Modules in Python:

- We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.
- Modules refer to a file containing Python statements and definitions.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Module creation:

Type the following and save it as `calci.py`

```
def add(a, b):
   return a + b
def sub(a, b):
   return a - b
def mul(a, b):
   return a * b
def div(a, b):
   return a / b
```

Using created module:

```
import calci
a=20
b=10
c=calci.add(a,b)
print(c)
```

or

```
from calci import *
a=20
b=10
c=add(a,b)
print(c)
```

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

Python standard modules:

- We can import a standard module using the `import` statement and access the definitions inside it using the dot operator
- For example, there are many pre-defined functions in standard math module. We need to import math module to support them. Some of them are as follows:

1. ceil() :- This function returns the **smallest integral value greater than the number**. If number is already integer, same number is returned.

2. floor() :- This function returns the **greatest integral value smaller than the number**. If number is already integer, same number is returned.

3. fabs() :- This function returns the **absolute value** of the number

4. exp(a) :- This function returns the value of **e raised to the power a (e**a)** .

5. log(a, b) :- This function returns the logarithmic **value of a with base b**. If base is not mentioned, the computed value is of natural log.

For Example:

```
import math

a = 2.3

# returning the ceil of 2.3
print ("The ceil of 2.3 is : ",
end="")
print (math.ceil(a))

# returning the floor of 2.3
print ("The floor of 2.3 is : ",
end="")
print (math.floor(a))
```

```
import math

# returning the exp of 4
print ("The e**4 value is : ",
end="")
print (math.exp(4))

# returning the log of 2,3
print ("The value of log 2 with
base 3 is : ", end="")
print (math.log(2,3))
```

Import with renaming:

```
# import module by renaming it

import math as m
```

```
print("The value of pi is", m.pi)
```

- We have renamed the `math` module as `m`. This can save us typing time in some cases.
- Note that the name `math` is not recognized in our scope. Hence, `math.pi` is invalid, and `m.pi` is the correct implementation.

Python from...import statement:

- We can import specific names from a module without importing the module as a whole. Here is an example.

```
# import only pi from math module

from math import pi
print("The value of pi is", pi)
```

- Here, we imported only the `pi` attribute from the `math` module.

Import all names:

- We can import all names(definitions) from a module using the following construct:

```
from math import *
print("The value of pi is", pi)
```

- Here, we have imported all the definitions from the math module.

Wikipedia:

We can now import Wikipedia in Python using Wikipedia module. Use the incessant flow of knowledge with Python for daily needs.
Install it as:

```
pip install wikipedia
```

And use it as:

```
import wikipedia
result = wikipedia.page("Bennett University")
print(result.summary)
```

If you wish to get a particular number of sentences from the summary, just pass that as an argument to the summary() function:

```
import wikipedia
print(wikipedia.summary("Debugging", sentences = 2))
```

Emoji:

Emojis have become a way to express and to enhance simple boring texts.
For this, `emoji` module is needed to be installed.
In terminal. Use:

```
pip install emoji
```

To upgrade to the latest packages of emojis. Here's how it can be done:

```
pip install emoji -upgrade
```

```
from emoji import emojize
print(emojize(":thumbs_up:"))
```

File Handling Mode (Read, Write, Create, Append)

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since Random Access Memory (RAM) is volatile (which loses its data when the computer is turned off), we use files for future use of the data by permanently storing them.
- When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order:

1. Open a file
2. Read or write (perform operation)
3. Close the file

Opening a file:

Python has a built-in `open()` function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
f = open("test.txt")     # open file in current directory
f = open("C:/Python38/README.txt")  # specifying full path
```

We can specify the mode while opening a file. In mode, we specify whether we want to read `r`, write `w` or append `a` to the file. We can also specify if we want to open the file in text mode or binary mode. Different kinds of opening modes ae given below:

Modes	Uses	Definition
"r"	Read	Opens a file for reading, error if the file does not exist
"a"	Append	Opens a file for appending, creates the file if it does not exist
"w"	Write	Opens a file for writing, creates the file if it does not exist
"x"	Create	Creates the specified file, returns an error if the file exists
"t"	Open	Creates the specified file, returns an error if the file exists
"rb"	Read binary	Opens in text mode. (default)
"r+"	Reading and Writing	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
"rb+"	Reading and Writing binary	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
"wb"	Writing only in binary format	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
"w+"	Both writing and reading	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist,

		creates a new file for reading and writing.
"wb+"	Both writing and reading in binary	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
"ab"	Appending in binary	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
"a+"	Both appending and reading	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
"ab+"	Both appending and reading in binary	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

Writing to file

- In order to write into a file in Python, we need to open it in write `w`, append `a` or exclusive creation `x` mode.
- We need to be careful with the `w` mode, as it will overwrite into the file if it already exists. Due to this, all the previous data are erased.
- Writing a string or sequence of bytes (for binary files) is done using the `write()` method. This method returns the number of characters written to the file.

```
with open("test.txt",'w',encoding = 'utf-8') as f:
   f.write("my first file\n")
   f.write("This file\n\n")
   f.write("contains three lines\n")
```

This program will create a new file named `test.txt` in the current directory if it does not exist. If it does exist, it is overwritten. Following syntax has been given for different file operations.

File Handling Functions (Open, Read, Write, Remove)

Functions	Syntax	Definition
`open()`	`f = open("demofile.txt")`	The `open()` function takes two parameters; *filename*, and *mode*.
`read()`	`f = open("demofile.txt", "r")` `print(f.read())`	The `open()` function returns a file object, which has a `read()`method for reading the content of the file
`write()`	`f = open("demofile2.txt", "a")` `f.write("Now the file has more content!")`	To write to an existing file, you must add a parameter to the `open()` function
`remove()`	`import os` `os.remove("demofile.txt")`	To delete a file, you must import the OS module, and run its `os.remove()` function

Closing the file:

- After performing all the actions file must be closed.
- It will free up the resources that were tied with the file. It is done using the `close()` method available in Python.
- Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.

```
f = open("test.txt")
# perform file operations
f.close()
```

- This method is not entirely safe. If an exception occurs when we are performing some operation with the file, the code exits without closing the file, hence we can use:

```
try:
   f = open("test.txt")
   # perform file operations
finally:
   f.close()
```

This way, we are guaranteeing that the file is properly closed even if an exception is raised that causes program flow to stop.

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

Various Python File Methods

close()	Closes an opened file. It has no effect if the file is already closed.
detach()	Separates the underlying binary buffer from the `TextIOBase` and returns it.
fileno()	Returns an integer number (file descriptor) of the file.
flush()	Flushes the write buffer of the file stream.
isatty()	Returns `True` if the file stream is interactive.
read(*n*)	Reads at most *n* characters from the file. Reads till end of file if it is negative or `None`.
readable()	Returns `True` if the file stream can be read from.
readline(*n*=-1)	Reads and returns one line from the file. Reads in at most *n* bytes if specified.
readlines(*n*=-1)	Reads and returns a list of lines from the file. Reads in at most *n* bytes/characters if specified.
seek(*offset*,*from*=`SEEK_SET`)	Changes the file position to *offset* bytes, in reference to *from* (start, current, end).
seekable()	Returns `True` if the file stream supports random access.
tell()	Returns the current file location.
truncate(*size*=`None`)	Resizes the file stream to *size* bytes. If *size* is not specified, resizes to current location.
writable()	Returns `True` if the file stream can be written to.
write(*s*)	Writes the string *s* to the file and returns the number of characters written.
writelines(*lines*)	Writes a list of *lines* to the file.

Modules and File Handling

Q1. Predict the output:

```
# assign list
l = ['Green','Red','Yellow']

# open file
with open('gry_list.txt', 'w+') as f:

    # write elements of list
    for items in l:
        f.write('%s\n' %items)

    print("File written successfully")

# close the file
f.close()
f = open("C:/Users/…/gry_list.txt", "r")
print(f.read())
```

Output:

```
File written successfully
Green
Red
Yellow
```

Q2. Predict the output:

```
list_of_tuples = [('Apples', '=', 10), ('Oranges', '<', 20)]
f = open('list_of_tuples_file.txt', 'w')
for t in list_of_tuples:
    line = ' '.join(str(x) for x in t)
    f.write(line + '\n')
f.close()
f = open("C:/Users/…/list_of_tuples_file.txt", "r")
print(f.read())
```

Output:

```
Apples = 10
Oranges < 20
```

Q3. Predict the output:

```
f = open("C:/Users/…/file1.txt", "w")
f.write("Bennett University \n")
f.write("Welcome !!")
f.close()
f = open("C:/Users/…/file1.txt", "r")
print(f.read())
```

Sol.

```
Bennett University
Welcome!!
```

Q4. Take dictionaries as an input, and save it into a file.

```
Input: dictionary = {'Red': 1, 'Green': True, 4: 'Yellow'}
```

Sol.

```
Input: dictionary = {'Red': 1, 'Green': True, 4: 'Yellow'}
dictionary = {'Red': 1, 'Green': True, 4: 'Yellow'}
rgy_file = open('RGY.txt', 'wt')
rgy_file.write(str(dictionary))
rgy_file.close()
```

Q5. If user will provide a specific line number in a given file (i.e., `News.txt`). Your task is read that specific line from a File and print the content.

E.g., `line_number = 4`

Sol.

```
line_number = 4
fo = open("C:/Users/…/News.txt", 'r')
currentline = 1
for line in fo:
    if(currentline == line_number):
        print(line)
        break
    currentline = currentline +1
```

Q6. You have list of fruits:

```
Fruits=['apples','oranges','bananas','mangoes','grapes','strawberry']
```

Your task is takes input as list (i.e., Fruits) and then write the items form list to a file. Also display the items from file.

Output:

```
apples
oranges
bananas
mangoes
grapes
strawberry
```

Sol.

```
Fruits=['apples','oranges','bananas','mangoes','grapes','strawberry']
with open('C:/Users/…/Fruits_file.txt', "w") as myfile:
        for c in Fruits:
                myfile.write("%s\n" % c)

content = open('C:/Users/…/Fruits_file.txt')
print(content.read())
```

Q7. Predict the output of following program using built-in module random and datetime

a)

```
import random
print(random.randint(0, 9))
print(random.random())
print(random.random() * 100)
List = [2.1, 4, False, True, "python", "Bennett"]
print(random.choice(List))
```

Sol. (output may change, due to random)

```
5
0.46295326831357253
3.0918120977435404
Bennett
```

b)

```
import datetime
now = datetime.datetime.now()
print ("date and time : ")
print (now.strftime("%Y-%m-%d %H:%M:%S"))
```

Sol.

```
date and time :
2021-11-20 16:36:45
```

Q8: Predict the output

from math import sqrt, factorial print(sqrt(16)) print(factorial(6))	import math print(math.sqrt(25)) print(math.factorial(3)) print(math.radians(60)) print(math.sin(2)) print(math.cos(0.5)) print(math.tan(0.23))

```
Sol: 4.0
     720
```

```
Sol: 5.0
     6
     1.0471975511965976
     0.9092974268256817
     0.8775825618903728
     0.23414336235146527
```

Q9. Write a Python program to remove newline characters from a file and put them in list.
Input: file:

It is the second-most populous country. Python is a widely used high-level, general-purpose, interpreted, dynamic programming language.

Output: ['It is the second-most populous country.', 'Python is a widely used high-level, general-purpose, interpreted, dynamic programming language.']
Sol.

```
def remove_newlines(fname):
    flist = open(fname).readlines()
    return [s.rstrip('\n') for s in flist]

print(remove_newlines('C:\Users\.....\file3.txt'))
```

Q10. Your course instructor provided two files (i.e., file1.txt and file2.txt), which may contain same text or different. He asked you to check, whether a file is identical and matches the content of anther file. Write a program for this:

Sol.

```
file1 = "file1.txt"
file2 = "file2.txt"

with open(file1) as f:
    f1 = f.read()

with open(file2) as f:
    f2= f.read()

if f1 == f2:
    print("Files are identical")
else:
    print("Files are not identical")
```

Q11. Your course instructor provided a file (i.e., `sample2.txt`), and asked to rename a file to "renamed_by_python.txt". Write a program for this:

Sol.

```
import os

oldname = "sample2.txt"
newname = "renamed_by_python.txt"
with open(oldname) as f:
    content = f.read()

with open(newname, "w") as f:
    f.write(content)

os.remove(oldname)
```

Q12. A file contains a word "Failed" multiple times. Your course instructor assigned you a task, write a program program replaces this word with "Pass" by updating the same file.

Sol.

```
with open("Sample1.txt") as f:
    content = f.read()

content = content.replace("Failed", "Pass")

with open("Sample1.txt", "w") as f:
    f.write(content)
```

Q13. The game () function in a program lets a user play a game and return the score as an integer. You need to read a file 'HiScore.txt' which is either blank or contains the previous Hi-score. You need to write a program to update the score whenever game() breaks the Hi-score.

Sol.

```
def game():
    return 44677

score = game()
with open("hiscore.txt") as f:
    hiScoreStr = f.read()

if hiScoreStr=='':
    with open("hiscore.txt", "w") as f:
        f.write(str(score))

elif int(hiScoreStr)<score :
    with open("hiscore.txt", "w") as f:
        f.write(str(score))
```

Q14. Write a program to read the text from a given file 'poems.txt' and find out whether it contains the word 'Twinkle'

Sol.

```
f = open('poems.txt')
t = f.read()
if 'twinkle' in t:
    print("Twinkle is present")
else:
    print("Twinkle is not present")
f.close()
```

Q15. Predict the output:

New.txt:

Farmer leaders on Saturday said that
they were yet to withdraw the proposed daily
tractor march to Parliament during the Winter session
They said that a call on the future course of
the agitation will be taken during a meeting on Sunday

```
def file_read(fname, nlines):
    from itertools import islice
    with open(fname) as f:
        for line in islice(f, nlines):
            print(line)
file_read("C:/Users/…/News.txt",2)
```

Sol.

```
Farmer leaders on Saturday said that

they were yet to withdraw the proposed daily
```