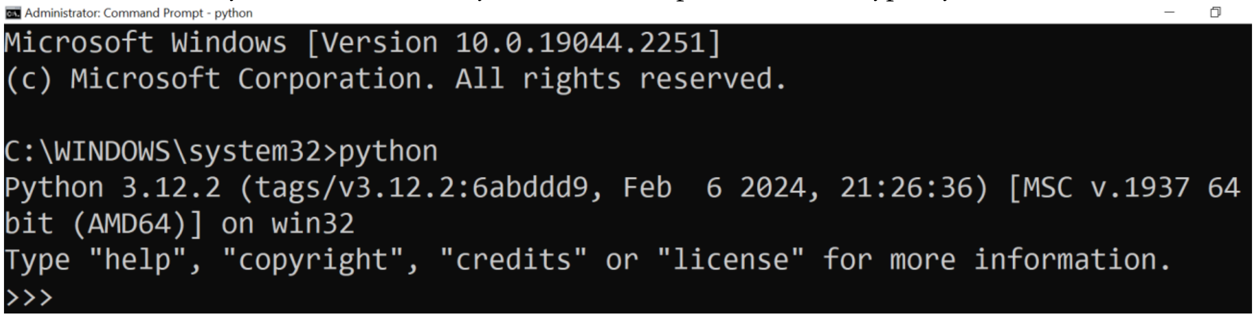
Requirements for Programming in Python Class

1. Download and Install Python 3.12.2 (Stable version)

https://www.python.org/downloads/

To confirm that Python is installed in your machine, open CMD and type Python.



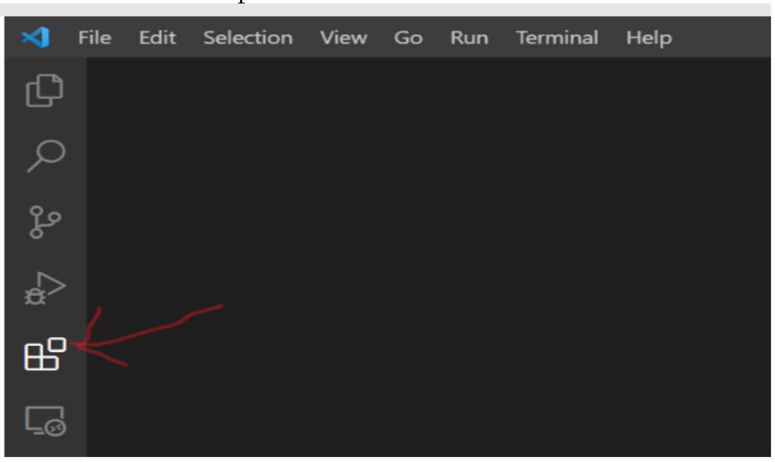
```
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>python
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

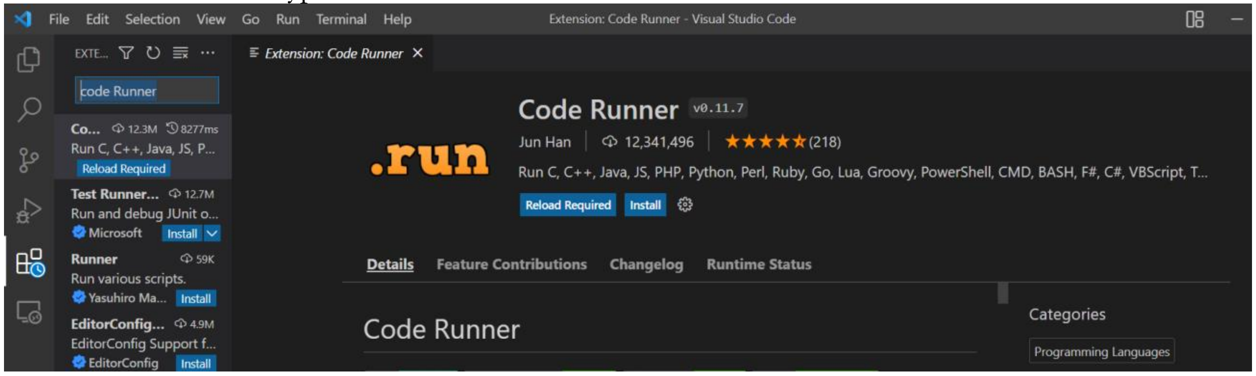
2. Download and Install Editor (VSCode)

https://code.visualstudio.com/download

3. Open VScode and do the configurations below.
   i) Click on Extension part.

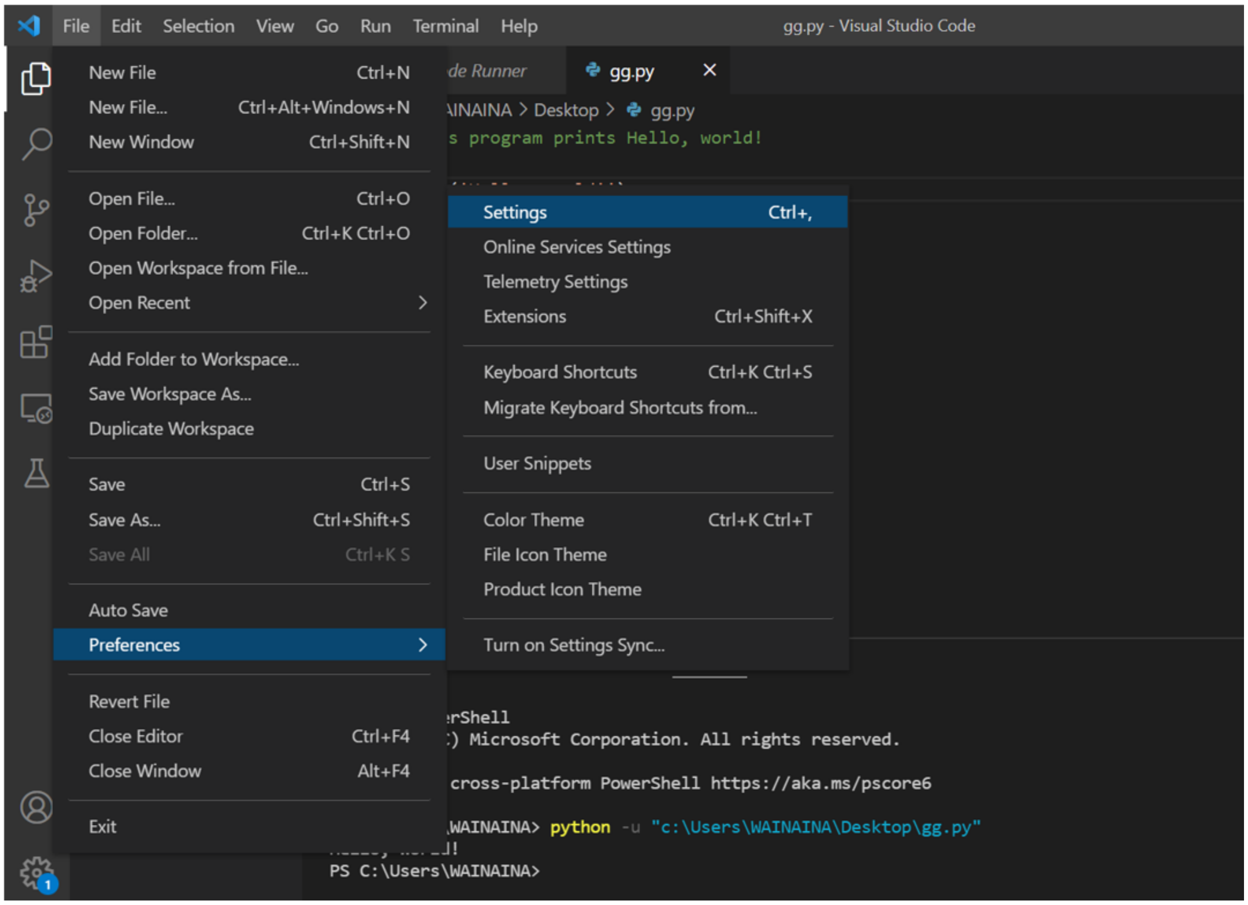


   ii) Within the search tab type “code Runner”



   iii) Click install to add the extension.

4. To allow the VSCode read the input from the user follow the steps below:
   i) Click File, then preference.
   ii) Select settings.



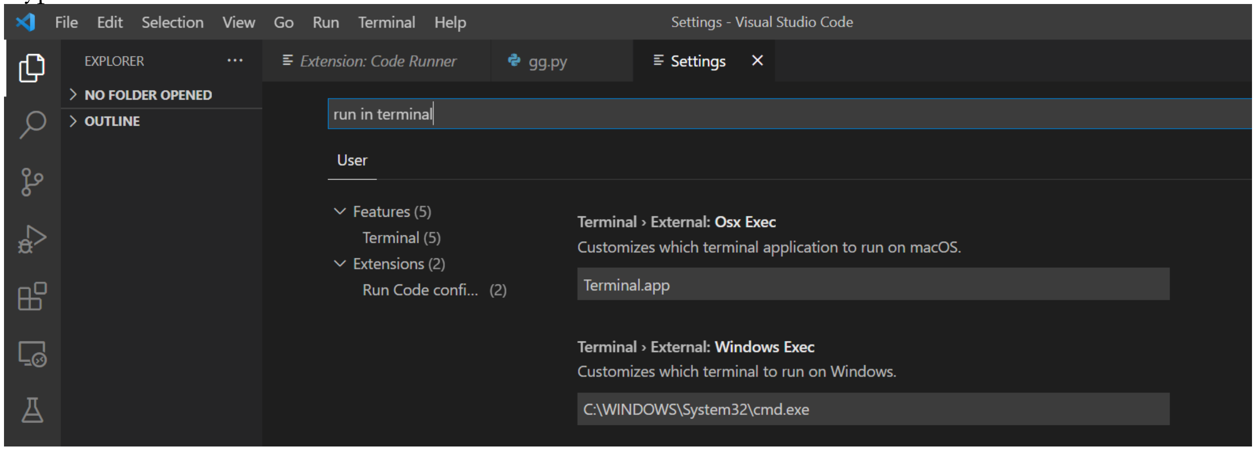
```
s program prints Hello, world!
```

```
rShell
) Microsoft Corporation. All rights reserved.

cross-platform PowerShell https://aka.ms/pscore6

WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
PS C:\Users\WAINAINA>
```

   iii) Type run in terminal.

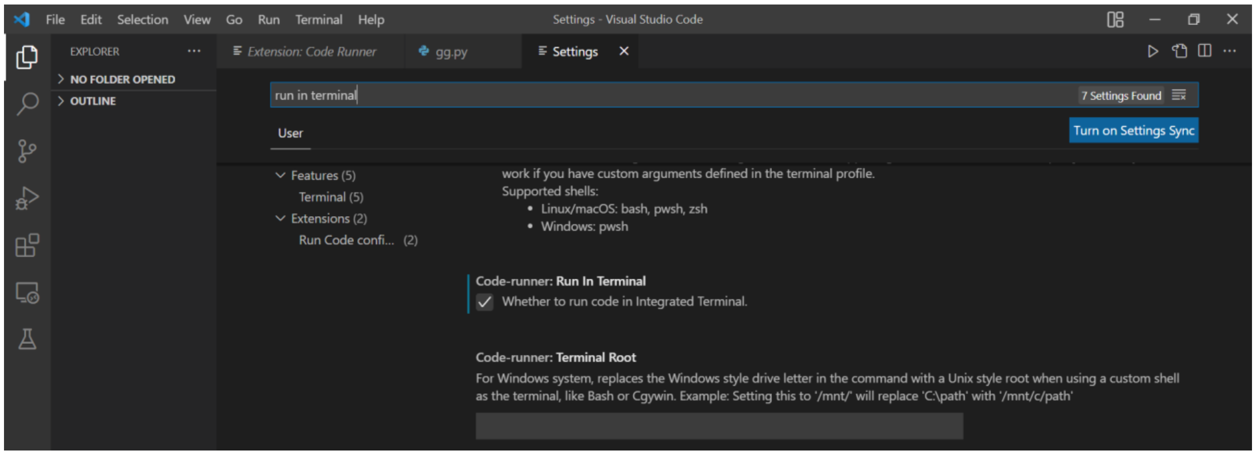


run in terminal

Terminal › External: Osx Exec
Customizes which terminal application to run on macOS.
Terminal.app

Terminal › External: Windows Exec
Customizes which terminal to run on Windows.
C:\WINDOWS\System32\cmd.exe

iv) Scroll down a little bit and find code runner – run in terminal and check the box next to it.



work if you have custom arguments defined in the terminal profile.
Supported shells:
- Linux/macOS: bash, pwsh, zsh
- Windows: pwsh

Code-runner: Run In Terminal
✓ Whether to run code in Integrated Terminal.

Code-runner: Terminal Root
For Windows system, replaces the Windows style drive letter in the command with a Unix style root when using a custom shell as the terminal, like Bash or Cgywin. Example: Setting this to '/mnt/' will replace 'C:\path' with '/mnt/c/path'

v) Cancel the setting and go back to the editor and continue writing your codes.

What is Python?

Python is a popular programming language. It is a high-level object-oriented programming language for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

Syntax Error:

```
if 5 > 2:
print("Five is greater than two!")
```

The number of spaces is up to you as a programmer, the most common use is four, but it has to be at least one.

Example

```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than two!")
```

Python Comment

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code

Creating a Comment

Comments starts with a #, and Python will ignore them:

Example

```
#This is a comment
print("Hello, World!")
```

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

Example

```
print("Hello, World!") #This is a comment
```

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code:

Example

```
#print("Hello, World!")
print("Server side, Programming!")
```

Multi Line Comments

Python does not really have a syntax for multi line comments.

To add a multiline comment you could insert a # for each line:

Example

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

Or, not quite as intended, you can use a multiline string.

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

Example

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

```
x = 5
y = "John"
print(x)
print(y)
```

Single or Double Quotes?

String variables can be declared either by using single or double quotes:

Example

```
x = "John"
# is the same as
x = 'John'
```

Case-Sensitive

Variable names are case-sensitive.

Example

This will create two variables:

```
a = 4
A = "Sally"
#A will not overwrite a
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Example

Legal variable names:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Output Variables

The Python print() function is often used to output variables.

Example

```
x = "Python is awesome"
print(x)
```

In the print() function, you output multiple variables, separated by a comma:

Example

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

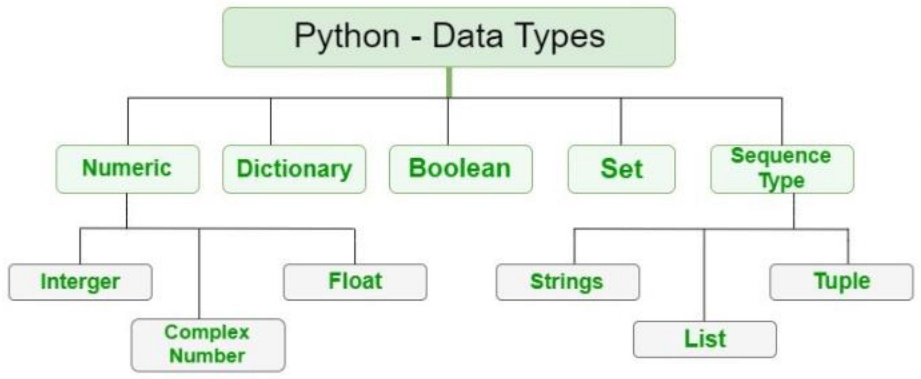
You can also use the + operator to output multiple variables:

Example

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

Python Data Types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.



a) **Integers** – This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.

b) **Float** – This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.

c) **Complex Numbers** – Complex number is represented by complex class. It is specified as *(real part) + (imaginary part)j*. For example – 2+3j

d) **String** - In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class.

e) **List** - Lists are just like the arrays, declared in other languages which is a ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

f) **Tuple** - Just like list, tuple is also an ordered collection of Python objects. The only difference between tuple and list is that tuples are immutable i.e. tuples cannot be modified after it is created. It is represented by tuple class.

g) **Boolean** - Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

h) **Set** - In Python, Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

i) **Dictionary** - Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a 'comma'.

EXAMPLES OF PYTHON PROGRAMS

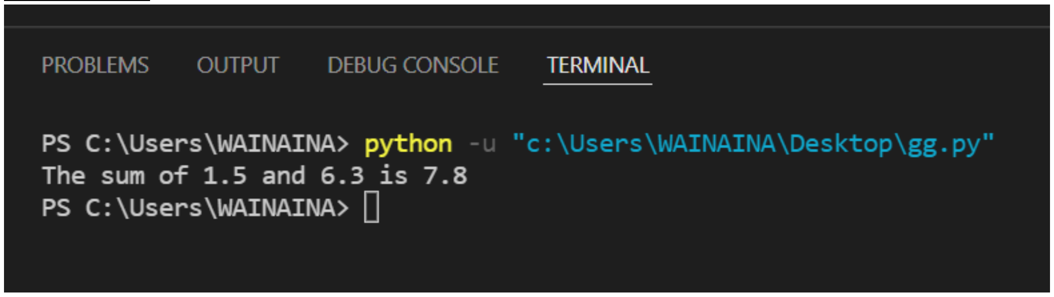
1. Add two numbers initialized by the user:

```python
# This program adds two numbers

num1 = 1.5
num2 = 6.3
# Add two numbers
sum = num1 + num2

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

OUTPUT:



```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
The sum of 1.5 and 6.3 is 7.8
PS C:\Users\WAINAINA> 
```

2. Add two numbers Entered by the user:

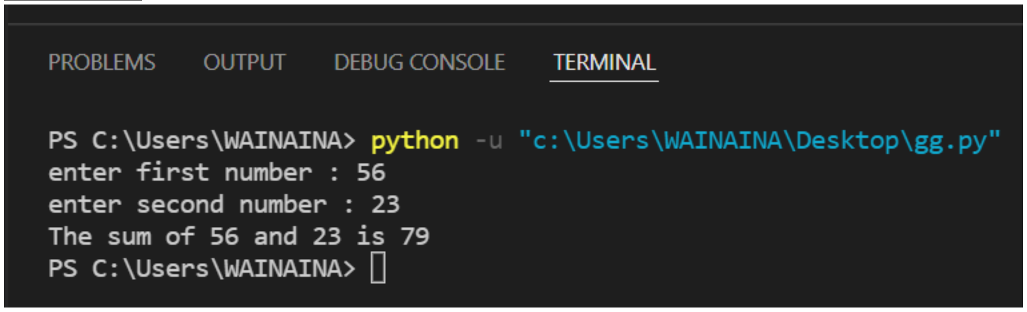
```python
# This program adds two numbers entered by the user

a=int(input("enter first number : "))

b=int(input("enter second number : "))
# Add two numbers
sum = a + b

# Display the sum
print('The sum of {0} and {1} is {2}'.format(a, b, sum))
```

OUTPUT



```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
enter first number : 56
enter second number : 23
The sum of 56 and 23 is 79
PS C:\Users\WAINAINA> 
```

3. Square root of a value initialized by the user (using **):

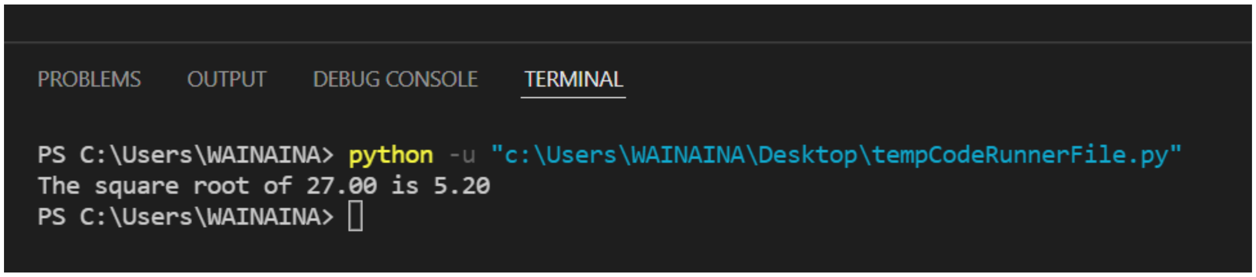
The ** exponent operator works for all positive real numbers.

```python
# Python Program to calculate the square root

ict = 27

ans = ict ** 0.5
print('The square root of %0.2f is %0.2f'%(ict ,ans))
```

OUTPUT:



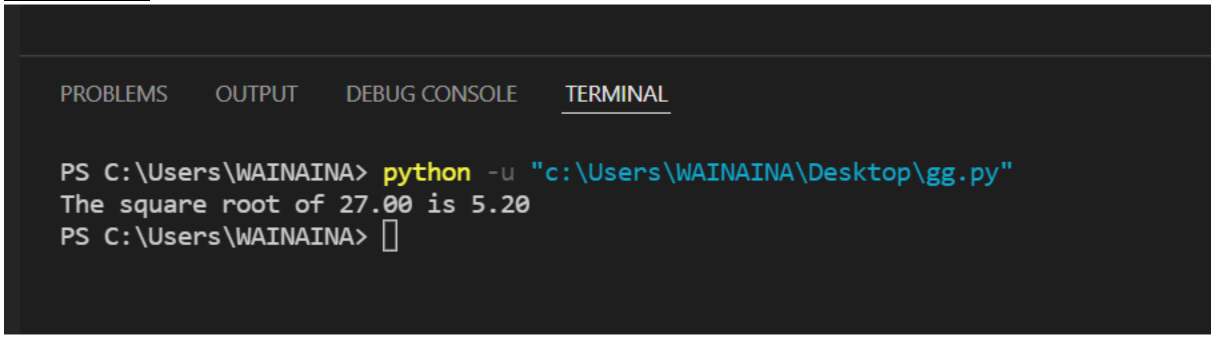
```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\tempCodeRunnerFile.py"
The square root of 27.00 is 5.20
PS C:\Users\WAINAINA> 
```

4. Square root of a value initialized by the user (using math.sqrt() function):

```python
# Python Program to calculate the square root
# import the math module
import math
ict = 27

ans =math.sqrt(ict)
print('The square root of %0.2f is %0.2f'%(ict ,ans))
```

OUTPUT:



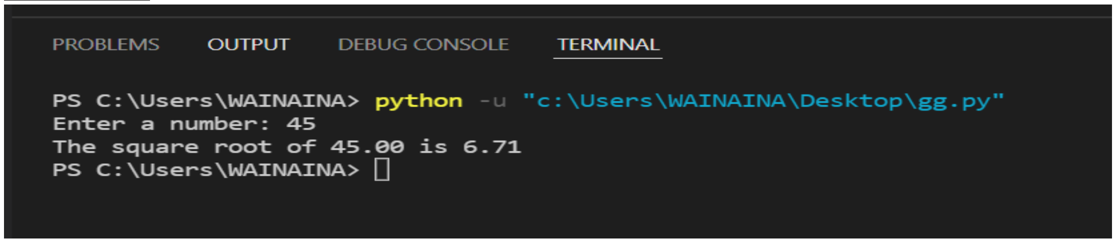
```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
The square root of 27.00 is 5.20
PS C:\Users\WAINAINA> 
```

5. Square root of a value entered by the user (using math.sqrt () function):

```
# Python Program to calculate the square root
# import the math module
import math

ict = float(input('Enter a number: '))
ans =math.sqrt(ict)
print('The square root of %0.2f is %0.2f'%(ict ,ans))
```

OUTPUT:



```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
Enter a number: 45
The square root of 45.00 is 6.71
PS C:\Users\WAINAINA> []
```

6.Python to Calculate the area of a Triangle:

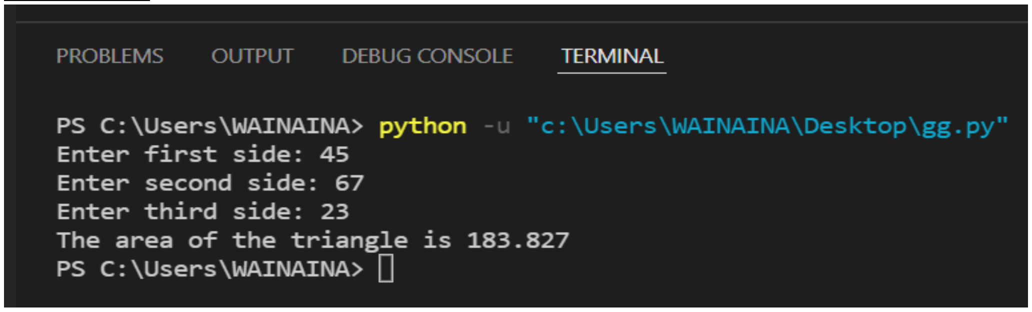
The area A of a triangle is obtained using the formula A = $\sqrt{S(S-a)(S-b)(S-c)}$ where a, b and c are the dimensions of a triangle and S =$\frac{a+b+c}{2}$. Write a Python program that would prompt for the three dimensions of a triangle, computes the area and displays the results to the nearest 3 decimal places.

```
# Python Program to find the area of triangle
# Take inputs from the user
a = float(input('Enter first side: '))
b = float(input('Enter second side: '))
c = float(input('Enter third side: '))

# calculate the semi-perimeter
s = (a + b + c) / 2

# calculate the area
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is %0.3f' %area)
```

OUTPUT:

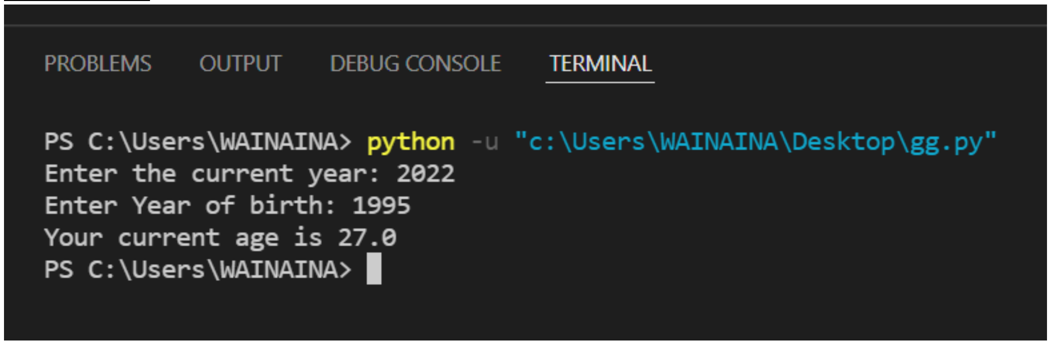


```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
Enter first side: 45
Enter second side: 67
Enter third side: 23
The area of the triangle is 183.827
PS C:\Users\WAINAINA> []
```

7. A program to calculate your age:

```python
# Python Program to calculate your age
# Take inputs from the user
current = float(input('Enter the current year: '))
yob = float(input('Enter Year of birth: '))
# calculate the age
age=current-yob
print('Your current age is {0}'.format(age))
```

OUTPUT:



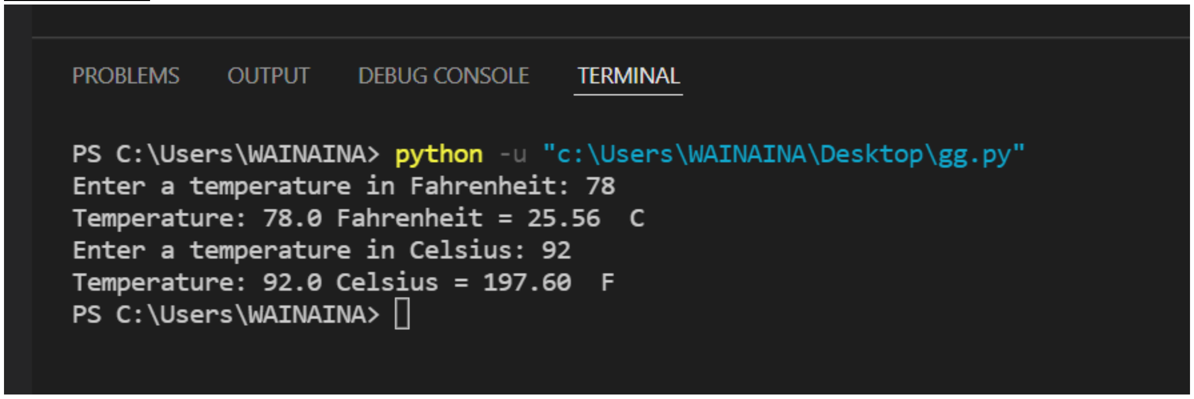
```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
Enter the current year: 2022
Enter Year of birth: 1995
Your current age is 27.0
PS C:\Users\WAINAINA> 
```

8. Write a Python program to convert temperatures to and from celsius, fahrenheit.
Formula :
$^\circ F = (^\circ C \times 9/5) + 32$
$^\circ C = (^\circ F - 32) \times 5/9$
[where c = temperature in celsius and f = temperature in fahrenheit]:

```python
Fahrenheit = float(input("Enter a temperature in Fahrenheit: "))
Celsius = (Fahrenheit - 32) * 5.0/9.0
print ("Temperature:", Fahrenheit, "Fahrenheit = %0.2f" %Celsius, " C")

Celsius = float(input("Enter a temperature in Celsius: "))
Fahrenheit = 9.0/5.0 * Celsius + 32
print("Temperature:", Celsius, "Celsius = %0.2f" %Fahrenheit, " F")
```

OUTPUT:



```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
Enter a temperature in Fahrenheit: 78
Temperature: 78.0 Fahrenheit = 25.56  C
Enter a temperature in Celsius: 92
Temperature: 92.0 Celsius = 197.60  F
PS C:\Users\WAINAINA> 
```

9. A program to calculate cuberoot of a number (using math.ceil ()function):

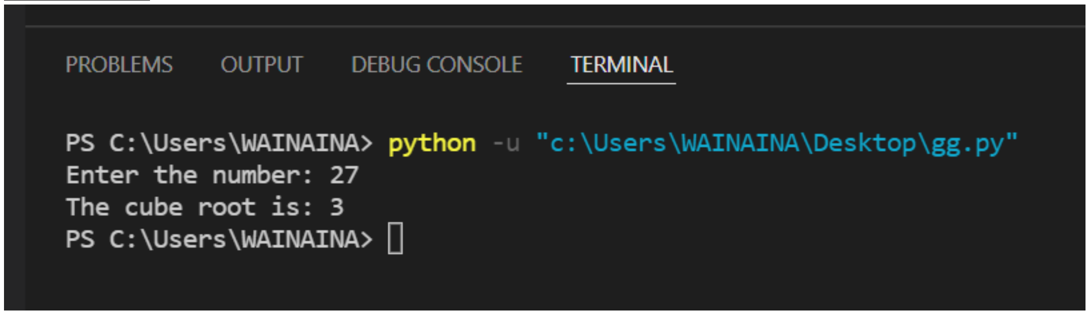
```python
import math

number = int(input("Enter the number: "))

cuberoot = math.ceil(number ** (1/3))

print("The cube root is:",cuberoot)
```

OUTPUT:



```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\gg.py"
Enter the number: 27
The cube root is: 3
PS C:\Users\WAINAINA>
```

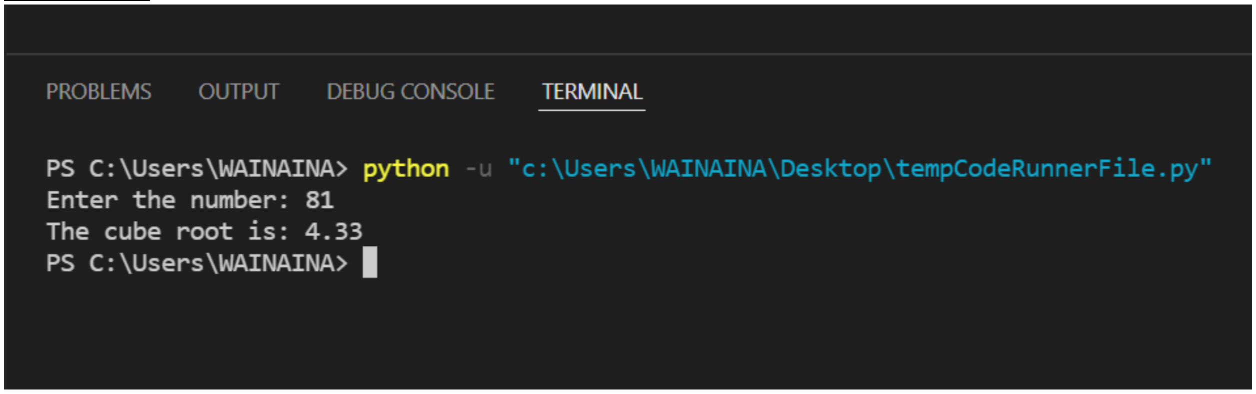
10. A program to calculate cuberoot of a number (using **):

```python
number = int(input("Enter the number: "))

cuberoot = number**(1/3)

print("The cube root is: %0.2f" %cuberoot)
```

OUTPUT:

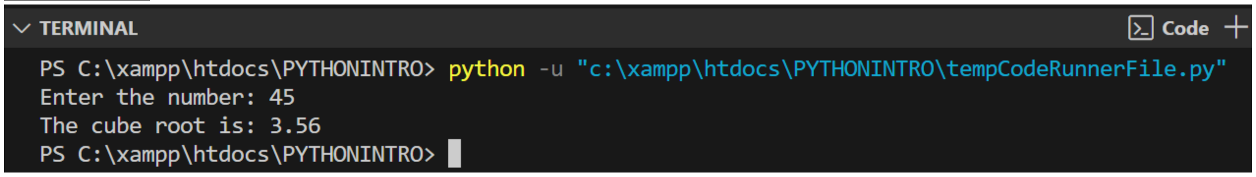


```
PS C:\Users\WAINAINA> python -u "c:\Users\WAINAINA\Desktop\tempCodeRunnerFile.py"
Enter the number: 81
The cube root is: 4.33
PS C:\Users\WAINAINA>
```

11. A program to calculate cube root of a number (using Round Function):

```python
import math
number = int(input("Enter the number: "))
cuberoot = round(number ** (1/3), 2)
print("The cube root is:", cuberoot)
```

OUTPUT:



```
PS C:\xampp\htdocs\PYTHONINTRO> python -u "c:\xampp\htdocs\PYTHONINTRO\tempCodeRunnerFile.py"
Enter the number: 45
The cube root is: 3.56
PS C:\xampp\htdocs\PYTHONINTRO>
```

12.

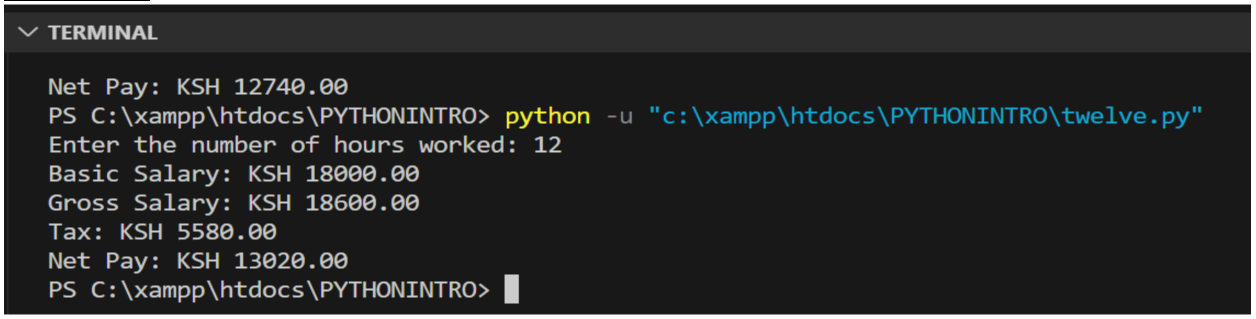
A program is required to calculate wages for employees of ABC Ltd. Using the following formulae.

i. Basic salary = No of Hours * hourly rate
ii. Lunch Allowance = 200
iii. Gross Salary = Basic Pay + Lunch Allowance
iv. Income Tax charged on gross pay is 30% of the gross
v. Net pay = Gross pay – Tax

Hourly rate and lunch allowance are constant values. The number of hours worked should be read from the keyboard. Write a program to process salary for an employee.

```python
# Constants
hourly_rate = 1500
lunch_allowance = 600
tax_rate = 0.30
# Input number of hours worked from the user
hours_worked = float(input("Enter the number of hours worked: "))
# Calculate Basic Pay
basic_salary = hours_worked * hourly_rate
# Calculate Gross Salary
gross_salary = basic_salary + lunch_allowance
# Calculate Tax
tax = gross_salary * tax_rate
# Calculate Net Pay
net_pay = gross_salary - tax
# Display the calculated values
print("Basic Salary: KSH", format(basic_salary, ".2f"))
print("Gross Salary: KSH", format(gross_salary, ".2f"))
print("Tax: KSH", format(tax, ".2f"))
print("Net Pay: KSH", format(net_pay, ".2f"))
```

OUTPUT:



```
Net Pay: KSH 12740.00
PS C:\xampp\htdocs\PYTHONINTRO> python -u "c:\xampp\htdocs\PYTHONINTRO\twelve.py"
Enter the number of hours worked: 12
Basic Salary: KSH 18000.00
Gross Salary: KSH 18600.00
Tax: KSH 5580.00
Net Pay: KSH 13020.00
PS C:\xampp\htdocs\PYTHONINTRO>
```