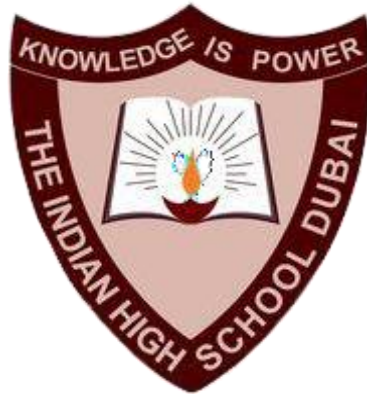


THE INDIAN HIGH SCHOOL - DUBAI



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

S. No.	Title	Page No.
Functions		
1.	Count number of odd and even elements	1
2.	Evaluate the series $1 + x^2 + x^3 + \dots + x^n$	2
3.	Find the factorial of number	3
4.	Get valid email id's	4
5.	Find words longer than 5 letters	5
6.	Menu driven program to <ul style="list-style-type: none"> • Check if number is odd or even • Check if number is prime 	6
7.	Find greater number	7
8.	Move elements divisible by 5 to the end of list	8
9.	Find cars released in 2020 & sort the dict alphabetically	9
10.	Take tuple and return indices of non-zero elements	10
11.	Count number of vowels in user input	11
File Handling 1 – Text Files		
12.	Count lines starting with 'W' or 'H'	12
13.	Count total number of words in a file	13
14.	Move all data from one file to another that doesn't start with '@'	14
15.	Count occurrence of vowels in file	15
16.	Count occurrence of 'The' and 'This' in a file	16
17.	Count occurrence of 'IS', 'TO' & 'UP' in file	17
18.	Display all lines starting with 'P'	18
19.	Count number of lines starting with 'H'	19
20.	Count occurrence of 'a' and 'm' in a file (case insensitive)	20
21.	Count occurrence of 'Catholic' and 'mother in a file	21
22.	Display all lines with 2 characters	22
23.	Count occurrence of 'and' (case insensitive)	23
24.	Menu driven program to do file operations	24
File Handling 2 – Binary Files		
25.	Write a function 'createb()' to create a binary file with details and 'Search()' to look for an employee	25
26.	Insert data at the end of the file	28
27.	Split file in twain based on given condition	29
28.	Menu driven program to do file operations	30

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: $1 + x^2 + x^3 + \dots + x^n$

Modules used: N/A

Data types used: Int

Script:

```
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:

```
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:

```
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:

```
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:

```
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:

```
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

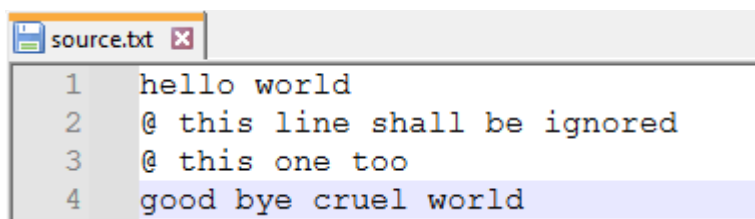
Script:

```
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")
```

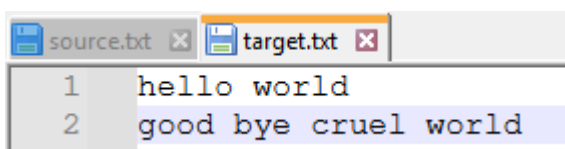
Output:

source.txt:

A screenshot of a text editor window titled 'source.txt'. It contains four lines of text: '1 hello world', '2 @ this line shall be ignored', '3 @ this one too', and '4 good bye cruel world'. The fourth line is highlighted in blue.

```
1 hello world
2 @ this line shall be ignored
3 @ this one too
4 good bye cruel world
```

target.txt

A screenshot of a text editor window titled 'target.txt'. It contains two lines of text: '1 hello world' and '2 good bye cruel world'. The second line is highlighted in blue.

```
1 hello world
2 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:

```
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

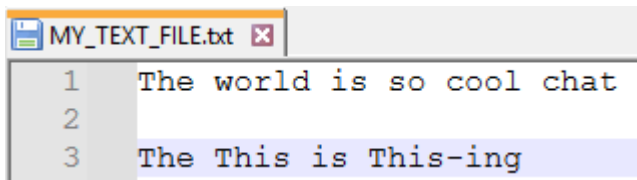
Data types used: Int

Script:

python:

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

MY_TEXT_FILE.txt:



```
1 The world is so cool chat
2
3 The This is This-ing
```

Output:

```
>>> Occurance 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:

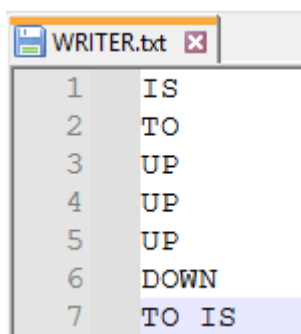
python:

```
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:



1	IS
2	TO
3	UP
4	UP
5	UP
6	DOWN
7	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

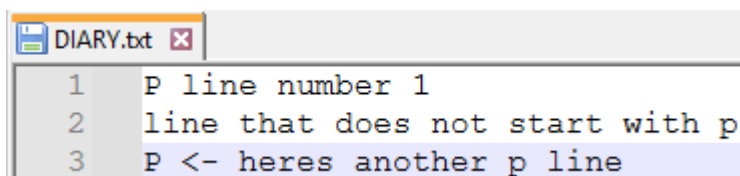
Script:

python:

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

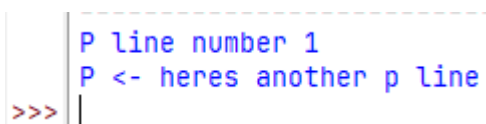
p()
```

DIARY.txt:



```
DIARY.txt
1 P line number 1
2 line that does not start with p
3 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

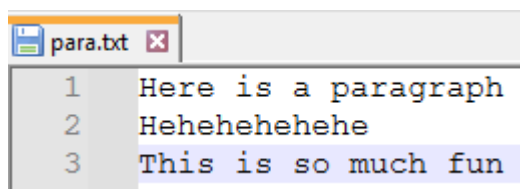
Script:

python:

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

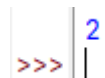
h()

para.txt:



A screenshot of a text editor window titled 'para.txt'. The window contains three lines of text: '1 Here is a paragraph', '2 Hehehehehehe', and '3 This is so much fun'. The third line is highlighted in blue.

Output:



A screenshot of a Python REPL (Read-Eval-Print Loop) showing the output of the function h(). The prompt is '>>>' and the output is '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:

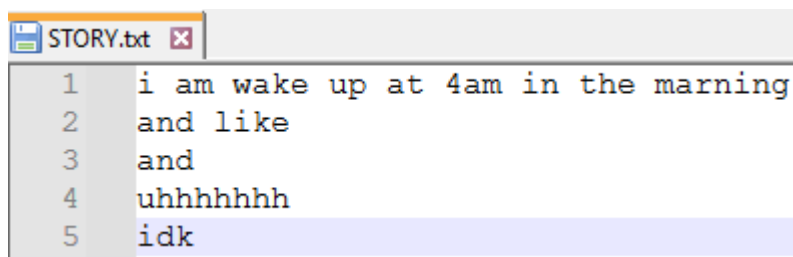
python:

```
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))

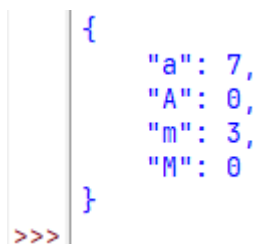
AMCount()
```

STORY.txt:



```
1 i am wake up at 4am in the marning
2 and like
3 and
4 uhhhhhhh
5 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:

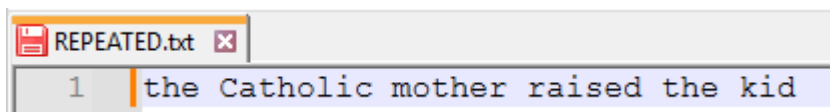
python:

```
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:



```
1 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

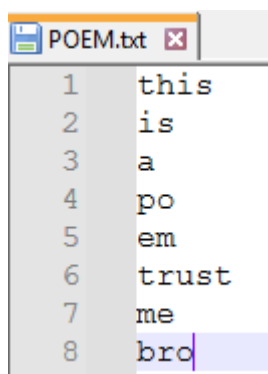
Data types used: Int, List, Str

Script:

python:

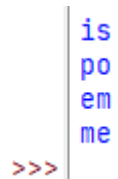
```
def _2chars():  
    with open("dump/Journal Files/POEM.txt") as f:  
        [print(w) for l in f for w in l.split() if len(w) == 2]  
  
_2chars()
```

POEM.txt:



```
POEM.txt x  
1 this  
2 is  
3 a  
4 po  
5 em  
6 trust  
7 me  
8 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

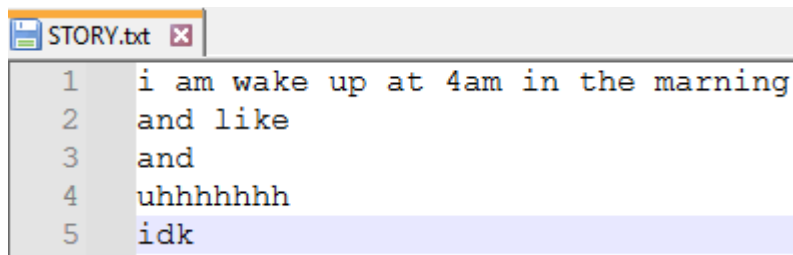
Script:

python:

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

COUNT_AND()

STORY.txt:



A screenshot of a text editor window titled 'STORY.txt'. The window contains five lines of text, numbered 1 to 5 on the left margin. The text is as follows:
1 i am wake up at 4am in the marning
2 and like
3 and
4 uhhhhhhh
5 idk

Output:

```
>>> Occurance 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:

```
_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
            elif 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(f"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(f"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(f"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(f"Original text: text, 'Changed text': ct}")

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, "FDOSA", "employeeeee", -1000000000000000000]
>>> |
```

employee.dat (hex editor, function 1):

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]"(€asdf"€mana
00000020	67	65	72	94	4B	14	65	4B	02	5D	94	28	8C	04	66	64	ger"G€h.....eK.
00000030	73	61	94	8C	0B	63	6C	61	73	73	20	63	6C	6F	77	6E]"(€fdsa"€clas
00000040	94	4D	D0	07	65	4B	03	5D	94	28	8C	06	6E	65	72	64	s clown"MB.eK]"
00000050	33	37	94	8C	0B	6B	6E	6F	77	20	69	74	20	61	6C	6C	(€.nerd37"€know
00000060	94	4A	88	0D	01	00	65	4B	04	5D	94	28	8C	04	41	53	it all"J'....eK.
00000070	44	46	94	68	03	4B	03	65	4B	05	5D	94	28	8C	04	46]"(€ASDF"h.K.eK
00000080	44	53	41	94	8C	0C	65	6D	70	6C	6F	79	65	65	65	65	."(€FDSA"€emp
00000090	65	65	94	8A	07	00	00	3F	90	0D	79	DC	65	75	2E		loyeeeeee"GAAÿ7

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):

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]"(€asdf"€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]"(€fdsa"€clas
00000040	73	20	63	6C	6F	77	6E	94	4D	D0	07	65	4B	03	5D	94	s clown"MB.eK]"
00000050	28	8C	06	6E	65	72	64	33	37	94	8C	0B	6B	6E	6F	77	(€.nerd37"€know
00000060	20	69	74	20	61	6C	6C	94	4A	88	0D	01	00	65	4B	04	it all"J'....eK.
00000070	5D	94	28	8C	04	41	53	44	46	94	68	03	4B	03	65	4B]"(€ASDF"h.K.eK
00000080	05	5D	94	28	8C	04	46	44	53	41	94	8C	0C	65	6D	70	."(€FDSA"€emp
00000090	6C	6F	79	65	65	65	65	65	65	94	47	C3	41	C3	79	37	loyeeeeee"GAAÿ7
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:

```
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):

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
00000010	5D	94	28	8C	04	61	73	64	66	94	8C	07	6D	61	6E	61
00000020	67	65	72	94	47	40	AF	68	00	00	00	00	00	65	4B	02
00000030	5D	94	28	8C	04	66	64	73	61	94	8C	0B	63	6C	61	73
00000040	73	20	63	6C	6F	77	6E	94	4D	D0	07	65	4B	03	5D	94
00000050	28	8C	06	6E	65	72	64	33	37	94	8C	0B	6B	6E	6F	77
00000060	20	69	74	20	61	6C	6C	94	4A	88	0D	01	00	65	4B	04
00000070	5D	94	28	8C	04	41	53	44	46	94	68	03	4B	03	65	4B
00000080	05	5D	94	28	8C	04	46	44	53	41	94	8C	0C	65	6D	70
00000090	6C	6F	79	65	65	65	65	65	65	94	47	C3	41	C3	79	37
000000A0	E0	7C	18	65	4B	06	5D	94	28	8C	07	6E	65	72	64	20
000000B0	37	35	94	8C	08	62	6F	73	73	20	6D	61	6E	94	4A	F8
000000C0	24	01	00	65	75	2E										

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

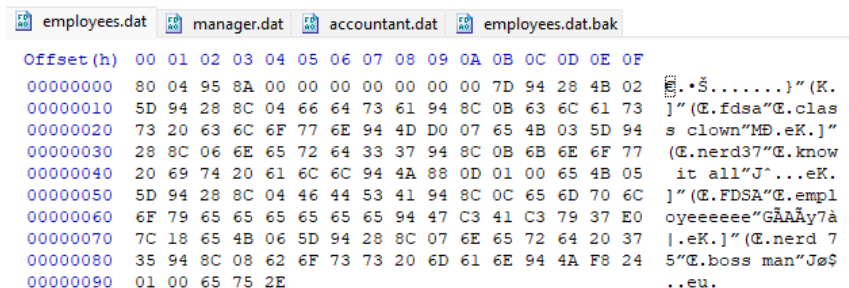
    with open("dump/Journal Files/bin/employees.dat", "wb") as f1:
        pickle.dump(x, f1)

split()
```

Output:

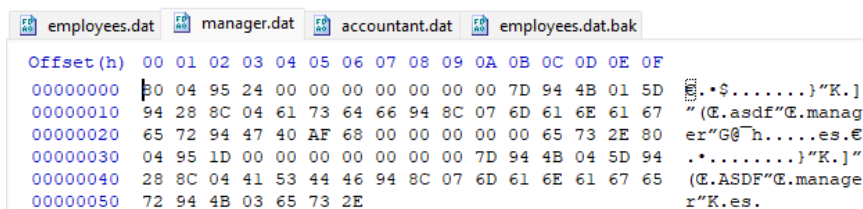
python – no output

employee.dat (hex editor):



Hex editor view of `employee.dat`. The editor shows four tabs: `employees.dat`, `manager.dat`, `accountant.dat`, and `employees.dat.bak`. The `employees.dat` tab is active, displaying a hex dump with offsets from 00 to 0F. The data is a pickled dictionary containing employee records, including names like 'K.', 'fdsa', 'clown', 'nerd37', 'J', 'FDSA', 'nerd 7', 'boss man', and 'eu'.

manager.dat (hex editor):



Hex editor view of `manager.dat`. The editor shows four tabs: `employees.dat`, `manager.dat`, `accountant.dat`, and `employees.dat.bak`. The `manager.dat` tab is active, displaying a hex dump with offsets from 00 to 0F. The data is a pickled dictionary containing manager records, including names like 'K.', 'asdf', 'h', 'K.', 'ASDF', and '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:

```
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)

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")

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
>>> |
```