

## 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)
```

## 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:

<pre>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))</pre>	<pre>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))</pre>
--	--

## 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 emoji  
print(emoji(":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 are 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.
<b>“ab”</b>	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.
<b>“a+”</b>	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.
<b>“ab+”</b>	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
<code>open()</code>	<code>f = open("demofile.txt")</code>	The <code>open()</code> function takes two parameters; <i>filename</i> , and <i>mode</i> .
<code>read()</code>	<code>f = open("demofile.txt", "r")</code> <code>print(f.read())</code>	The <code>open()</code> function returns a file object, which has a <code>read()</code> method for reading the content of the file
<code>write()</code>	<code>f = open("demofile2.txt", "a")</code> <code>f.write("Now the file has more content!")</code>	To write to an existing file, you must add a parameter to the <code>open()</code> function
<code>remove()</code>	<code>import os</code> <code>os.remove("demofile.txt")</code>	To delete a file, you must import the OS module, and run its <code>os.remove()</code> 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.

## Various Python File Methods

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



### Question-1: Predict the output

<pre># A simple module, calc.py  def add(x, y):     return (x+y)  def subtract(x, y):     return (x-y)</pre>	<pre># importing module calc.py import calc  print(add(10, 2)) print(subtract(10, 2))</pre>
--	---

Solution:

12, 8

### Question-2: Predict the output

<pre>from math import sqrt, factorial  print(sqrt(16)) print(factorial(6))</pre>	<pre>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))</pre>
--	---

Sol: 4.0

720

Sol: 5.0

6

1.0471975511965976

0.9092974268256817

0.8775825618903728

0.23414336235146527

### Question-3: Predict the output of following program using built-in module random and datetime

<pre>import random print(random.randint(0, 5)) print(random.random()) print(random.random() * 100) List = [1, 4, True, 800, "python", 27, "hello"] print(random.choice(List))</pre>	<pre>import datetime now = datetime.datetime.now() print ("Current date and time : ") print (now.strftime("%Y-%m-%d %H:%M:%S"))</pre>
---	---

Sol: 2

0.42631005602725625  
63.99485363905726

Sol: Current date and time :

2021-11-19 05:20:22

**Question-4: Write a python program to print the factorial of a number by using module.**

**fact.py**

```
def fl(x):  
    s=1  
    for I in range(1, x+1)  
        s=s*i  
    return(s)
```

**main.py**

```
import fact  
a=int(input())  
print("Factorial of number", a, "is", fact.fl(a))
```

**Question-5: Reverse a number by using python module.**

**rev.py**

```
def r(x):  
    res=0  
    while(x>0):  
        y=x%10  
        res=(res*10)+y  
        x=x//10  
    return(res)
```

**main.py**

```
import rev  
a=int(input())  
print("Reverse of", a, "is", rev.r(a))
```

**Question-6: Display the natural, and even number by using python module.**

**numcom.py**

```
def nutu(x):  
    print("Natural numbers")  
    for i in range(1,x+1):  
        print(i, end="")  
    print("")  
def odd(x):  
    print("Odd numbers")  
    for i in range(1,x+1,2):
```

## Modules and File Handling



```
        print(i, end='')
    print("")
def even(x):
    print("Even numbers")
    for i in range(2,x+1,2):
        print(i, end='')
    print("")
main.py
import numcom
a=int(input())
numcom.nutu(a)
numcom.even(a)
```

### Question-7: Predict the output:

file1.txt =

Hello! Welcome to Bennett university.  
This file is for testing purposes.  
Good Luck!

```
f = open("C:/Users/home/Desktop/Formatted/file1.txt", "r")
print(f.read())
```

### Output:

Hello! Welcome to Bennett university.  
This file is for testing purposes.  
Good Luck!

### Question-8: Predict the output:

```
f = open("C:/Users/home/Desktop/Formatted/file1.txt", "r")
print(f.read(7))
print(f.read(10))
```

### Output:

Hello!  
Welcome to

### Question-9: Predict the output:

```
f = open("C:/Users/home/Desktop/Formatted/file1.txt", "r")
```

```
print(f.readline())
```

**Output:**

Hello! Welcome to Bennett university.

**Question-10: Predict the output:**

```
f = open("C:/Users/home/Desktop/Formatted/file1.txt", "r")
for x in f:
    print(x)
```

**Output:**

Hello! Welcome to Bennett university.

This file is for testing purposes.

Good Luck!

**Question-11: Create a textfile named file.txt, it is having alphanumeric characters. Read this file and write to another file named file2.txt, this new file must not have numbers of digits.**

**Input:**

file.txt

Hi class, you have been added to eb10

**Output:**

file2.txt

Hi class, you have been added to eb

**Solution:**

```
fh=open("file.txt",'r')
fw=open("file2.txt",'w')
rec=fh.read();
for a in rec:
    if (a.isdigit() != True):
        print(a,end=' ')
        fw.write(a)
##    print("hi")
fh.close()
```

```
fw.close()  
##print("hi")
```

**Question-12: Create a textfile named python.txt, it is having both uppercase and lowercase letters, Read the file and count the uppercase and lowercase letters.**

**Input:**

**python.txt**

I am Rahul

Attending python tutorial

I am from India

**Output:**

Total no. of upper-case alphabets : 5

Total no. of lower-case alphabets : 38

**Solution:**

```
def lettercount():  
    upper=0  
    lower=0  
    f1=open("python.txt",'r')  
    line=f1.read()  
    for i in line:  
        if (i.isupper() == True):  
            upper+=1  
        if (i.islower() == True):  
            lower+=1  
    print("Total no. of upper-case alphabets :",upper)  
    print("Total no. of lower-case alphabets :",lower)  
lettercount()
```

**Question-13: Create a textfile named python.txt, it is having many lines, Read the lines and display all the lines along with the line number.**

**Input:**

Python.txt

I am Rahul

Attending python tutorial

I am from India

**Output:**

1 I am Rahul

2 Attending python tutorial

## Modules and File Handling



3 I am from India

**Solution:**

```
fh=open("python.txt","r")
count=0
lines=fh.readlines()
for a in lines:
    count=count+1
    print(count,a)
fh.close()
```