**Task**

**File operations:** It performed by using built-in functions and methods provided by the language. In this there reading from, writing to, and managing files on your system. Python provides simple interfaces to interact with files and supports text files, binary files, and more.

There are six types of modes in this:

1. Open a file
2. Read a file
3. Write a file (Tuncate)
4. Appending a file
5. Closing a file
6. Checking a file

**Open a file:** The open function is used to open a file, before it going to perform any operations.

**Syntax:** open (file.name, mode)

file= open("example.txt","w")

**Read a file:** It is used to read data from a file, used for the reading files.

**Syntax:** file = open (“example.txt”, ”r”)

file=open("example.txt","r")  
content=file.read()  
print(content)  
file.close()

**Write a file:** we can use the write ( ) the method to write the data.

**Syntax:**

File = open(“example.txt”,”w”)

File . write(“hello world”\n)

File . close ( )

with open("example.txt", "w") as file:  
 file.write("hello world.\n")  
 file.write("Python are easy!")  
  
print("first line.")

Output: first line

**Appending a file:** The 'a' mode opens the file to add content to the end of the file without overwriting existing data.

with open("example.txt", "a") as file:  
 file.write("This is an append mode file \n")  
  
print("Data appended")

Output:

Data appended

**Closing a File:**

While using the with statement, the file is automatically closed. Otherwise, you must manually close the file.

Syntax: file.close ( )

file = open("example.txt", "r")  
content = file.read()  
print(content)  
file.close()

Output:

hello world.

Python are easy!This is an append mode file

**Checking a file:**  It is used to check if a file exists in Python or not we will use the (.os) module or the pathlib module. They both provide easy ways to verify whether a file exists.

import os  
file\_path = "example.txt"  
if os.path.exists(file\_path):  
 print("The file exists.")  
else:  
 print("The file does not exist.")

Output:

The exists

**File Handling:** It is used for reading, writing, deleting, creating of a file is called file handling.

**Advantages of file handling:**

**Versatility:** It is used to allows you to perform a wide range of operations, such as creating, reading, writing, appending, renaming, and deleting files.

**Flexibility:** InFile modes ('r', 'w', 'a', etc.) provide precise control over file operations, including text and binary files. It is highly flexible.

It allows you to work with different file types (e.g. text files, binary files etc.), and to perform different operations on files (e.g. read, write, append, etc.).

**User – Friendly:** In python it provides a user-friendly interface for file handling, making it easy to create, read, and manipulate files.

**Cross-platform**: In python file-handling functions work across different platforms (e.g. Windows, Mac, Linux).

It will allow for seamless integration and compatibility.

**Disadvantages of file handling:**

**Error-prone:**It can be prone to errors, especially if the code is not carefully written or if there are issues with the file system (e.g. file permissions, file locks, etc…).

**Security risks**: It can also pose security risks, especially if the program accepts user input that can be used to access or modify sensitive files on the system.

**Complexity**: It can be complex, especially when working with more advanced file formats or operations. Careful attention must be paid to the code to ensure that files are handled properly and securely.

**Performance**: It can be slower than other programming languages, especially when dealing with large files or performing complex operations.

1. Write a python code to print my name 100 times.

a='sravya'  
count=0  
foriinrange(1,100+1):  
 count=count+i  
 print("str:",a)

Output:

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

str: sravya

1. Write a program to check it is an odd or even number.

num=int(input('enter a number:'))

ifnum%2==0:  
 print("a is an even")

else:  
 print("a is an odd")

Enter a value: 7

Output:

7 is an odd number

1. Write a 7 table by using python code

num=int(input('enteranumber:'))  
i=1  
whilei<11:  
 print(num,'\*',i,'=',num\*i)  
 i += 1

Output:

7 \* 1 = 7

7 \* 2 = 14

7 \* 3 = 21

7 \* 4 = 28

7 \* 5 = 35

7 \* 6 = 42

7 \* 7 = 49

7 \* 8 = 56

7 \* 9 = 63

7 \* 10 = 70

1. Write a python code which number is largest number of 3 number.

a=12  
b=27  
c=9  
ifa>bandb>c:  
 print('aislargest')  
elifb>c:  
 print('bislargest')  
else:  
 print('c is largest')

Output:

B is largest

table1=[  
 {"a":5,"b":10,"c":15},  
 {"a":3,"b":5,"c":7},  
 {"a":15,"b":30,"c":45}  
]  
  
table2=[  
 {"a":7,"b":14,"c":21},  
 {"a":11,"b":22,"c":33},  
 {"a":9,"b":18,"c":27}  
]  
  
  
variables=table1[0].keys()  
min\_values={}  
max\_values={}  
  
variables = table1[0].keys()  
min\_values = {}  
max\_values = {}  
  
for var in variables:  
 values = [row[var] for row in table1 + table2]  
 min\_values[var] = min(values)  
 max\_values[var] = max(values)  
  
# Print the results  
print("Minimum values for each variable:")  
for var in min\_values:  
 print(f"{var}: {min\_values[var]}")  
  
print("\nMaximum values for each variable:")  
for var in max\_values:  
 print(f"{var}: {max\_values[var]}")

Output:

Minimum values for each variable:

a: 3

b: 5

c: 7

Maximum values for each variable:

a: 15

b: 30

c: 45

Task: 2

my\_list=[1,2,3,4,5]  
print(my\_list[3])

my\_list[:2]=[9,13]  
print(my\_list)

my\_list[:5]=[55,56,57,58,59]  
print(my\_list)

my\_list.append(60)  
print(my\_list)

my\_list.insert(3,61)  
print(my\_list)

my\_list.extend([4,71,77])  
print(my\_list)

my\_list=[]  
print(my\_list)

Output:

4

[9, 13, 3, 4, 5]

[55, 56, 57, 58, 59]

[55, 56, 57, 58, 59, 60]

[55, 56, 57, 61, 58, 59, 60]

[55, 56, 57, 61, 58, 59, 60, 4, 71, 77]

[]