

# Python Programming

## 1. Write a program to demonstrate different number data types in Python.

```
'''Aim:Write a program to demonstrate different number data types in Python.'''

a=10; #Integer Datatype
b=11.5; #Float Datatype
c=2.05j; #Complex Number
print("a is Type of",type(a)); #prints type of variable a
print("b is Type of",type(b)); #prints type of variable b
print("c is Type of",type(c)); #prints type of variable c
```

### Output:

```
E:\Python>python week1.py
a is Type of <class 'int'>
b is Type of <class 'float'>
c is Type of <class 'complex'>
```

## 2. Write a program to perform different Arithmetic Operations on numbers in Python.

```
'''Aim:Write a program to perform different Arithmetic Operations on numbers in Python.'''

a=int(input("Enter a value")); #input() takes data from console at runtime as string.
b=int(input("Enter b value")); #typecast the input string to int.
print("Addition of a and b ",a+b);
print("Subtraction of a and b ",a-b);
print("Multiplication of a and b ",a*b);
print("Division of a and b ",a/b);
print("Remainder of a and b ",a%b);
print("Exponent of a and b ",a**b); #exponent operator (a^b)
print("Floar division of a and b ",a//b); # floar division
```

### Output:

```
E:\Python>python week2.py
Enter a value3
Enter b value2
Addition of a and b 5
Subtraction of a and b 1
Multiplication of a and b 6
Division of a and b 1.5
Remainder of a and b 1
Exponent of a and b 9
Floar division of a and b 1
```

### 3. Write a program to create, concatenate and print a string and accessing sub-string from a given string.

```
''' Aim:Write a program to create, concatenate and print a string and accessing sub-string
from a given string.'''

s1=input("Enter first String : ");
s2=input("Enter second String : ");
print("First string is : ",s1);
print("Second string is : ",s2);
print("concatenations of two strings :",s1+s2);
print("Substring of given string :",s1[1:4]);
```

#### Output:

```
E:\Python>python week3.py
Enter first String : COMPUTER-
Enter second String : SCIENCE
First string is :  COMPUTER-
Second string is :  SCIENCE
concatenations of two strings : COMPUTER-SCIENCE
Substring of given string : OMP
```

### 4. Write a python script to print the current date in the following format "Sun May 29 02:26:23 IST 2017"

```
'''Aim: . Write a python script to print the current date in the following format Sun May 29
02:26:23 IST 2017 '''
import time;
ltime=time.localtime();
print(time.strftime("%a %b %d %H:%M:%S %Z %Y",ltime)); #returns the formatted time

'''
%a : Abbreviated weekday name.
%b : Abbreviated month name.
%d : Day of the month as a decimal number [01,31].
%H : Hour (24-hour clock) as a decimal number [00,23].
%M : Minute as a decimal number [00,59].
%S : Second as a decimal number [00,61].
%Z : Time zone name (no characters if no time zone exists).
%Y : Year with century as a decimal number.'''
```

#### Output:

```
E:\Python>python week4.py
Sat Mar 06 14:52:49 India Standard Time 2021
```

### 5. Write a program to create, append, and remove lists in python.

## How to create a list?

In Python programming, a list is created by placing all the items (elements) inside square brackets `[]`, separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

```
# empty list
my_list = []

# list of integers
my_list = [1, 2, 3]

# list with mixed data types
my_list = [1, "Hello", 3.4]
```

A list can also have another list as an item. This is called a nested list.

```
# nested list
my_list = ["mouse", [8, 4, 6], ['a']]
```

We can add one item to a list using the `append()` method or add several items using `extend()` method.

```
# Appending and Extending lists in Python
odd = [1, 3, 5]

odd.append(7)

print(odd)

odd.extend([9, 11, 13])

print(odd)
```

### Output

```
[1, 3, 5, 7]
[1, 3, 5, 7, 9, 11, 13]
```

We can also use `+` operator to combine two lists. This is also called concatenation.

## Delete/Remove List Elements

We can delete one or more items from a list using the keyword `del`. It can even delete the list entirely.

```
# Deleting list items
my_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']

# delete one item
del my_list[2]

print(my_list)

# delete multiple items
del my_list[1:5]

print(my_list)

# delete entire list
del my_list

# Error: List not defined
print(my_list)
```

### Output


```
['p', 'r', 'b', 'l', 'e', 'm']
['p', 'm']
Traceback (most recent call last):
  File "<string>", line 18, in <module>
NameError: name 'my_list' is not defined
```

We can use `remove()` method to remove the given item or `pop()` method to remove an item at the given index.

The `pop()` method removes and returns the last item if the index is not provided. This helps us implement lists as stacks (first in, last out data structure).

**6. Write a program to demonstrate working with tuples in python.**

A tuple can have any number of items and they may be of different types (integer, float, list, [string](#), etc.).



```
# Different types of tuples

# Empty tuple
my_tuple = ()
print(my_tuple)

# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

## Output

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
```

A tuple can also be created without using parentheses. This is known as tuple packing.

```
my_tuple = 3, 4.6, "dog"
print(my_tuple)

# tuple unpacking is also possible
a, b, c = my_tuple

print(a)      # 3
print(b)      # 4.6
print(c)      # dog
```

### Output

```
(3, 4.6, 'dog')
3
4.6
dog
```

Having one element within parentheses is not enough. We will need a trailing comma to indicate that it is, in fact, a tuple.

```
my_tuple = ("hello")
print(type(my_tuple)) # <class 'str'>

# Creating a tuple having one element
my_tuple = ("hello",)
print(type(my_tuple)) # <class 'tuple'>

# Parentheses is optional
my_tuple = "hello",
print(type(my_tuple)) # <class 'tuple'>
```

### Output

```
<class 'str'>
<class 'tuple'>
<class 'tuple'>
```

7. Write a program to demonstrate working with dictionaries in python.

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

## Accessing Elements from Dictionary

While indexing is used with other data types to access values, a dictionary uses `keys`. Keys can be used either inside square brackets `[]` or with the `get()` method.

If we use the square brackets `[]`, `KeyError` is raised in case a key is not found in the dictionary. On the other hand, the `get()` method returns `None` if the key is not found.

```
# get vs [] for retrieving elements
my_dict = {'name': 'Jack', 'age': 26}

# Output: Jack
print(my_dict['name'])

# Output: 26
print(my_dict.get('age'))

# Trying to access keys which doesn't exist throws error
# Output None
print(my_dict.get('address'))

# KeyError
print(my_dict['address'])
```

## Output

```
Jack
26
None
Traceback (most recent call last):
  File "<string>", line 15, in <module>
    print(my_dict['address'])
KeyError: 'address'
```

## 8. Write a python program to find largest of three numbers.

```
# Python program to find the largest number among the three input numbers

# change the values of num1, num2 and num3
# for a different result
num1 = 10
num2 = 14
num3 = 12

# uncomment following lines to take three numbers from user
#num1 = float(input("Enter first number: "))
#num2 = float(input("Enter second number: "))
#num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3

print("The largest number is", largest)
```

## Output

```
The largest number is 14.0
```

## 9. Write a Python program to convert temperatures to and from Celsius, Fahrenheit. [ Formula: $c = (f-32)(5/9)$ ]



```

temp = float(input("Enter Temperature: "))
unit = input("Enter unit('C' for Celsius or 'F' for Fahrenheit): ")

if unit == 'C' or unit == 'c' :
    newTemp = 9 / 5 * temp + 32
    print("Temperature in Fahrenheit =", newTemp)
elif unit == 'F' or unit == 'f' :
    newTemp = 5 / 9 * (temp - 32)
    print("Temperature in Celsius =", newTemp)
else :
    print("Unknown unit", unit)

```

## OUTPUT

```

Enter Temperature: 38
Enter unit('C' for Celsius or 'F' for Fahrenheit): C
Temperature in Fahrenheit = 100.4

```

**10. Write a Python program to construct the following pattern, using a nested for loop**

```

*

* *

* * *

* * * *

* * * * *

* * * *

* * *

* *

*

```

**Python Code:**

```
1  n=5;
2  for i in range(n):
3      for j in range(i):
4          print ('* ', end="")
5      print('')
6
7  for i in range(n,0,-1):
8      for j in range(i):
9          print('* ', end="")
10     print('')
11
```

**Sample Output:**

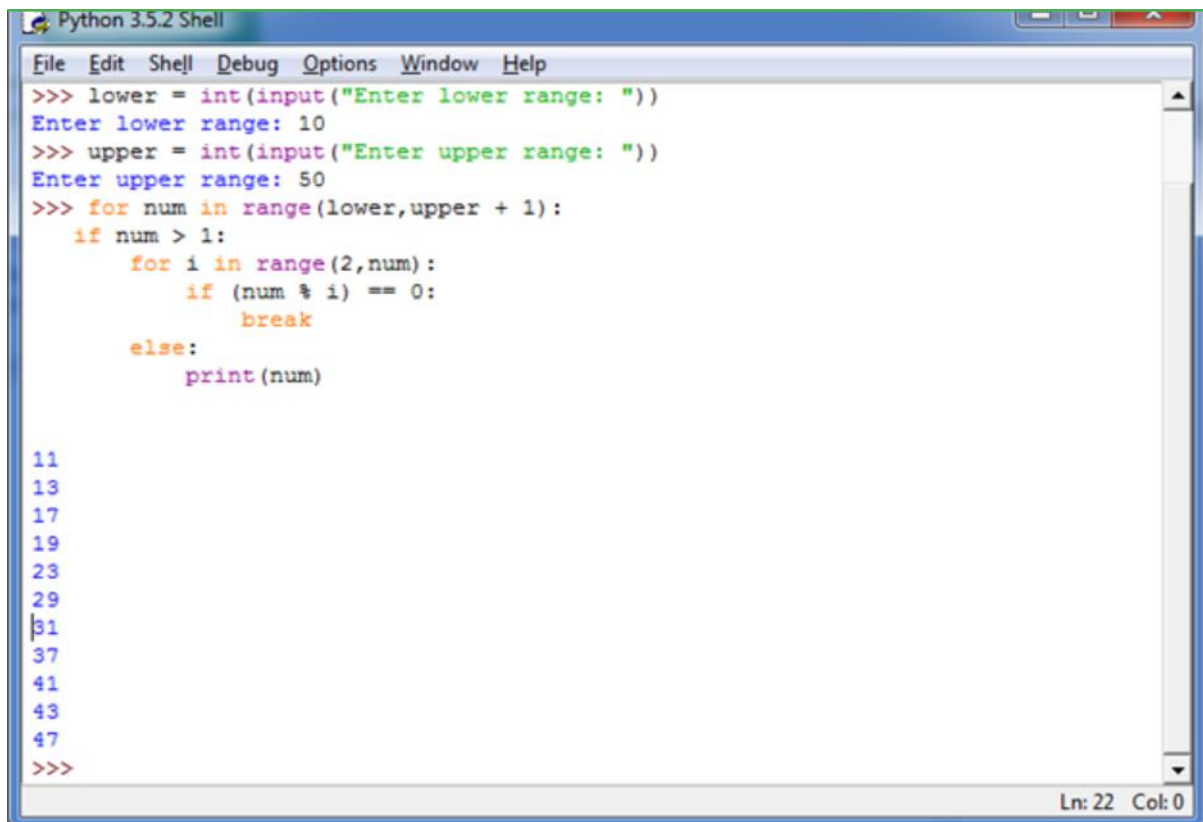
```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

**11. Write a Python script that prints prime numbers less than 20.**

```
#Take the input from the user:
lower = int(input("Enter lower range: "))
upper = int(input("Enter upper range: "))

for num in range(lower,upper + 1):
    if num > 1:
        for i in range(2,num):
            if (num % i) == 0:
                break
        else:
            print(num)
```

This example will show the prime numbers between 10 and 50.



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> lower = int(input("Enter lower range: "))
Enter lower range: 10
>>> upper = int(input("Enter upper range: "))
Enter upper range: 50
>>> for num in range(lower,upper + 1):
    if num > 1:
        for i in range(2,num):
            if (num % i) == 0:
                break
        else:
            print(num)

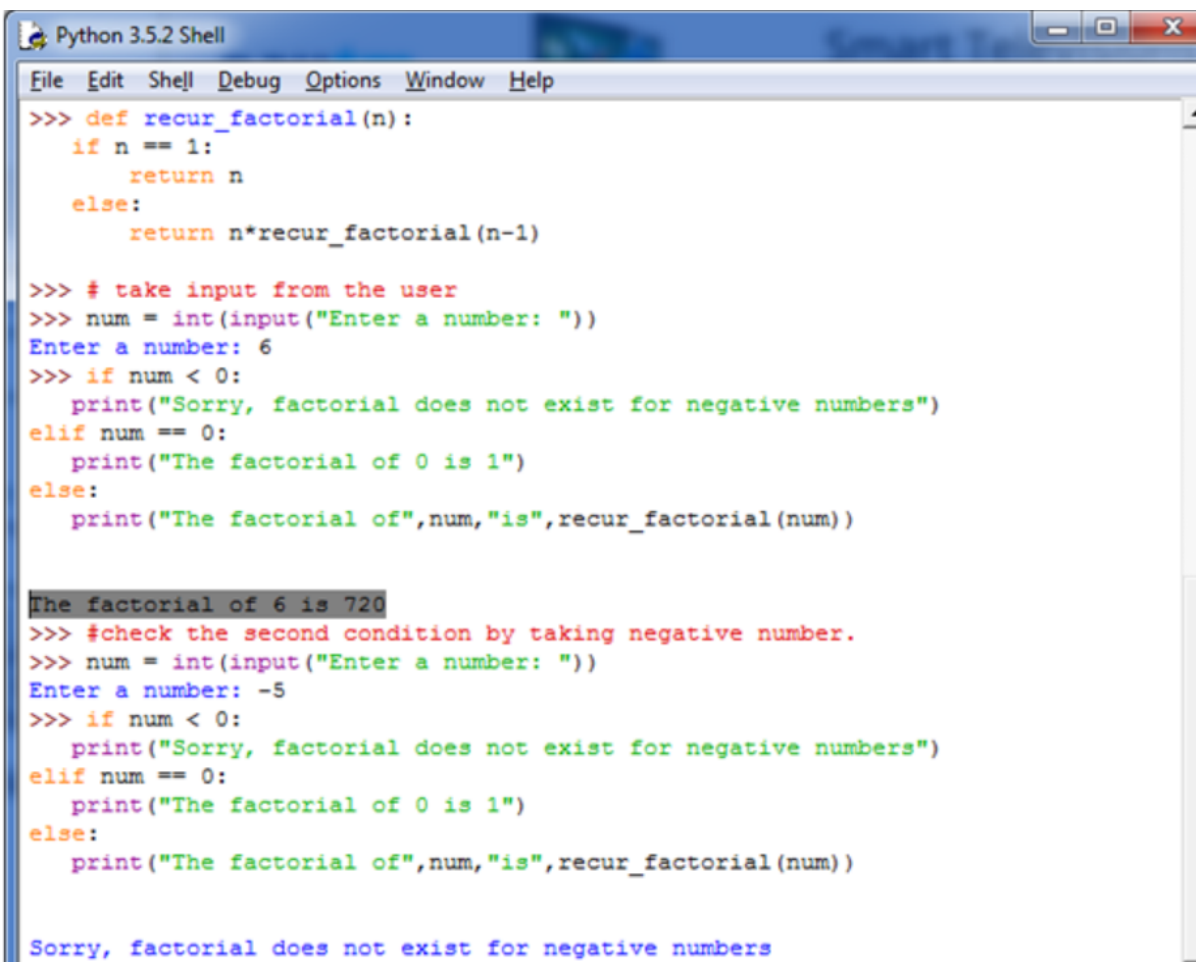
11
13
17
19
23
29
31
37
41
43
47
>>>
```

12. Write a python program to find factorial of a number using Recursion.

```

def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)
# take input from the user
num = int(input("Enter a number: "))
# check is the number is negative
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    print("The factorial of",num,"is",recur_factorial(num))

```



```

Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> def recur_factorial(n):
>>>     if n == 1:
>>>         return n
>>>     else:
>>>         return n*recur_factorial(n-1)
>>> # take input from the user
>>> num = int(input("Enter a number: "))
Enter a number: 6
>>> if num < 0:
>>>     print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
>>>     print("The factorial of 0 is 1")
else:
>>>     print("The factorial of",num,"is",recur_factorial(num))

The factorial of 6 is 720
>>> #check the second condition by taking negative number.
>>> num = int(input("Enter a number: "))
Enter a number: -5
>>> if num < 0:
>>>     print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
>>>     print("The factorial of 0 is 1")
else:
>>>     print("The factorial of",num,"is",recur_factorial(num))

Sorry, factorial does not exist for negative numbers

```

**13. Write a program that accepts the lengths of three sides of a triangle as inputs. The program output should indicate whether or not the triangle is a right triangle (Recall from the Pythagorean Theorem that in a right triangle, the square of one side equals the sum of the squares of the other two sides).**

```
'''Write a program that accepts the lengths of three sides of a triangle as inputs. The
program output should indicate whether or not the triangle is a right triangle (Recall
from the Pythagorean Theorem that in a right triangle, the square of one side equals
the sum of the squares of the other two sides).'''
base=float(input("Enter length of Base : "))
perp=float(input("Enter length of Perpendicular : "))
hypo=float(input("Enter length of Hypotenuse : "))

if hypo**2==((base**2)+(perp**2)):
    print("It's a right triangle")
else:
    print("It's not a right triangle")
```

### Output:

```
E:\Python>python week13.py
Enter length of Base : 3
Enter length of Perpendicular : 4
Enter length of Hypotenuse : 5
It's a right triangle

E:\Python>python week13.py
Enter length of Base : 2
Enter length of Perpendicular : 3
Enter length of Hypotenuse : 4
It's not a right triangle
```

**14. Write a python program to define a module to find Fibonacci Numbers and import the module to another program.**

```
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

Then open an other file `__main__.py` in the same folder as `fibonacci.py`. The content of `__main__.py` should be:

```
import fibonacci
print(fibonacci.fib(3))
```

## 15. Write a python program to define a module and import a specific function in that module to another program

### Writing Modules

A module is simply a Python file with the `.py` extension. The name of the file becomes the module name. Inside the file, we can have definitions and implementations of classes, variables, or functions. These can then be used in other Python programs.

Let us begin by creating a function that simply prints "Hello World". To do this, create a new Python file and save it as `hello.py`. Add the following code to the file:

```
def my_function():
    print("Hello World")
```

If you run the above code, it will return nothing. This is because we have not told the program to do anything. It is true that we have created a function named `my_function()` within the code, but we have not called or invoked the function. When invoked, this function should print the text "Hello World".

Now, move to the same directory where you have saved the above file and create a new file named `main.py`. Add the following code to the file:

```
import hello

hello.my_function()
```

### Output

```
Hello World
```

The function was invoked successfully. We began by importing the module. The name of the file was `hello.py`, hence the name of the imported module is `hello`.

Also, note the syntax that we have used to invoke the function. This is called the "dot notation", which allows us to call the function by first specifying the module name, and then the name of the function.

However, that is just one way of importing the module and invoking the function. We could have done it as follows:

```
from hello import my_function

my_function()
```

### Output

```
Hello World
```

In the above example, the first line commands the Python interpreter to import a function named `my_function` from a module named `hello`. In such a case, you don't have to use the dot notation to access the function, you can just call it directly.

However, in the case where our `hello` module has multiple functions, the statement `from hello import my_function` will not import all `hello`'s functions into our program, only `my_function`. If you attempt to access any other function, an error will be generated. You have to import the whole module or import each individual functions in order to use them.

**16. Write a script named copyfile.py. This script should prompt the user for the names of two text files. The contents of the first file should be input and written to the second file.**

Source Code :

```
# to prompt the user to enter the file1 which is input file
infile=input("enter the input filename with extension ");

# to prompt the user to enter the file2 which is output file
outfile=input("enter the output filename with extension ");

#opening the file1 in reading mode
f1=open(infile,"r");

#opening the file2 in output mode
f2=open(outfile,"w+");

#reading the content of file1 to content variable
content=f1.read();

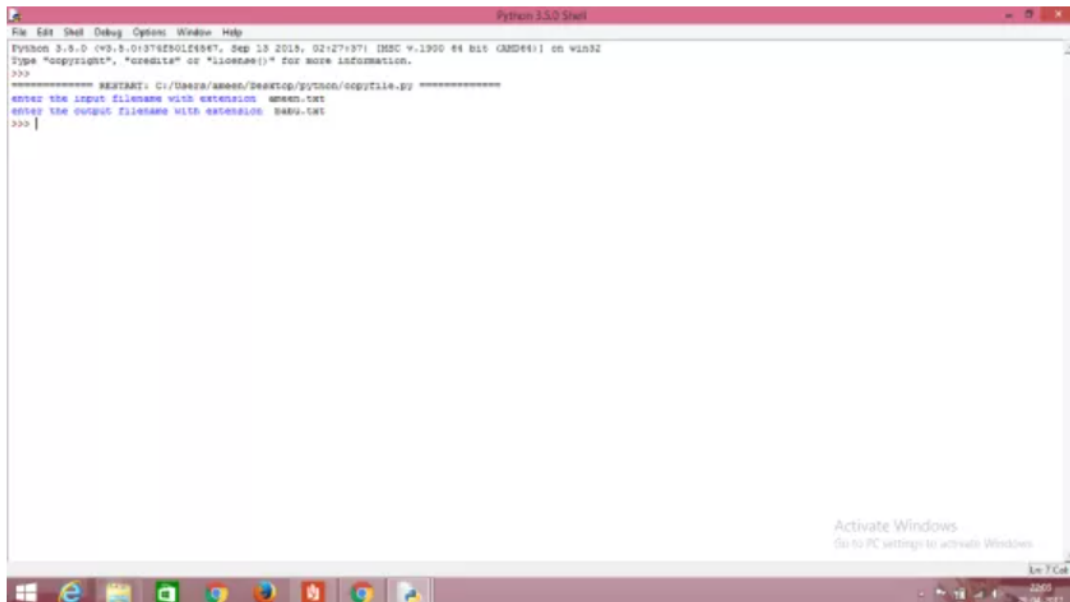
#writing to the value of content variable to file2
f2.write(content);

#closing the file1 and file2
f1.close();
f2.close();
```



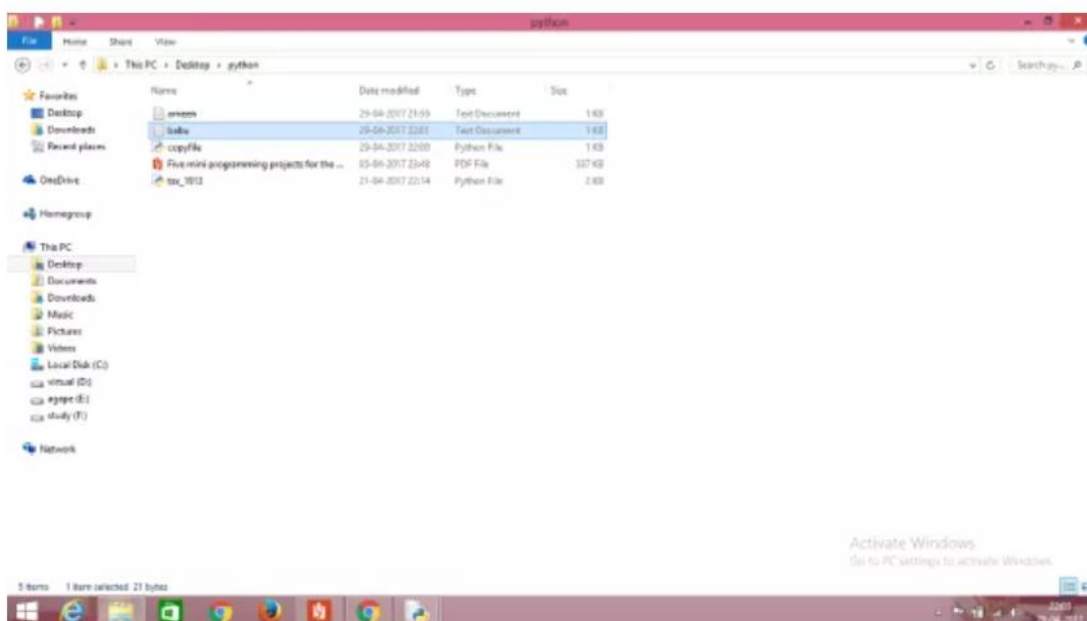
Output :

Go to the path of the output file and open it with corresponding editor .Then you get the output .

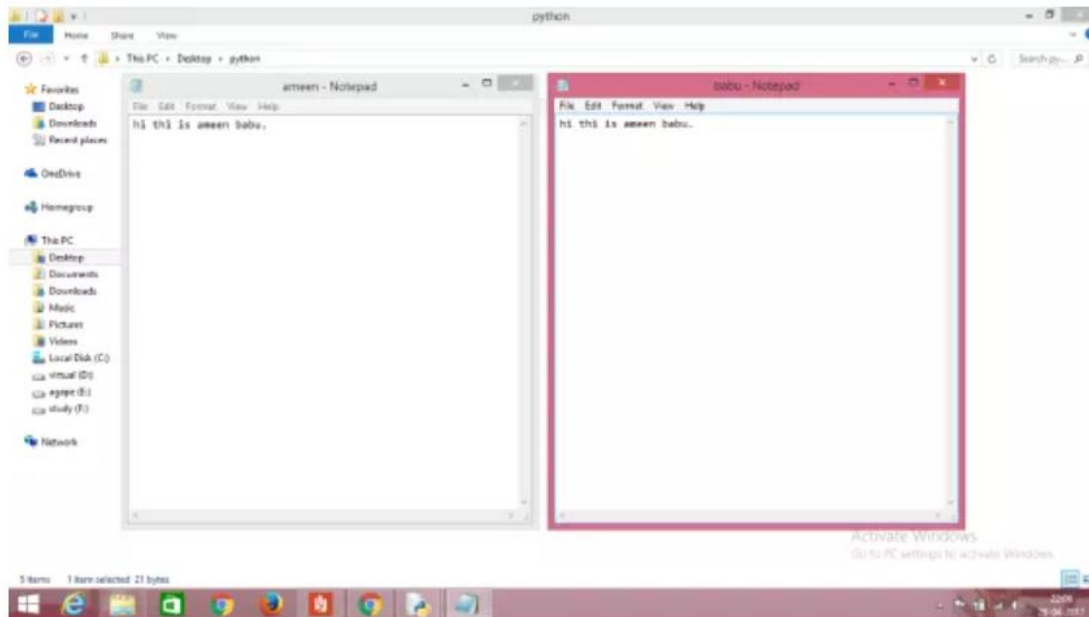


```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:13762501e5e7, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\amano\Desktop\python\copyfile.py =====
enter the input filename with extension input.txt
enter the output filename with extension output.txt
>>>
```

File Path :



Output file Content :



**17. Write a program that inputs a text file. The program should print all of the unique words in the file in alphabetical order.**

```
def unique_file(input_filename, output_filename):
    input_file = open(input_filename, 'r')
    file_contents = input_file.read()
    input_file.close()
    duplicates = []
    word_list = file_contents.split()
    file = open(output_filename, 'w')
    for word in word_list:
        if word not in duplicates:
            duplicates.append(word)
            file.write(str(word) + "\n")
    file.close()

unique_file('sample.txt', 'unique_output.txt')
for line in sorted(open('unique_output.txt')):
    print(line, end="")
```

**Output:**

This  
is  
the  
most  
import  
task.  
you  
have  
performed  
in  
a  
while.  
python  
rocks  
yeah.

### 18. Write a Python class to convert an integer to a roman numeral.

```
1 class py_solution:
2     def int_to_Roman(self, num):
3         val = [
4             1000, 900, 500, 400,
5             100, 90, 50, 40,
6             10, 9, 5, 4,
7             1
8         ]
9         syb = [
10            "M", "CM", "D", "CD",
11            "C", "XC", "L", "XL",
12            "X", "IX", "V", "IV",
13            "I"
14        ]
```

```

15         roman_num = ''
16         i = 0
17         while num > 0:
18             for _ in range(num // val[i]):
19                 roman_num += syb[i]
20                 num -= val[i]
21             i += 1
22         return roman_num
23
24
25 print(py_solution().int_to_Roman(1))
26 print(py_solution().int_to_Roman(4000))

```

Output:

```

I
MMMM

```

## 19. Write a Python class to implement pow (x, n)

```

# Python3 program to calculate pow(x,n)

# Function to calculate x
# raised to the power y
def power(x, y):

    if (y == 0): return 1
    elif (int(y % 2) == 0):
        return (power(x, int(y / 2)) *
                power(x, int(y / 2)))
    else:
        return (x * power(x, int(y / 2)) *
                power(x, int(y / 2)))

# Driver Code
x = 2; y = 3
print(power(x, y))

```

Output:

```

8

```

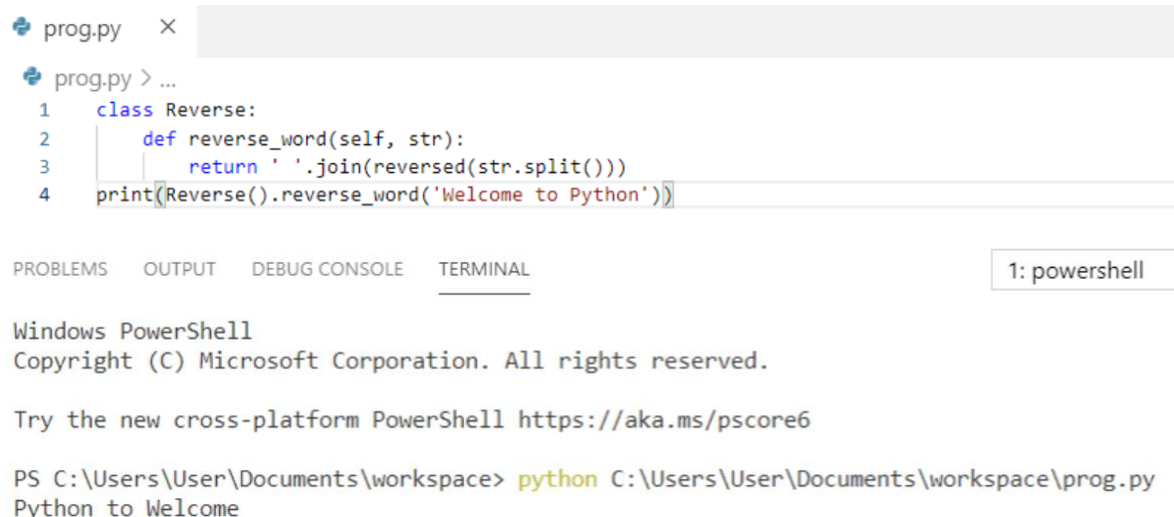
## 20. Write a Python class to reverse a string word by word.

- Firstly, I have created a class called **Reverse**
- After that, we have defined the function inside the class
- Now, we will split the sentence into a list of words, and to form the new sentence join() is used for joining the words in the list.
- At last, print **Reverse().reverse\_word('Welcome to Python')** to get the output.

```
class Reverse:
    def reverse_word(self, str):
        return ' '.join(reversed(str.split()))
print(Reverse().reverse_word('Welcome to Python'))
```

To get the output, I have used **print(Reverse().reverse\_word('Welcome to Python'))**.

You can refer to the below screenshot for the output.



The screenshot shows a code editor with a file named 'prog.py'. The code defines a class 'Reverse' with a method 'reverse\_word' that reverses the words of a string. The main execution line is 'print(Reverse().reverse\_word('Welcome to Python'))'. Below the editor, the 'TERMINAL' tab is active, showing the Windows PowerShell prompt. The command 'python C:\Users\User\Documents\workspace\prog.py' has been executed, resulting in the output 'Python to Welcome'.

```
prog.py > ...
1 class Reverse:
2     def reverse_word(self, str):
3         return ' '.join(reversed(str.split()))
4     print(Reverse().reverse_word('Welcome to Python'))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\User\Documents\workspace> python C:\Users\User\Documents\workspace\prog.py  
Python to Welcome

## 21. Python Program for Find remainder of array multiplication divided by n

```

# Python3 program to
# find remainder when
# all array elements
# are multiplied.

# Find remainder of arr[0] * arr[1]
# * .. * arr[n-1]
def findremainder(arr, lens, n):
    mul = 1

    # find the individual
    # remainder and
    # multiple with mul.
    for i in arr:
        mul = mul * (i % n)

    return mul % n

# Driven code
arr = [100, 10, 5, 25, 35, 14]
lens = len(arr)
n = 11

# print the remainder
# of after multiple
# all the numbers
print(findremainder(arr, lens, n))

```

**Output:**

9

## 22. Python Program for cube sum of first n natural numbers

Print the sum of series  $1^3 + 2^3 + 3^3 + 4^3 + \dots + n^3$  till n-th term.

```
# Simple Python program to find sum of series  
# with cubes of first n natural numbers
```

```
# Returns the sum of series
```

```
def sumOfSeries(n):  
    sum = 0  
    for i in range(1, n+1):  
        sum += i*i*i  
  
    return sum
```

```
# Driver Function
```

```
n = 5  
print(sumOfSeries(n))
```

**Output:**

225

## 23. Python program to check whether a number is Prime or not



```

# Python program to check if
# given number is prime or not

num = 11

# If given number is greater than 1
if num > 1:

    # Iterate from 2 to n / 2
    for i in range(2, int(num/2)+1):

        # If num is divisible by any number between
        # 2 and n / 2, it is not prime
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
        else:
            print(num, "is a prime number")

else:
    print(num, "is not a prime number")

```

#### Output

```
11 is a prime number
```

## 24. Python Program for Program to find area of a circle

Area of a circle can simply be evaluated using following formula.

```

Area = pi * r2
where r is radius of circle

```

# Python program to find Area of a circle

```
def findArea(r):  
    PI = 3.142  
    return PI * (r*r);
```

# Driver method

```
print("Area is %.6f" % findArea(5));
```

## 25. Python Program for compound interest

# Python3 program to find compound  
# interest for given values.

```
def compound_interest(principle, rate, time):  
  
    # Calculates compound interest  
    Amount = principle * (pow((1 + rate / 100), time))  
    CI = Amount - principle  
    print("Compound interest is", CI)
```

# Driver Code

```
compound_interest(10000, 10.25, 5)
```

**Output:**

```
Compound interest is 6288.946267774416
```

## 26. Python program to print positive numbers in a list

```
# Python program to print positive Numbers in a List

# list of numbers
list1 = [11, -21, 0, 45, 66, -93]

# iterating each number in list
for num in list1:

    # checking condition
    if num >= 0:
        print(num, end = " ")
```

Output:

```
11 0 45 66
```

## 27. Remove multiple elements from a list in Python

```
# Python program to remove multiple
# elements from a list

# creating a list
list1 = [11, 5, 17, 18, 23, 50]

# Iterate each element in list
# and add them in variable total
for ele in list1:
    if ele % 2 == 0:
        list1.remove(ele)

# printing modified list
print("New list after removing all even numbers: ", list1)
```

Output:

```
New list after removing all even numbers: [11, 5, 17, 23]
```

## 28. Python program to interchange first and last elements in a list

```
# Python3 program to swap first
# and last element of a list

)
# Swap function
def swapList(newList):
    size = len(newList)

    # Swapping
    temp = newList[0]
    newList[0] = newList[size - 1]
    newList[size - 1] = temp

    return newList

# Driver code
newList = [12, 35, 9, 56, 24]

print(swapList(newList))
```

**Output:**

```
[24, 35, 9, 56, 12]
```

## 29. Program to accept the strings which contains all vowels

```
# Python program to accept the strings
# which contains all the vowels

# Function for check if string
# is accepted or not
def check(string) :

    string = string.lower()

    # set() function convert "aeiou"
    # string into set of characters
    # i.e.vowels = {'a', 'e', 'i', 'o', 'u'}
    vowels = set("aeiou")

    # set() function convert empty
    # dictionary into empty set
    s = set({})

    # looping through each
    # character of the string
    for char in string :

        # Check for the character is present inside
        # the vowels set or not. If present, then
        # add into the set s by using add method
        if char in vowels :
            s.add(char)
        else:
            pass

    # check the length of set s equal to length
    # of vowels set or not. If equal, string is
    # accepted otherwise not
    if len(s) == len(vowels) :
        print("Accepted")
    else :
        print("Not Accepted")
```

```
# Driver code
if __name__ == "__main__" :

    string = "SEEquoiaL"

    # calling function
    check(string)
```

**Output:**

Accepted

### 30. Convert Snake case to Pascal case

```
# Python3 code to demonstrate working of
# Convert Snake case to Pascal case
# Using title() + replace()

# initializing string
test_str = 'geeksforgeeks_is_best'

# printing original string
print("The original string is : " + test_str)

# Convert Snake case to Pascal case
# Using title() + replace()
res = test_str.replace("_", " ").title().replace(" ", "")

# printing result
print("The String after changing case : " + str(res))
```

**Output:**

The original string is : geeksforgeeks\_is\_best  
The String after changing case : GeeksforgeeksIsBest

### 31. Convert a list of Tuples into Dictionary

# Python code to convert into dictionary

```
def Convert(tup, di):  
    for a, b in tup:  
        di.setdefault(a, []).append(b)  
    return di  
  
# Driver Code  
tups = [("akash", 10), ("gaurav", 12), ("anand", 14),  
        ("suraj", 20), ("akhil", 25), ("ashish", 30)]  
dictionary = {}  
print (Convert(tups, dictionary))
```

Output:

```
{'akash': [10], 'gaurav': [12], 'anand': [14],  
 'ashish': [30], 'akhil': [25], 'suraj': [20]}
```

### 32. Python program to sort a list of tuples by second item

```
# Python program to sort a list of tuples by the second Item
```

```
# Function to sort the list of tuples by its second item
```

```
def Sort_Tuple(tup):
```

```
    # getting length of list of tuples
```

```
    lst = len(tup)
```

```
    for i in range(0, lst):
```

```
        for j in range(0, lst-i-1):
```

```
            if (tup[j][1] > tup[j + 1][1]):
```

```
                temp = tup[j]
```

```
                tup[j]= tup[j + 1]
```

```
                tup[j + 1]= temp
```

```
    return tup
```

```
# Driver Code
```

```
tup = [('for', 24), ('is', 10), ('Geeks', 28),
```

```
       ('Geeksforgeeks', 5), ('portal', 20), ('a', 15)]
```

```
print(Sort_Tuple(tup))
```

## Output:

```
[('Geeksforgeeks', 5), ('is', 10), ('a', 15), ('portal', 20),  
( 'for', 24), ('Geeks', 28)]
```

## 33. Python program to check if a string is palindrome or not

```
# function which return reverse of a string
```

```
def isPalindrome(s):
```

```
    return s == s[::-1]
```

```
# Driver code
```

```
s = "malayalam"
```

```
ans = isPalindrome(s)
```

```
if ans:
```

```
    print("Yes")
```

```
else:
```

```
    print("No")
```



Output :

Yes

### 34. Python program to check whether the string is Symmetrical or Palindrome

```
# Python program to demonstrate
# symmetry and palindrome of the
# string

)
# Function to check whether the
# string is plaindrome or not
def palindrome(a):
)
    # finding the mid, start
    # and last index of the string
    mid = (len(a)-1)//2
    start = 0
    last = len(a)-1
    flag = 0

    # A loop till the mid of the
    # string
    while(start<mid):

        # comparing letters from right
        # from the letters from left
        if (a[start]== a[last]):
```

```

        start += 1
        last -= 1

    else:
        flag = 1
        break;

# Checking the flag variable to
# check if the string is palindrome
# or not
if flag == 0:
    print("The entered string is palindrome")
else:
    print("The entered string is not palindrome")

# Function to check whether the
# string is symmetrical or not
def symmetry(a):

    n = len(a)
    flag = 0

    # Check if the string's length
    # is odd or even
    if n%2:
        mid = n//2 +1
    else:
        mid = n//2

    start1 = 0
    start2 = mid

    while(start1 < mid and start2 < n):

        if (a[start1]== a[start2]):
            start1 = start1 + 1
            start2 = start2 + 1
        else:
            flag = 1
            break

```

```

# Checking the flag variable to
# check if the string is symmetrical
# or not
if flag == 0:
    print("The entered string is symmetrical")
else:
    print("The entered string is not symmetrical")

# Driver code
string = 'amaama'
palindrome(string)
symmetry(string)

```

#### Output:

```

The entered string is palindrome
The entered string is symmetrical

```

### 35. Python Program for Sum of squares of first n natural numbers

```

# Python3 Program to
# find sum of square
# of first n natural
# numbers
)

# Return the sum of
# square of first n
# natural numbers
def squaresum(n) :

    # Iterate i from 1
    # and n finding
    # square of i and
    # add to sum.
    sm = 0
    for i in range(1, n+1) :
        sm = sm + (i * i)

    return sm

```

```
# Driven Program  
n = 4  
print(squaresum(n))
```

**Output:**

30