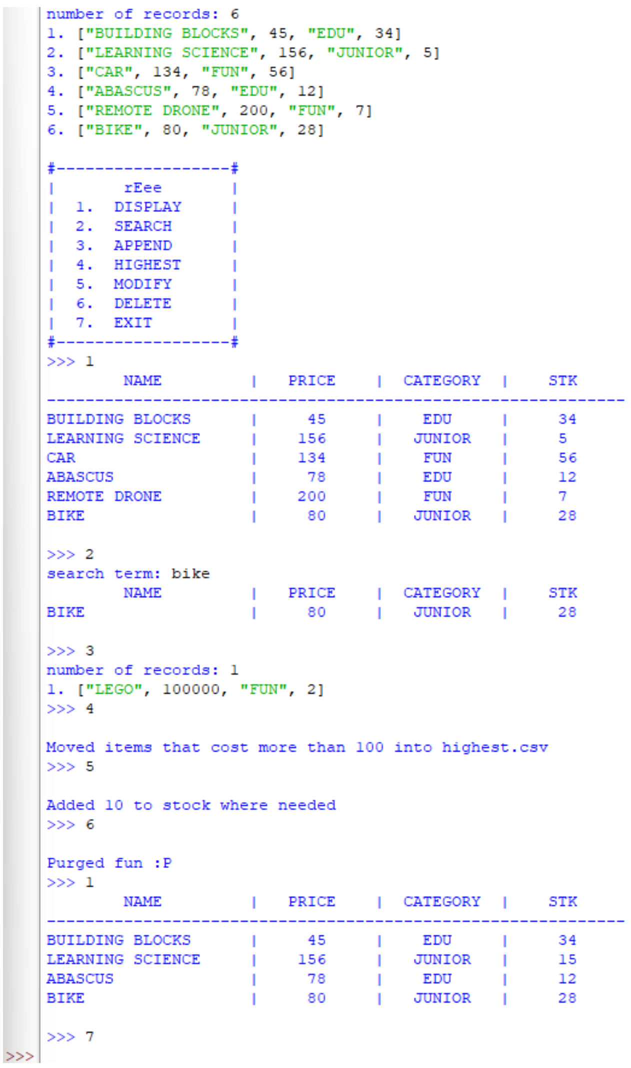
Data types used: Int, Str, List

Script:

```
from csv import writer as wr, reader as re

k = []
def _load():
    global k
    k.clear()
    with open("dump/Journal Files/csv/toy.csv", "r", newline="") as f:
        for i , j in enumerate(re(f)):
            if i == 0:
                continue
            k.append(j)

def CREATE():
    with open("dump/Journal Files/csv/toy.csv", "w", newline="") as f:
        w = wr(f)
        w.writerow(["NAME", "PRICE", "CATEGORY", "STK"])
        for i in range(int(input("number of records: "))):
            w.writerow(eval(input(f"{i + 1}. ")))
    _load()

def DISPLAY():
    with open("dump/Journal Files/csv/toy.csv", "r", newline="") as f:
        for i , j in enumerate(re(f)):
            if i == 0:
                print(f"{j[0]:^20} | {j[1]:^10} | {j[2]:^10} | {j[3]:^10}")
                print("-"*60)
                continue
            print(f"{j[0]:<20} | {j[1]:^10} | {j[2]:^10} | {j[3]:^10}")
    print()

def SEARCH():
    t = input("search term: ").lower()
    for i in k:
        if i[0].lower() == t:
            print("        NAME          |   PRICE    |  CATEGORY  |    STK    ")
            print(f"{i[0]:<20} | {i[1]:^10} | {i[2]:^10} | {i[3]:^10}")
            print()
            return
    print("Record not found")
    print()

def APPEND():
    with open("dump/Journal Files/csv/toy.csv", "a", newline="") as f:
        w = wr(f)
        for i in range(int(input("number of records: "))):
            w.writerow(eval(input(f"{i + 1}. ")))
    _load()

def HIGHEST():
    _load()
    l = []
    with open("dump/Journal Files/csv/highest.csv", "w", newline="") as f:
        for i in k:
            if int(i[1]) > 100:
                l.append(i)
        w = wr(f)
        w.writerow(["NAME", "PRICE", "CATEGORY", "STK"])
        w.writerows(l)
    print("\nMoved items that cost more than 100 into highest.csv")
```

```
def MODIFY():
    _load()
    l = []
    for i in k:
        if int(i[-1]) < 10:
            i[-1] = int(i[-1]) + 10
        l.append(i)
    with open("dump/Journal Files/csv/toy.csv", "w", newline="") as f:
        w = wr(f)
        w.writerow(["NAME", "PRICE", "CATEGORY", "STK"])
        w.writerows(l)
    _load()
    print("\nAdded 10 to stock where needed")

def DELETE():
    _load()
    global k
    l = k.copy()
    k.clear()
    for i in l:
        if i[-2] == "FUN":
            continue
        k.append(i)
    with open("dump/Journal Files/csv/toy.csv", "w", newline="") as f:
        w = wr(f)
        w.writerow(["NAME", "PRICE", "CATEGORY", "STK"])
        w.writerows(k)
    _load()
    print("\nPurged fun :P")


CREATE()
print("""
#-----------------#
|       rEee      |
|  1.  DISPLAY    |
|  2.  SEARCH     |
|  3.  APPEND     |
|  4.  HIGHEST    |
|  5.  MODIFY     |
|  6.  DELETE     |
|  7.  EXIT       |
#-----------------#""")
while (o := int(input(">>> "))) != 7: [DISPLAY, SEARCH, APPEND, HIGHEST, MODIFY, DELETE][o - 1]() if 1 <= o <= 6 else print("Invalid option")
```

Output:



```
number of records: 6
1. ["BUILDING BLOCKS", 45, "EDU", 34]
2. ["LEARNING SCIENCE", 156, "JUNIOR", 5]
3. ["CAR", 134, "FUN", 56]
4. ["ABASCUS", 78, "EDU", 12]
5. ["REMOTE DRONE", 200, "FUN", 7]
6. ["BIKE", 80, "JUNIOR", 28]

#------------------#
|       rEee       |
|  1.  DISPLAY     |
|  2.  SEARCH      |
|  3.  APPEND      |
|  4.  HIGHEST     |
|  5.  MODIFY      |
|  6.  DELETE      |
|  7.  EXIT        |
#------------------#
>>> 1
          NAME          |   PRICE    |  CATEGORY  |     STK
------------------------------------------------------------
BUILDING BLOCKS         |     45     |    EDU     |     34
LEARNING SCIENCE        |    156     |   JUNIOR   |     5
CAR                     |    134     |    FUN     |     56
ABASCUS                 |     78     |    EDU     |     12
REMOTE DRONE            |    200     |    FUN     |     7
BIKE                    |     80     |   JUNIOR   |     28

>>> 2
search term: bike
          NAME          |   PRICE    |  CATEGORY  |     STK
BIKE                    |     80     |   JUNIOR   |     28

>>> 3
number of records: 1
1. ["LEGO", 100000, "FUN", 2]
>>> 4

Moved items that cost more than 100 into highest.csv
>>> 5

Added 10 to stock where needed
>>> 6

Purged fun :P
>>> 1
          NAME          |   PRICE    |  CATEGORY  |     STK
------------------------------------------------------------
BUILDING BLOCKS         |     45     |    EDU     |     34
LEARNING SCIENCE        |    156     |   JUNIOR   |     15
ABASCUS                 |     78     |    EDU     |     12
BIKE                    |     80     |   JUNIOR   |     28

>>> 7
>>>
```

Program 2

Aim: Write a menu driven program to perform operations on a file

Modules used: csv

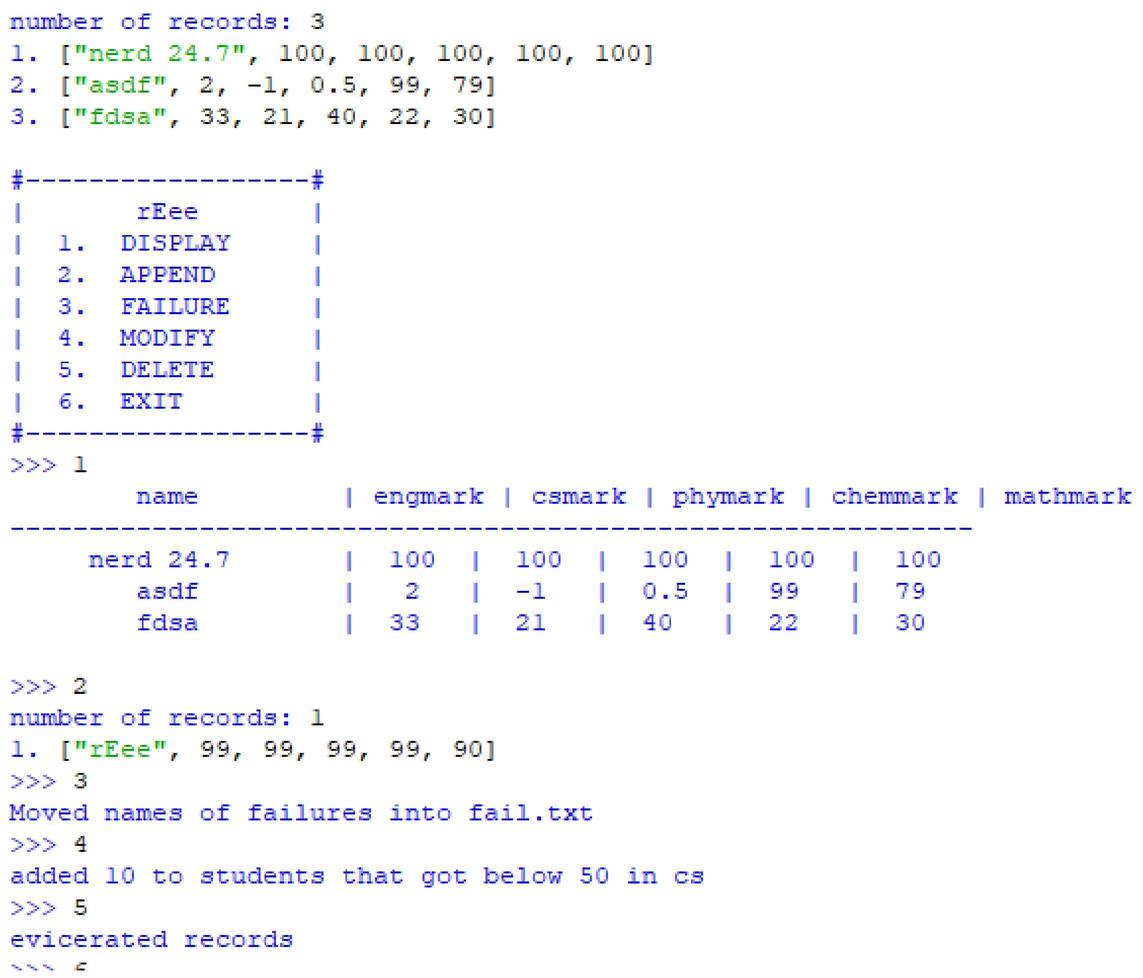
Data types used: Int, Str, List

Script:

```
from csv import writer as wr, reader as re

k = []
def _load():
    global k
    k.clear()
    with open("dump/Journal Files/csv/student.csv", "r", newline="") as f:
        for i , j in enumerate(re(f)):
            if i == 0:
                continue
            k.append(j)

def CREATE():
    with open("dump/Journal Files/csv/student.csv", "w", newline="") as f:
        w = wr(f)
        w.writerow(["name", "engmark", "csmark", "phymark", "chemmark", "mathmark"])
        for i in range(int(input("number of records: "))):
            w.writerow(eval(input(f"{i + 1}. ")))
    _load()

def DISPLAY():
    with open("dump/Journal Files/csv/student.csv", "r", newline="") as f:
        for i , j in enumerate(re(f)):
            if i == 0:
                print(f"{j[0]:^20} | {j[1]:^5} | {j[2]:^5} | {j[3]:^5} | {j[4]:^5} | {j[5]:^5}")
                print("-"*61)
                continue
            print(f"{j[0]:^20} | {j[1]:^5} | {j[2]:^5} | {j[3]:^5} | {j[4]:^5} | {j[5]:^5}")
    print()

def APPEND():
    with open("dump/Journal Files/csv/student.csv", "a", newline="") as f:
        w = wr(f)
        for i in range(int(input("number of records: "))):
            w.writerow(eval(input(f"{i + 1}. ")))
    _load()

def FAILURE():
    _load()
    x = []
    for i in k:
        if float(i[1]) > 26.5 and float(i[-1]) > 26.5 and all(float(x) > 23.5 for x in i[2:-1]):
            x.append(i)
    with open("dump/Journal Files/csv/fail.csv", "w") as f:
        for i in x:
            f.write(f"{i[0]}\n")
    print("Moved names of failures into fail.txt")
```

```python
def MODIFY():
    _load()
    global k
    l = k.copy()
    k.clear()
    for i in l:
        if int(i[2]) < 50:
            i[2] = int(i[2]) + 10
        k.append(l)
    with open("dump/Journal Files/csv/student.csv", "w", newline="") as f:
        w = wr(f)
        w.writerow(["name", "engmark", "csmark", "phymark", "chemmark", "mathmark"])
        w.writerows(k)
    _load()
    print("added 10 to students that got below 50 in cs")

def DELETE():
    _load()
    l = k.copy()
    k.clear()
    for i in l:
        if sum([int(x) for x in i[1:]])/5 < 0.4:
            continue
        k.append(i)
    with open("dump/Journal Files/csv/student.csv", "w", newline="") as f:
        w = wr(f)
        w.writerow(["name", "engmark", "csmark", "phymark", "chemmark", "mathmark"])
        w.writerows(k)
    _load()
    print("evicerated records")


CREATE()
print("""
#-----------------#
|       rEee      |
|  1.  DISPLAY    |
|  2.  APPEND     |
|  3.  FAILURE    |
|  4.  MODIFY     |
|  5.  DELETE     |
|  6.  EXIT       |
#-----------------#""")
while (o := int(input(">>> "))) != 6: [DISPLAY, APPEND, FAILURE, MODIFY, DELETE][o - 1]() if 1 <= o <= 5 else print("Invalid option")
```

Output:



```
number of records: 3
1. ["nerd 24.7", 100, 100, 100, 100, 100]
2. ["asdf", 2, -1, 0.5, 99, 79]
3. ["fdsa", 33, 21, 40, 22, 30]

#------------------#
|       rEee       |
|  1.  DISPLAY     |
|  2.  APPEND      |
|  3.  FAILURE     |
|  4.  MODIFY      |
|  5.  DELETE      |
|  6.  EXIT        |
#------------------#
>>> 1
        name         | engmark | csmark | phymark | chemmark | mathmark
-------------------------------------------------------------
     nerd 24.7       |  100    |  100    |  100    |  100    |  100
        asdf         |   2     |  -1     |  0.5    |  99     |  79
        fdsa         |  33     |  21     |  40     |  22     |  30

>>> 2
number of records: 1
1. ["rEee", 99, 99, 99, 99, 90]
>>> 3
Moved names of failures into fail.txt
>>> 4
added 10 to students that got below 50 in cs
>>> 5
evicerated records
>>> 6
```

Program 3

Aim: Write a function Accept() and wonCount() to accept records and count number of wins

Modules used: csv

Data types used: Int, Str, List

Script:

```
from csv import writer as wr, reader as re

def Accept():
    with open("dump/Journal Files/csv/Result.csv", "a", newline='') as f:
        w = wr(f)
        f.seek(0, 2)
        if f.tell() == 0:
            w.writerow(["St_Id", "St_Name", "Game_Name", "Result"])

        n = int(input("number of records: "))
        for i in range(n):
            w.writerow(eval(input(f"{i + 1}. ")))

def wonCount():
    cnt = 0
    with open("dump/Journal Files/csv/Result.csv", newline='') as f:
        r = re(f)
        next(r, None)
        for row in r:
            if row[3].strip().lower() == 'won':
                cnt += 1
    print(f"No. of wins: {cnt}")

Accept()
print()
wonCount()
```

Output:

```
    number of records: 3
    1. [-3, "asdf", "expedition 34", "Won"]
    2. [2, "nerd 932", "lies of p(eak)", "Loss"]
    3. [449, "fdsa", "Old ring Day Hail", "Tie"]

    No. of wins: 2
>>>
```

STACKS

Program 1

Aim: Write a menu driven program to perform stack operations on a stack of train data

Modules used: N/A

Data types used: Int, Str, List, Tuple

Script:

```python
trains = eval(input(">>> "))
top = None

def updateTop(): global top; top = len(trains) -1

def display():
    updateTop()
    for i in range(top, -1, -1):
        print(trains[i])

def pop():
    global trains
    if len(trains) == 0:
        print("Stack underflow")
        return None
    else:
        k = trains.pop()
        updateTop()
        return k

def push():
    global trains; trains.append(eval(input(">>> ")))
    updateTop()

def peek():
    if len(trains) == 0:
        print("Stack Underflow")
    else:
        updateTop()
        print(trains[top])


print("""
#------------------#
|       rEee       |
|  1.   Display    |
|  2.   Push       |
|  3.   Pop        |
|  4.   Peek       |
|  5.   Exit       |
#------------------#""")

while (o := int(input(">>> "))) != 5: [display, push, pop, peek][o - 1]() if 1 <= o <= 4 else print("Invalid option")
```

Output:

```
>>> [(1, "asdf"), (2, "fdsa")]

#------------------#
|       rEee       |
|  1.   Display    |
|  2.   Push       |
|  3.   Pop        |
|  4.   Peek       |
|  5.   Exit       |
#------------------#
>>> 1
(2, 'fdsa')
(1, 'asdf')
>>> 2
>>> (3, "fast train")
>>> 1
(3, 'fast train')
(2, 'fdsa')
(1, 'asdf')
>>> 4
(3, 'fast train')
>>> 3
>>> 1
(2, 'fdsa')
(1, 'asdf')
>>> 5
>>>
```

Program 2

Aim: Write a program with user defined functions to perform stack operations on a stack of product data

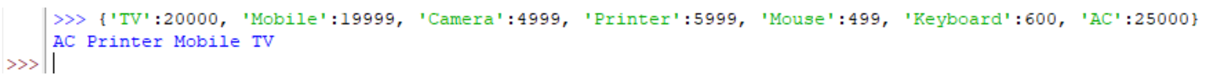
Modules used: N/A

Data types used: Int, Str, List

Script:

```
Product = eval(input(">>> "))
stk = []
top = None

def updateTop() -> None: global top; top = len(stk) - 1

def push() -> None:
    for k, v in Product.items():
        if 5000 <= v and v <= 25000:
            stk.append(k)
    updateTop()

def pop():
    global stk
    if len(stk) == 0:
        print("Stack underflow")
        return None
    else:
        k = stk.pop()
        updateTop()
        return k


push()

while len(stk) != 0:
    print(pop(), end=" ")
```

Output:



```
>>> {'TV':20000, 'Mobile':19999, 'Camera':4999, 'Printer':5999, 'Mouse':499, 'Keyboard':600, 'AC':25000}
AC Printer Mobile TV
>>> |
```

Program 3

Aim: Write a program with user defined functions to perform stack operations on a stack of student data

Modules used: N/A

Data types used: Int, Str, List

Script:

```
Students = eval(input(">>> "))

stk = []
top = None

def updateTop() -> None: global top; top = len(stk) - 1

def push() -> None:
    for k, v in Students.items():
        if v[0] in "Aa":
            stk.append(k)
    updateTop()

def pop():
    global stk
    if len(stk) == 0:
        print("Stack underflow")
        return None
    else:
        k = stk.pop()
        updateTop()
        return k


push()

while len(stk) != 0:
    print(pop(), end=" ")
```

Output:

```
>>> {'S001':'asdf', 'S002':'fdsa', 'S003':'Asdf', 'S004':'nerd 254', 'S005':'rEee', 'S006':'abcd'}
S006 S003 S001
>>> 
```

Program 4

Aim: Write a program with user defined functions to perform stack operations on a stack of integers

Modules used: N/A

Data types used: Int, Str, List

Script:

```python
n = eval(input(">>> "))

stk = []
top = None

def updateTop() -> None: global top; top = len(stk) - 1

def push() -> None:
    for i in n:
        if i%2 == 0:
            stk.append(i)
    updateTop()

def pop():
    global stk
    if len(stk) == 0:
        print("Stack underflow")
        return None
    else:
        k = stk.pop()
        updateTop()
        return k


push()

while len(stk) != 0:
    print(pop(), end=" ")
```

Output:

```
>>> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
8 6 4 2 0
>>> |
```