

# PRACTICAL FILE

**SESSION SEPTEMBER 2022-FEB2023**

*Name: - Nishi Kant Nayak*

*Roll Number: - 202590800142*

*Trade: - Computer Science Engineering*

*Subject: - Computer Programming Using Python*

# Practical – 1

**Aim :-** Getting Started with Python and IDLE in interactive and batch modes.

## What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

## How to Install Python in Windows

1. Select Version of Python to be installed
2. Download Python Executable installed from its official website(<https://python.org/downloads>)
3. Select the desired version of python, the latest release of python is Python 3.11.1

### Python Releases for Windows

- [Latest Python 3 Release - Python 3.11.1](#)

#### Stable Releases

- [Python 3.11.1 - Dec. 6, 2022](#)

**Note that Python 3.11.1 cannot be used on Windows 7 or earlier.**

- [Download Windows embeddable package \(32-bit\)](#)
- [Download Windows embeddable package \(64-bit\)](#)
- [Download Windows embeddable package \(ARM64\)](#)
- [Download Windows installer \(32-bit\)](#)
- [Download Windows installer \(64-bit\)](#)
- [Download Windows installer \(ARM64\)](#)

- [Python 3.10.9 - Dec. 6, 2022](#)

**Note that Python 3.10.9 cannot be used on Windows 7 or earlier.**

#### Pre-releases

- [Python 3.12.0a3 - Dec. 6, 2022](#)

- [Download Windows embeddable package \(32-bit\)](#)
- [Download Windows embeddable package \(64-bit\)](#)
- [Download Windows embeddable package \(ARM64\)](#)
- [Download Windows installer \(32-bit\)](#)
- [Download Windows installer \(64-bit\)](#)
- [Download Windows installer \(ARM64\)](#)

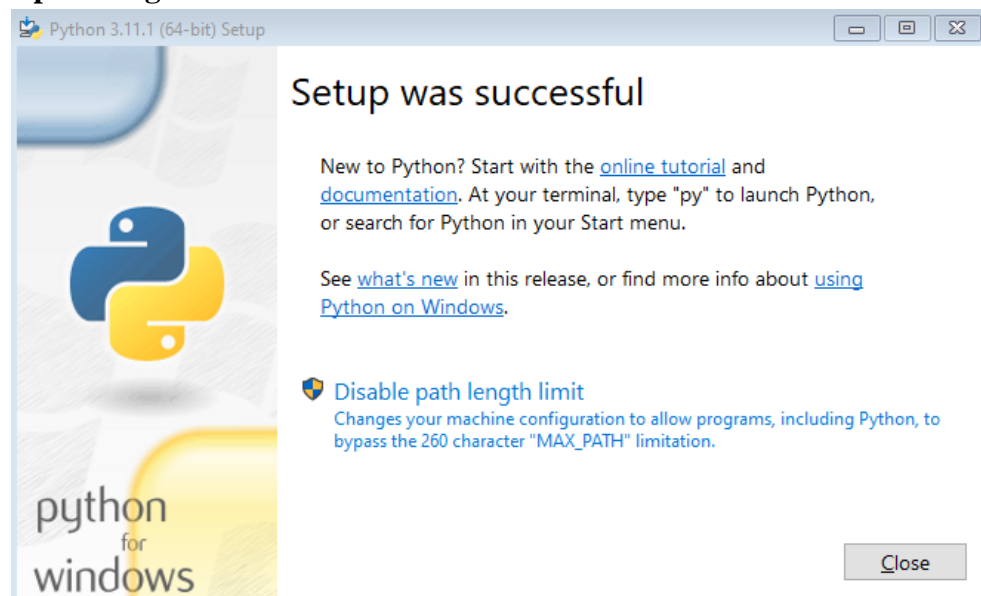
- [Python 3.12.0a2 - Nov. 15, 2022](#)

- [Download Windows embeddable package \(32-bit\)](#)
- [Download Windows embeddable package \(64-bit\)](#)

4. Run the executable installer once downloaded



5. Click on the Install Now option with check in the Add python.exe to PATH
6. After the installation the Screen will prompt to Successful installation and ask for **Disable path length limit**.



Disable Path Length Limit will help to bypass the 260 character MAX\_PATH limitation. Effectively, it enables Python to use long path names

In the python programming language, there are 2 ways in which we can run our code :-

1. Interactive Mode
2. Script Mode (Batch Mode)

## Interactive Mode

Interactive etymologically means “working simultaneously and creating impact of our work on the other’s work”. Interactive mode is based on this ideology only. In the interactive mode as we enter a command and press enter, the very next step we get the output. The output of the code in the interactive mode is influenced by the last command we give. Interactive mode is very convenient for writing very short lines of code. In python it is also known as REPL which stands for Read Evaluate Print Loop. Here, the read function reads the input from the user and stores it in memory. Eval function evaluates the input to get the desired output. Print function outputs the evaluated result. The loop function executes the loop during the execution of the entire program and terminates when our program ends. This mode is very suitable for beginners in programming as it helps them evaluate their code line by line and understand the execution of code well.

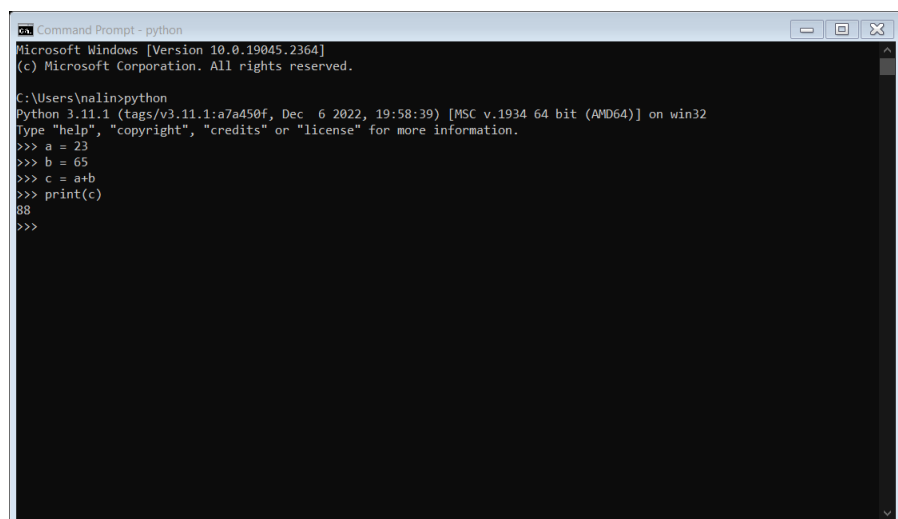
## How to run python code in interactive mode

In order to run our program in the interactive mode, we can use command prompt in windows, terminal in Linux, and macOS. Let us see understand the execution of python code in the command prompt with the help of an example:

**Example 1:** To run python in command prompt type “python”. Then simply type the Python statement on >>> prompt. As we type and press enter we can see the output in the very next line.

Python code for addition on two numbers and we want to get its output. We will declare two variables a & b and store the result in a third variable c. We further print c. All this is done in the command prompt.

```
# Python program to add
two numbers
a = 2
b = 3
# Adding a and b and
storing result in c
c = a + b
# Printing value of c
print(c)
```



```
Command Prompt - python
Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nalin>python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 2
>>> b = 3
>>> c = a+b
>>> print(c)
88
>>>
```

## Script Mode

Script etymologically means a system of writing. In the script mode, a python program can be written in a file. This file can then be saved and executed using the command prompt. We can view the code at any time by opening the file and editing becomes quite easy as we can open and view the entire code as many times as we want. Script mode is very suitable for writing long pieces of code. It is much preferred over interactive mode by experts in the program. The file made in the script mode is by default saved in the Python installation folder and the extension to save a python file is “.py”.

## How to run python code in script mode?

In order to run a code in script mode follow the following steps.

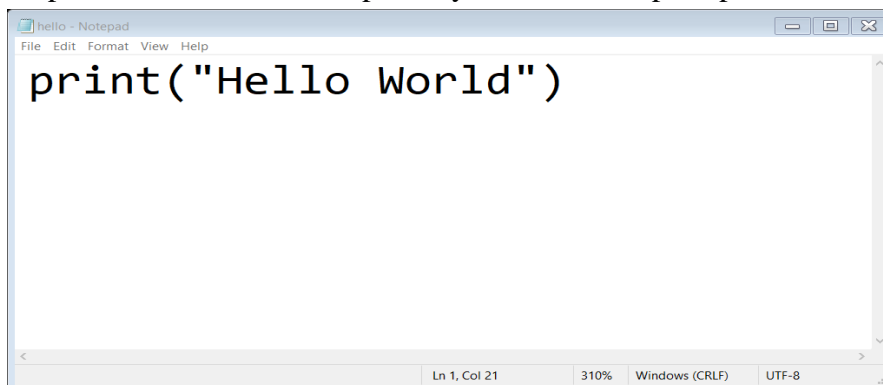
Step 1: Make a file using a text editor. You can use any text editor of your choice(Here I use notepad).

Step 2: After writing the code save the file using “.py” extension.

Step 3: Now open the command prompt and command directory to the one where your file is stored.

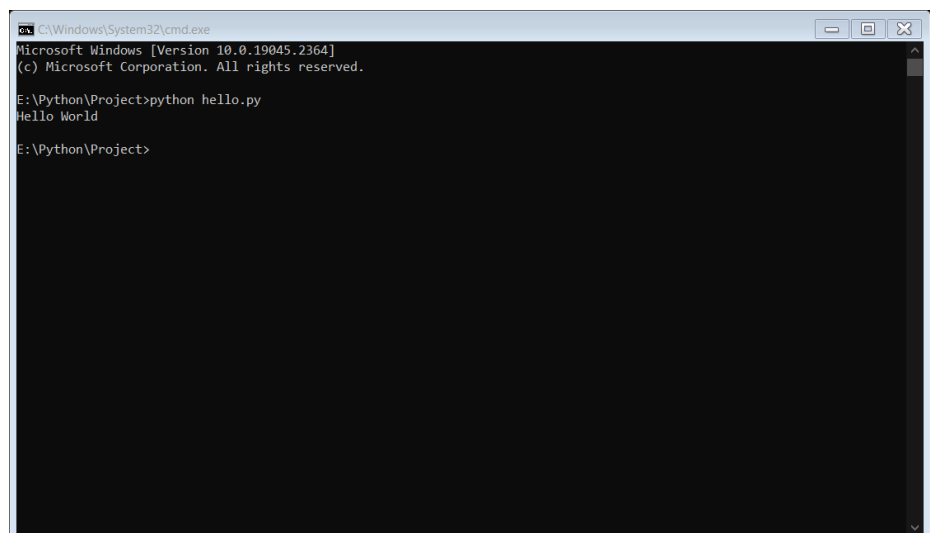
Step 4: Type python “filename.py” and press enter.

Step 5: You will see the output on your command prompt.



```
print("Hello World")
```

**Output:-**



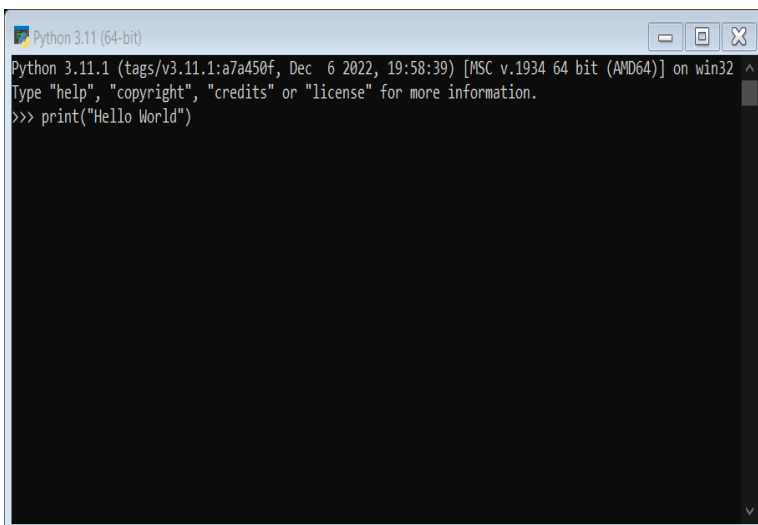
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.

E:\Python\Project>python hello.py
Hello World

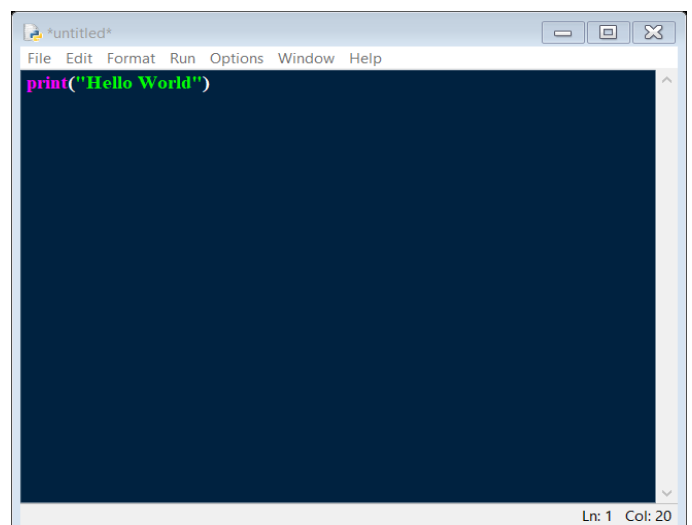
E:\Python\Project>
```

## Difference between Interactive and Scripting Mode:

Interactive Mode	Scripting Mode
It is a way of executing a Python program in which statements are written in command prompt and result is obtained on the same.	In the script mode, the Python program is written in a file. Python interpreter reads the file and then executes it and provides the desired result. The program is compiled in the command prompt,
The interactive mode is more suitable for writing very short programs.	Script mode is more suitable for writing long programs.
Editing of code can be done but it is a tedious task.	Editing of code can be easily done in script mode.
We get output for every single line of code in interactive mode i.e. result is obtained after execution of each line of code.	In script mode entire program is first compiled and then executed.
Code cannot be saved and used in the future.	Code can be saved and can be used in the future.
It is more preferred by beginners.	It is more preferred by experts. Beginners to use script mode.

A screenshot of the Python 3.11.1 (64-bit) interactive shell. The window title is "Python 3.11 (64-bit)". The text inside shows the version and build information: "Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32". It also shows the prompt "Type 'help', 'copyright', 'credits' or 'license' for more information." and the command ">>> print('Hello World')".

```
Python 3.11 (64-bit)
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
```

A screenshot of an untitled Python script file in a text editor. The window title is "\*untitled\*". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code inside is a single line: "print('Hello World')". The status bar at the bottom right shows "Ln: 1 Col: 20".

```
*untitled*
File Edit Format Run Options Window Help
print('Hello World')
Ln: 1 Col: 20
```

## IDLE

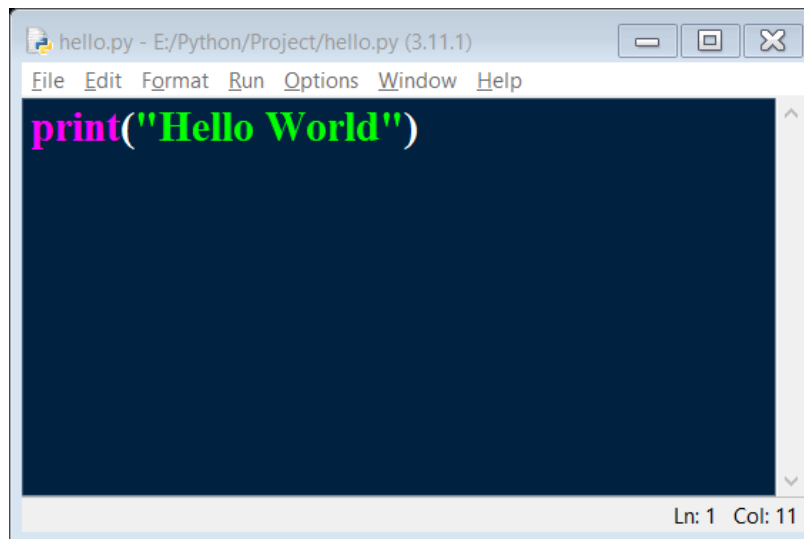
**IDLE** is Python's **I**ntegrated **D**evelopment and **L**earning **E**nvironment. It allows programmers to easily write Python code. Just like Python Shell, IDLE can be used to execute a single statement and create, modify, and execute Python scripts.

IDLE provides a fully-featured text editor to create Python scripts that include features like syntax highlighting, auto completion, and smart indent. It also has a debugger with stepping and breakpoints features. This makes debugging easier.

## How does it work?

The shell is the default mode of operation for Python IDLE. When you click on the icon to open the program, the shell is the first thing that you see:

**Code:**



The screenshot shows the Python IDLE code editor window. The title bar reads "hello.py - E:/Python/Project/hello.py (3.11.1)". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code area contains a single line: `print("Hello World")`, where "print" is in magenta and the string is in green. The status bar at the bottom right indicates "Ln: 1 Col: 11".

**Output: -**



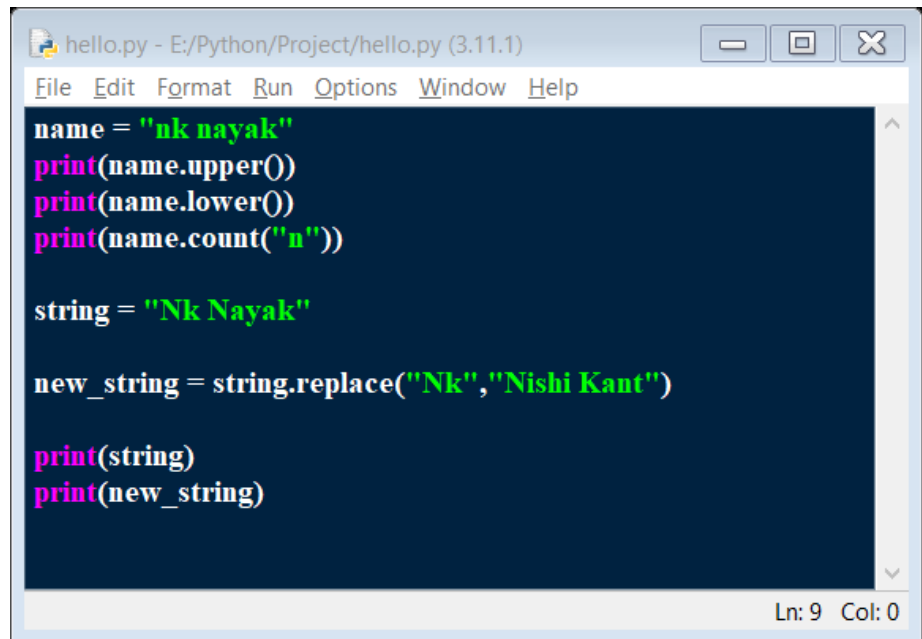
The screenshot shows the Python IDLE Shell window. The title bar reads "IDLE Shell 3.11.1". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The output text is as follows:  
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
== RESTART: E:/Python/Project/hello.py  
=>  
Hello World  
>>>  
The status bar at the bottom right indicates "Ln: 6 Col: 0".

## Practical – 2

**Aim :-** What do the following string methods do?

- Lower
- Count
- Replace

**Code :-**



```
hello.py - E:/Python/Project/hello.py (3.11.1)
File Edit Format Run Options Window Help
name = "nk nayak"
print(name.upper())
print(name.lower())
print(name.count("n"))

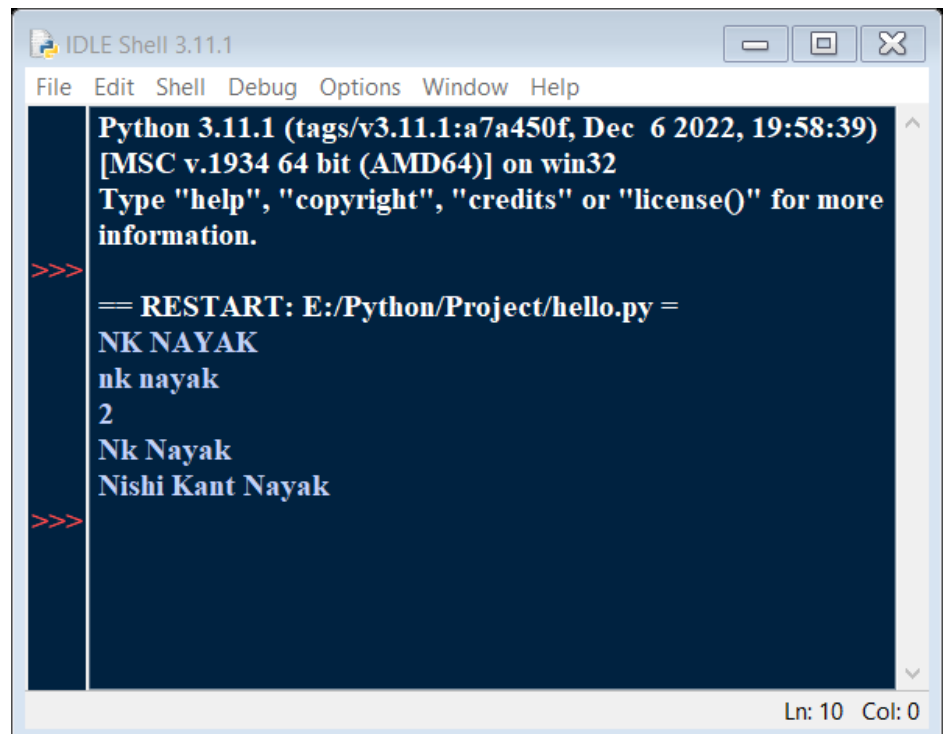
string = "Nk Nayak"

new_string = string.replace("Nk","Nishi Kant")

print(string)
print(new_string)
```

Ln: 9 Col: 0

**Output :-**



```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39)
[MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
== RESTART: E:/Python/Project/hello.py =
NK NAYAK
nk nayak
2
Nk Nayak
Nishi Kant Nayak
>>>
```

Ln: 10 Col: 0



## Practical – 3

**Aim:** - Write instructions to perform each of the steps below:-

- Create a string containing at least five words and store it in a variable.
- Print out the string.
- Convert the string to a list of words using the string split method.
- Sort the list into reverse alphabetical order using some of the list methods (you might need to use `dir(list)` or `help(list)` to find appropriate methods).
- Print out the sorted, reversed list of words.

**Code:-**

```
instruction.py > ...  
1  string = "I am Nishi Kant Nayak"  
2  print(string)  
3  li = list(string.split(" "))  
4  print(li)  
5  li1 = sorted(li)  
6  print(li1)  
7  def Reverse(list1) :  
8      return [element for element in reversed(list1)]  
9  print(Reverse (li1))
```

**Output: -**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  
  
PS E:\Python\NK> python -u "e:\Python\NK\instruction.py"  
I am Nishi Kant Nayak  
['I', 'am', 'Nishi', 'Kant', 'Nayak']  
['I', 'Kant', 'Nayak', 'Nishi', 'am']  
['am', 'Nishi', 'Nayak', 'Kant', 'I']  
PS E:\Python\NK>
```

## Practical – 4

**Aim -** Write a program that determines whether the number is prime.

**Code -**

```
prime_number.py > [?] num
1  num = int(input("Enter the Number to be checked :- "))
2  a = 2
3  while (a<num):
4      if num % a == 0:
5          print(f"The number, {num} is not a prime number")
6          break
7      a = a+1
8
9  if num == a:
10     print(f"The Number is prime")
```

**Output :-**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\prime_number.py"
Enter the Number to be checked :- 15
The number, 15 is not a prime number
PS E:\Python\NK> python -u "e:\Python\NK\prime_number.py"
Enter the Number to be checked :- 13
The Number is prime
PS E:\Python\NK> █
```

## Practical – 5

**Aim:-** Find all numbers which are multiple of 17, but not the multiple of 5, between 2000 and 2500.

**Code: -**

```
Multiples.py > ...
1  ''' Find all numbers which are multiple of 17,
2  but not the multiple of 5, between 2000 and 2500.'''
3  num = 17
4  for i in range(2000,2500):
5      if i % num == 0:
6          if i % 5:
7              print(i)
8
```

**Output: -**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\Multiples.py"
2006
2023
2057
2074
2091
2108
2142
2159
2176
2193
2227
2244
```

## Practical – 6

**Aim:** - Swap two integer numbers using a temporary variable. Repeat the exercise using the code format: a, b = b,

- a. Verify your results in both the cases.

**Code:** -

```
swap.py > ...  
1  a = int(input("ENter the Number A: "))  
2  b = int(input("ENter the Number B: "))  
3  temp = a  
4  a = b  
5  b = temp  
6  print("Result of using temporoary variable:" ,a)  
7  print("Result of using temporoary variable:" ,b)  
8  
9  a = int(input("Enter the value of A: "))  
10 b = int(input("Enter the value of B: "))  
11 a,b = b,a  
12 print("Result of A after Given equation is " ,a)  
13 print("Result of B after Given equation is " ,b)  
14 print("Both result are same")
```

**Output:** -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  
  
PS E:\Python\NK> python -u "e:\Python\NK\swap.py"  
Enter the Number A: 1  
Enter the Number B: 2  
Result of using temporoary variable: 2  
Result of using temporoary variable: 1  
Enter the value of A: 3  
Enter the value of B: 4  
Result of A after Given equation is 4  
Result of B after Given equation is 3  
Both result are same  
PS E:\Python\NK> 
```

## Practical – 7

**Aim:** - Find the largest of n numbers, using a user defined function largest().

**Code:** -

```
largest.py > ...
1  def largest(list):
2      max = list[0]
3      for l in list:
4          if l > max:
5              max = l
6      return max
7
8  list = [50,80,90,100,99,150]
9  print("Largest element is: ", largest(list))
```

**Output:** -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\largest.py"
Largest element is:  150
PS E:\Python\NK>
```

## Practical – 8

**Aim:** - Write a function myReverse() which receives a string as an input and returns the reverse of the string.

**Code:** -

```
reverse.py > ...  
1  def myReverse(n):  
2      return n[::-1]  
3  
4  word = myReverse(input())  
5  print(word)
```

**Output:** -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  
  
PS E:\Python\NK> python -u "e:\Python\NK\largest.py"  
Largest element is: 150  
PS E:\Python\NK> python -u "e:\Python\NK\reverse.py"  
My Name is Nk Nayak  
kayaN kN si emaN yM  
PS E:\Python\NK> █
```

## Practical – 9

**Aim:** - Check if a given string is palindrome or not.

**Code:** -

```
palindrome.py > ...
1 word = input("Check your word for palindrome, Enter the word: - ")
2 if word == word[::-1]:
3     print("This word is Palindrome")
4 else:
5     print(f"The word {word} is not a palindrome")
```

**Output:** -

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

GITLENS

```
PS E:\Python\NK> python -u "e:\Python\NK\palindrome.py"
Check your word for palindrome, Enter the word: - madam
This word is Palindrome
PS E:\Python\NK> python -u "e:\Python\NK\palindrome.py"
Check your word for palindrome, Enter the word: - Marvel
The word Marvel is not a palindrome
PS E:\Python\NK> █
```

## Practical – 10

**Aim:** - WAP to convert Celsius to Fahrenheit.

**Code:** -

```
temperature.py > ...  
1  '''assume farahenheit as F and Celsius as C'''  
2  F = int(input("Enter the temeperature in Farahenheit: - "))  
3  C = (F-32)*5.0/9.0  
4  print(C)
```

**Output:** -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS  
  
PS E:\Python\NK> python -u "e:\Python\NK\temperature.py"  
Enter the temeperature in Farahenheit: - 32  
0.0  
PS E:\Python\NK> python -u "e:\Python\NK\temperature.py"  
Enter the temeperature in Farahenheit: - 99  
37.22222222222222  
PS E:\Python\NK> █
```



## Practical – 11

**Aim:** - Find the ASCII Value of charades

**Code:** -

ASCII.py > ...

```
1 a = input("Enter the Number for ASCII :- ")
2 print(f"ASCII Value of {a} is", ord(a))
```

**Output:** -

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    GITLENS

```
PS E:\Python\NK> python -u "e:\Python\NK\ASCII.py"
Enter the Number for ASCII :- 9
ASCII Value of 9 is 57
PS E:\Python\NK> █
```

## Practical – 12

**Aim:** - WAP for a simple calculator.

**Code:** -

```
calculator.py > ...
1  def add(x, y):
2      |   return x + y
3  def subtract(x, y):
4      |   return x - y
5  def multiply(x, y):
6      |   return x * y
7  def divide(x, y):
8      |   return x / y
9  print("Select operation.")
10 print("1.Add")
11 print("2.Subtract")
12 print("3.Multiply")
13 print("4.Divide")
14 while True:
15     choice = input("Enter choice(1/2/3/4): ")
16     if choice in ('1', '2', '3', '4'):
17         num1 = int(input("Enter first number: "))
18         num2 = int(input("Enter second number: "))
19         if choice == '1':
20             |   print(num1, "+", num2, "=", add(num1, num2))
21         elif choice == '2':
22             |   print(num1, "-", num2, "=", subtract(num1, num2))
23         elif choice == '3':
24             |   print(num1, "*", num2, "=", multiply(num1, num2))
25         elif choice == '4':
26             |   print(num1, "/", num2, "=", divide(num1, num2))
27         next_calculation = input("Let's do next calculation? (yes/no): ")
28         if next_calculation == "no":
29             |   break
30     else:
31         |   print("Invalid Input")
```

**Output :** - On the Next Page

Addition:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\calculator.py"
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 1
Enter first number: 6
Enter second number: 5
6 + 5 = 11
PS E:\Python\NK> █
```

Subtraction:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\calculator.py"
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 2
Enter first number: 3
Enter second number: 2
3 - 2 = 1
PS E:\Python\NK> █
```

Multiply:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\calculator.py"
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 3
Enter first number: 5
Enter second number: 6
5 * 6 = 30
PS E:\Python\NK> █
```

Divide:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

PS E:\Python\NK> python -u "e:\Python\NK\calculator.py"
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 4
Enter first number: 6
Enter second number: 2
6 / 2 = 3.0
PS E:\Python\NK> █
```