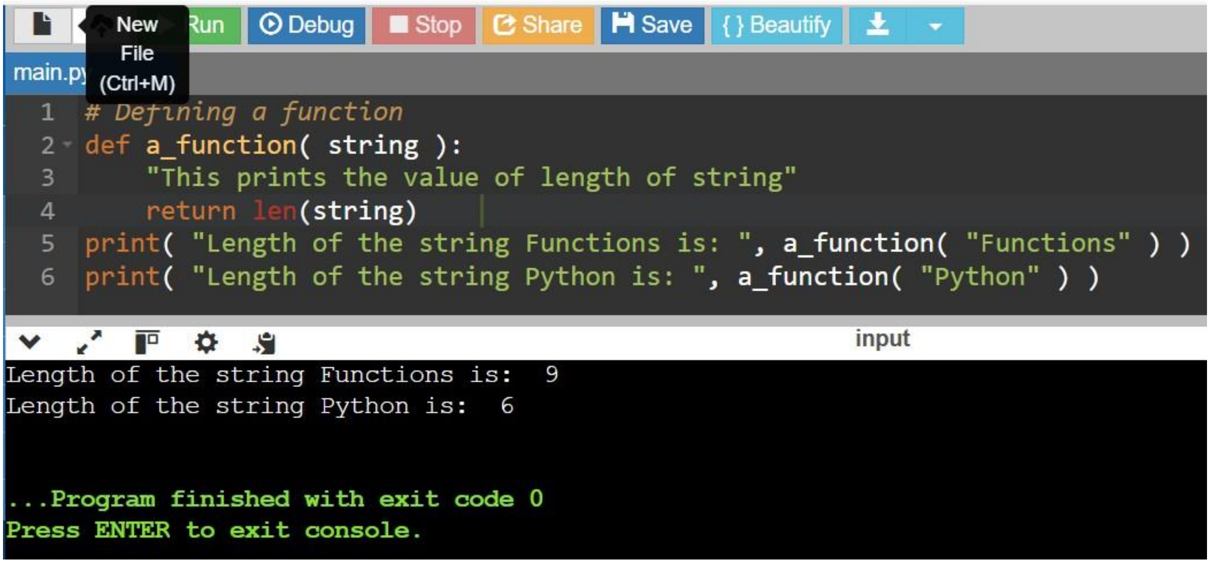
PYTHON FUNCTIONS

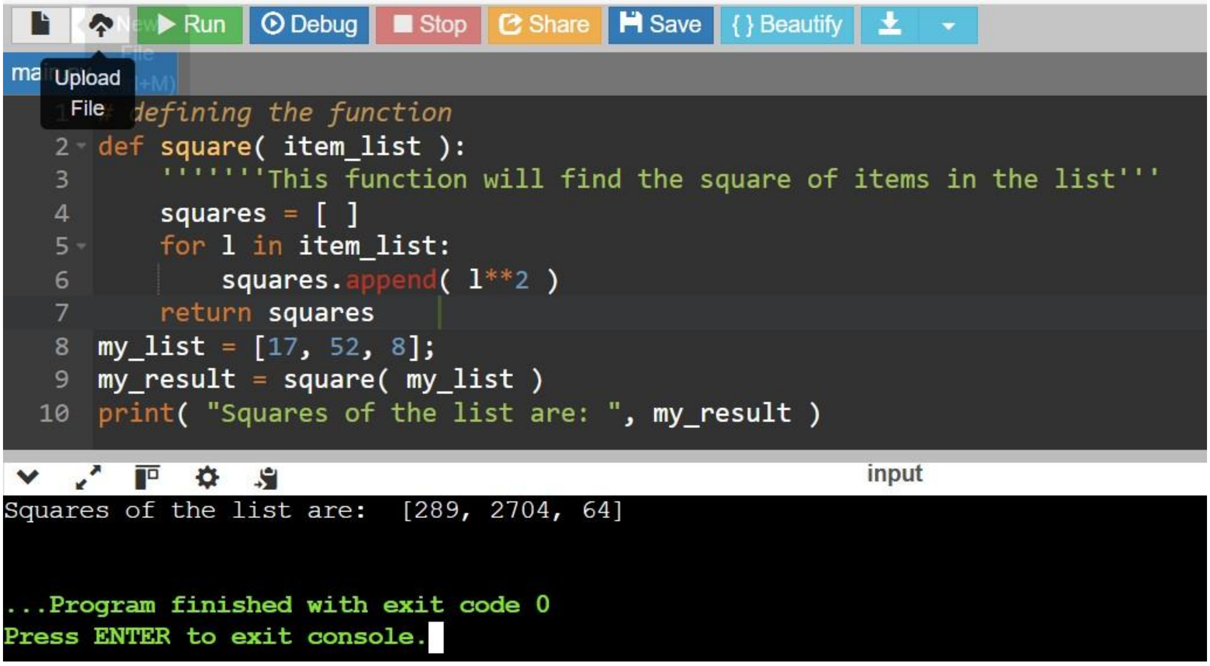
1. Calling a function



```python
# Defining a function
def a_function( string ):
    "This prints the value of length of string"
    return len(string)
print( "Length of the string Functions is: ", a_function( "Functions" ) )
print( "Length of the string Python is: ", a_function( "Python" ) )
```

```
Length of the string Functions is:  9
Length of the string Python is:  6


...Program finished with exit code 0
Press ENTER to exit console.
```

2. Pass by Reference Vs Pass by Value

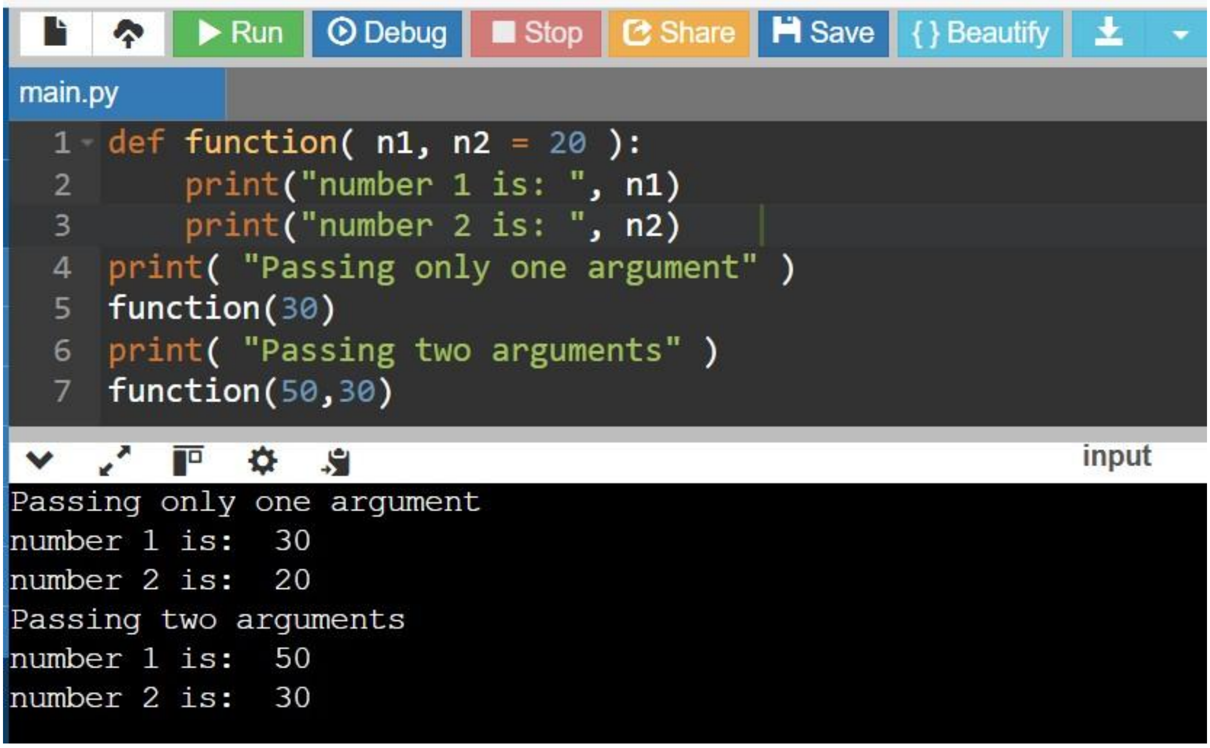


```python
# defining the function
def square( item_list ):
    '''''''This function will find the square of items in the list'''
    squares = [ ]
    for l in item_list:
        squares.append( l**2 )
    return squares
my_list = [17, 52, 8];
my_result = square( my_list )
print( "Squares of the list are: ", my_result )
```

```
Squares of the list are:  [289, 2704, 64]


...Program finished with exit code 0
Press ENTER to exit console.
```

FUNCTION ARGUMENTS

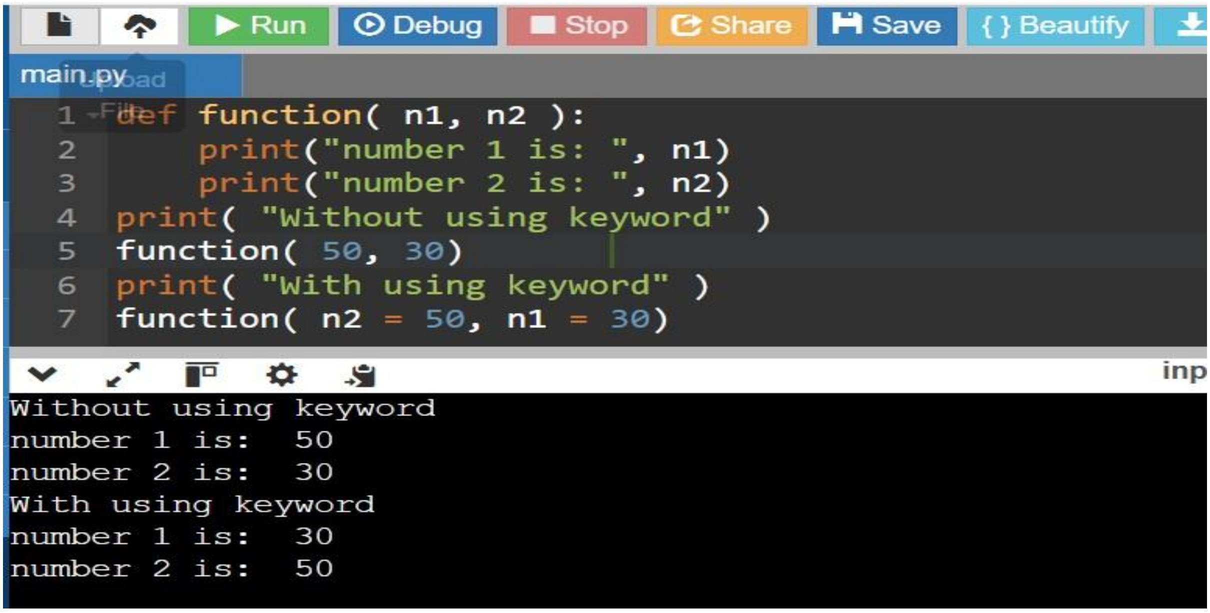
1. Default arguments



```python
def function( n1, n2 = 20 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
print( "Passing only one argument" )
function(30)
print( "Passing two arguments" )
function(50,30)
```

```
Passing only one argument
number 1 is:  30
number 2 is:  20
Passing two arguments
number 1 is:  50
number 2 is:  30
```

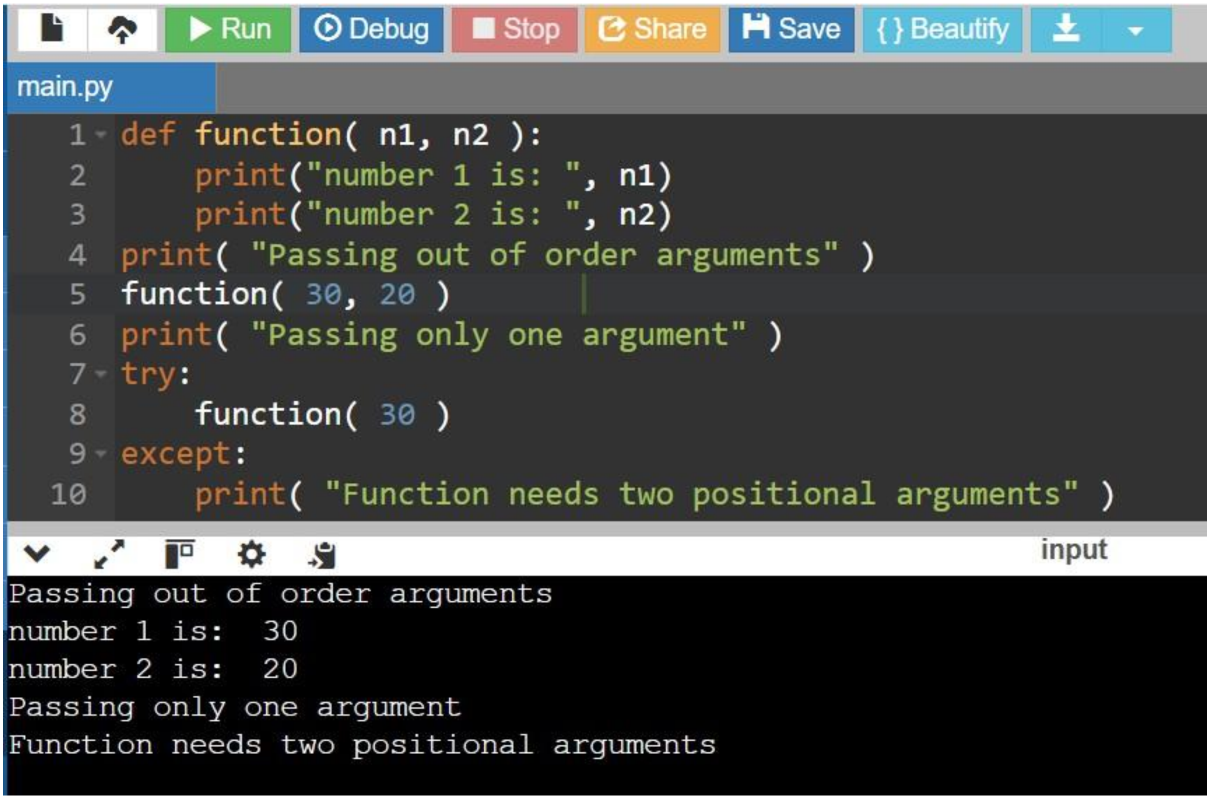
2.Keyword arguments



```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
print( "Without using keyword" )
function( 50, 30)
print( "With using keyword" )
function( n2 = 50, n1 = 30)
```

```
Without using keyword
number 1 is:  50
number 2 is:  30
With using keyword
number 1 is:  30
number 2 is:  50
```

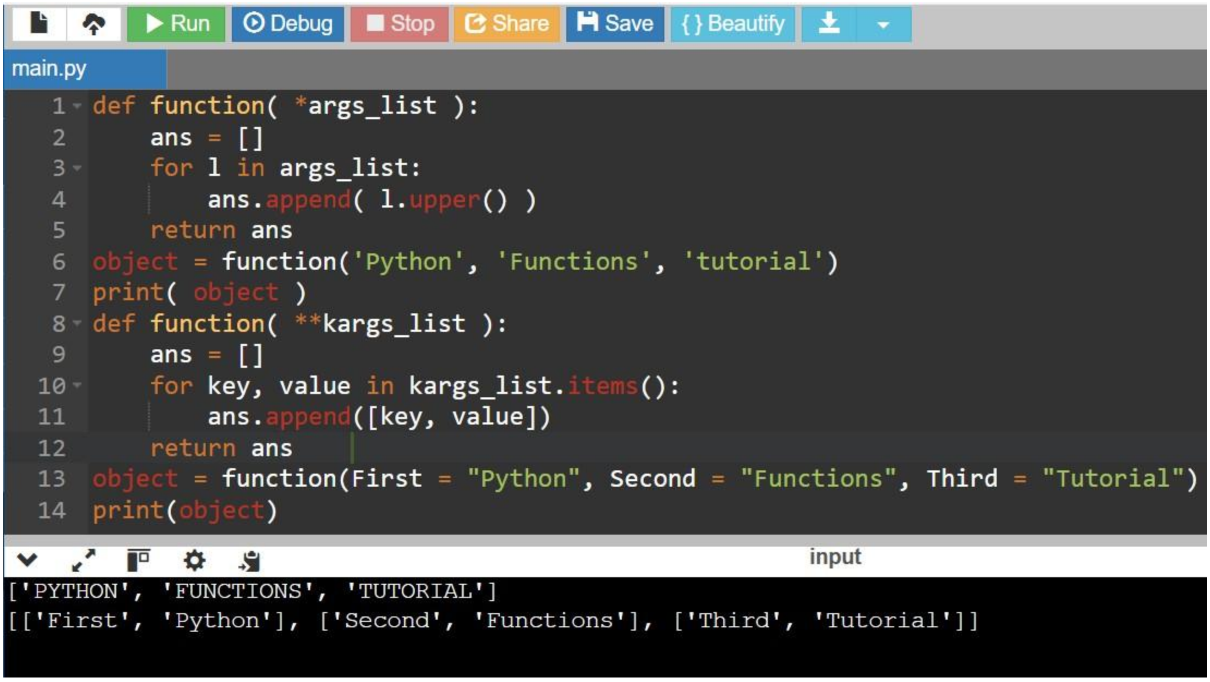
3.Required arguments



```python
def function( n1, n2 ):
    print("number 1 is: ", n1)
    print("number 2 is: ", n2)
print( "Passing out of order arguments" )
function( 30, 20 )
print( "Passing only one argument" )
try:
    function( 30 )
except:
    print( "Function needs two positional arguments" )
```

```
Passing out of order arguments
number 1 is:  30
number 2 is:  20
Passing only one argument
Function needs two positional arguments
```

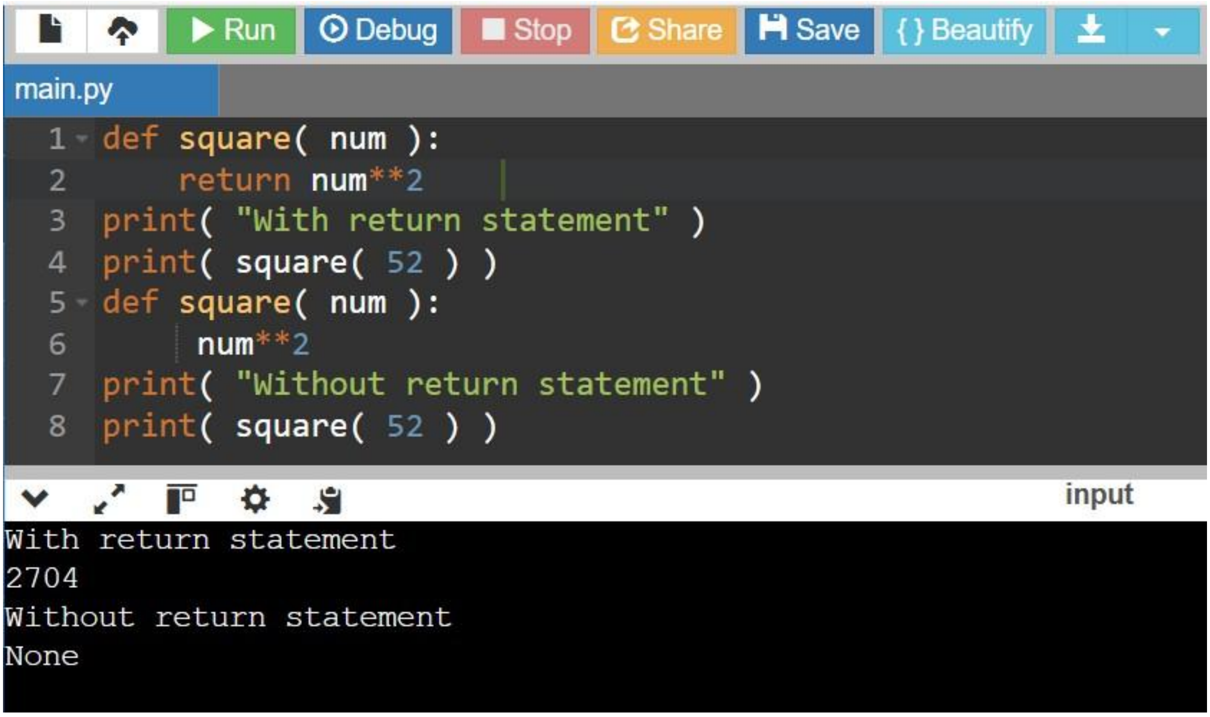
4.Variable-length arguments



```python
def function( *args_list ):
    ans = []
    for l in args_list:
        ans.append( l.upper() )
    return ans
object = function('Python', 'Functions', 'tutorial')
print( object )
def function( **kargs_list ):
    ans = []
    for key, value in kargs_list.items():
        ans.append([key, value])
    return ans
object = function(First = "Python", Second = "Functions", Third = "Tutorial")
print(object)
```

```
['PYTHON', 'FUNCTIONS', 'TUTORIAL']
[['First', 'Python'], ['Second', 'Functions'], ['Third', 'Tutorial']]
```

RETURN STATEMENT

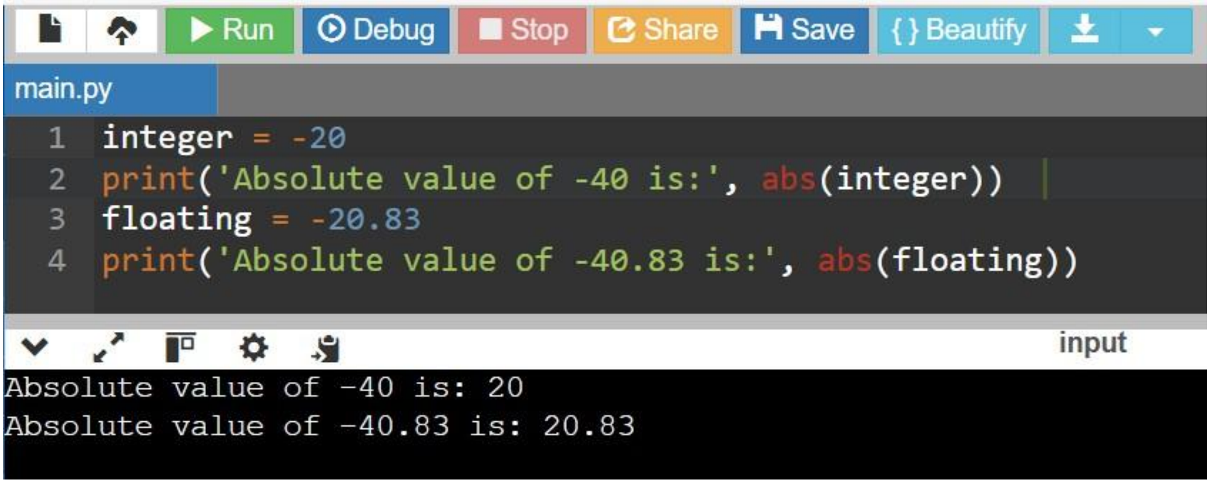


```python
def square( num ):
    return num**2
print( "With return statement" )
print( square( 52 ) )
def square( num ):
     num**2
print( "Without return statement" )
print( square( 52 ) )
```

```
With return statement
2704
Without return statement
None
```

PYTHON BUILT-IN FUNCTIONS

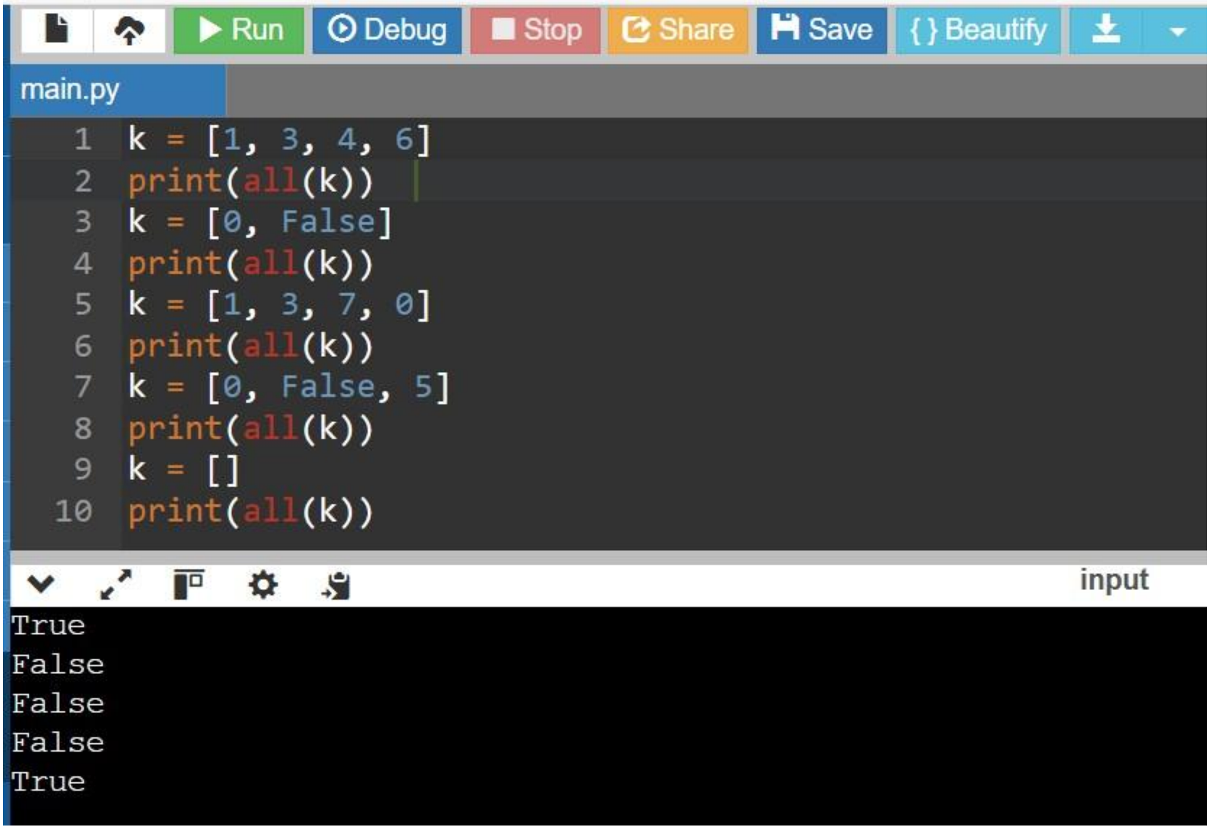
1. **Abs () function**



```python
integer = -20
print('Absolute value of -40 is:', abs(integer))
floating = -20.83
print('Absolute value of -40.83 is:', abs(floating))
```

```
Absolute value of -40 is: 20
Absolute value of -40.83 is: 20.83
```

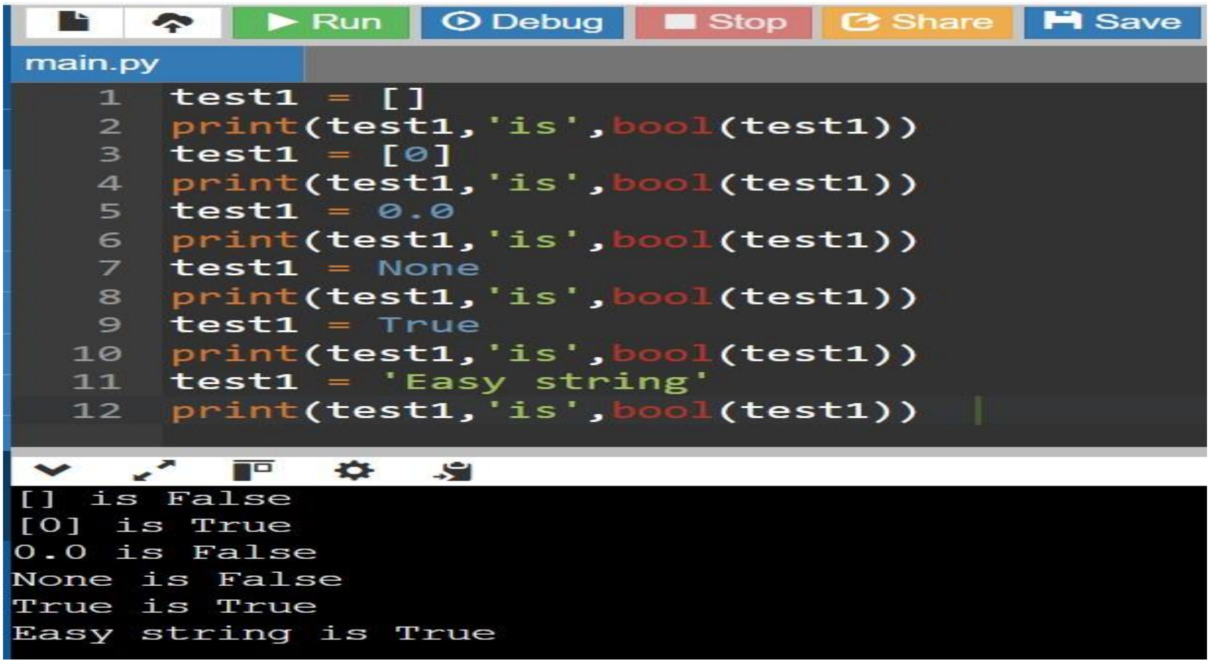
2. All () function



```
k = [1, 3, 4, 6]
print(all(k))
k = [0, False]
print(all(k))
k = [1, 3, 7, 0]
print(all(k))
k = [0, False, 5]
print(all(k))
k = []
print(all(k))
```

```
True
False
False
False
True
```

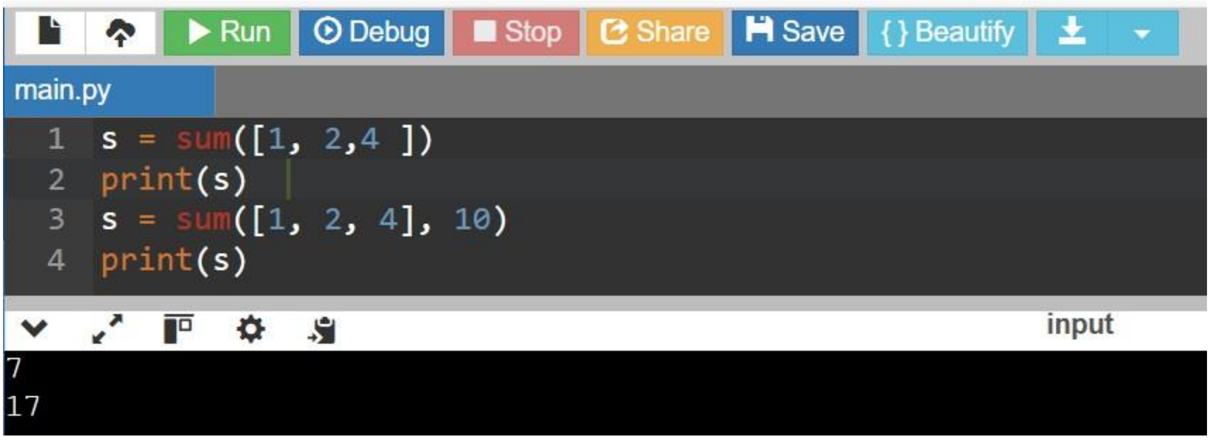
3.Bool () function



```
test1 = []
print(test1,'is',bool(test1))
test1 = [0]
print(test1,'is',bool(test1))
test1 = 0.0
print(test1,'is',bool(test1))
test1 = None
print(test1,'is',bool(test1))
test1 = True
print(test1,'is',bool(test1))
test1 = 'Easy string'
print(test1,'is',bool(test1))
```

```
[] is False
[0] is True
0.0 is False
None is False
True is True
Easy string is True
```

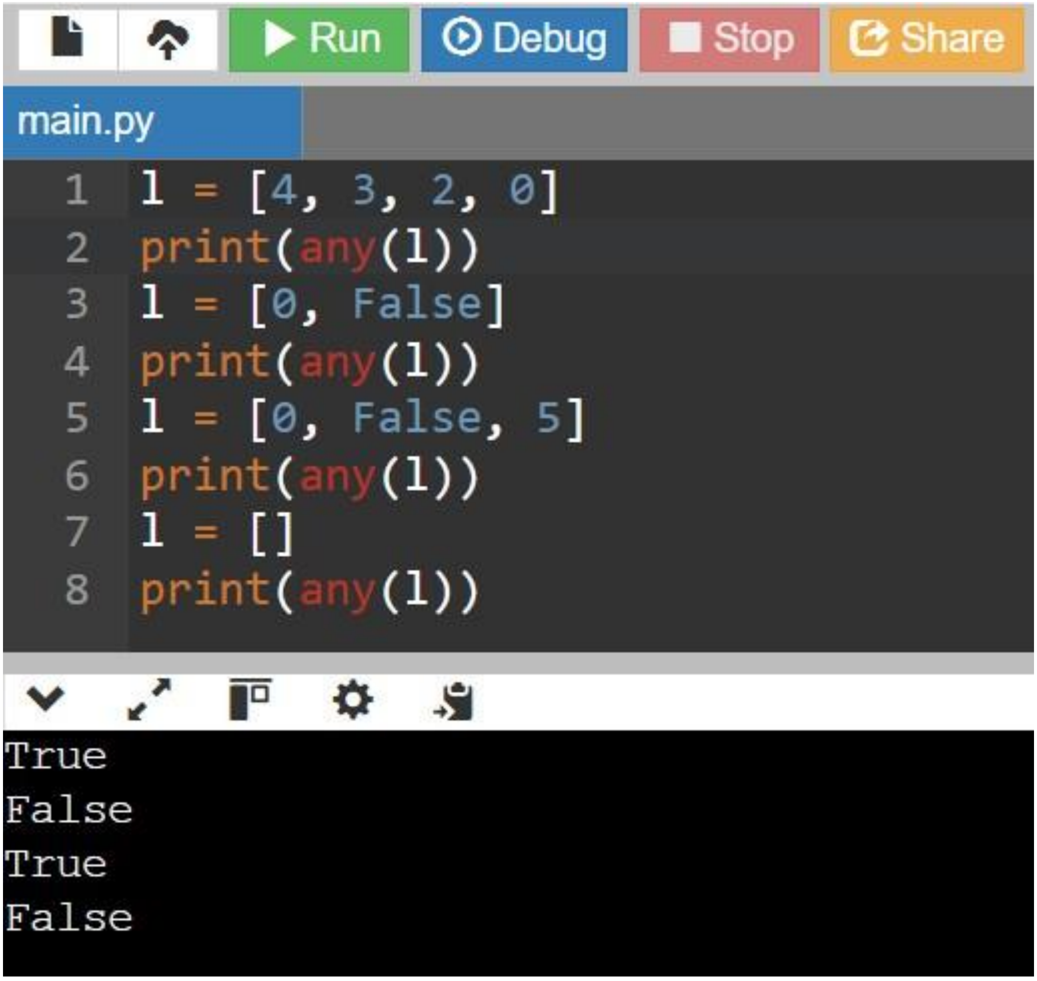
4.Sum () Function



```python
s = sum([1, 2,4 ])
print(s)
s = sum([1, 2, 4], 10)
print(s)
```

```
7
17
```

5.Any () function

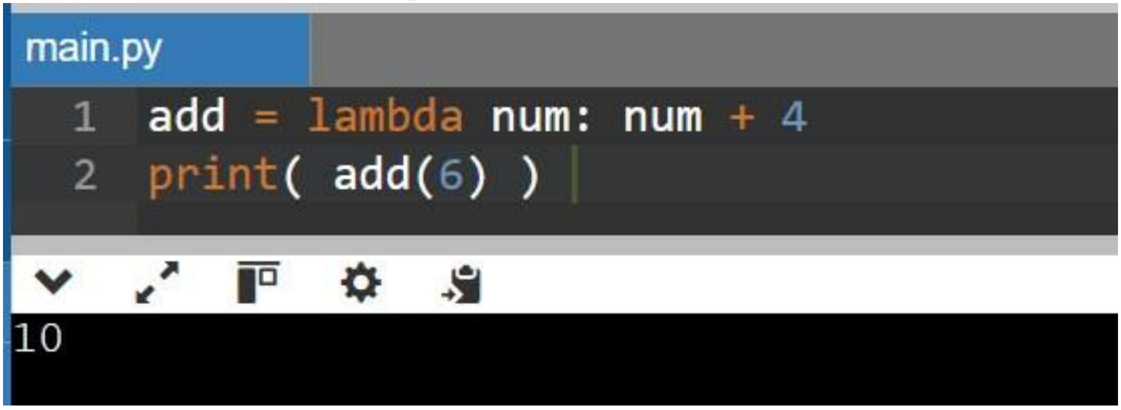


```python
l = [4, 3, 2, 0]
print(any(l))
l = [0, False]
print(any(l))
l = [0, False, 5]
print(any(l))
l = []
print(any(l))
```

```
True
False
True
False
```

PYTHON LAMBDA FUNCTION

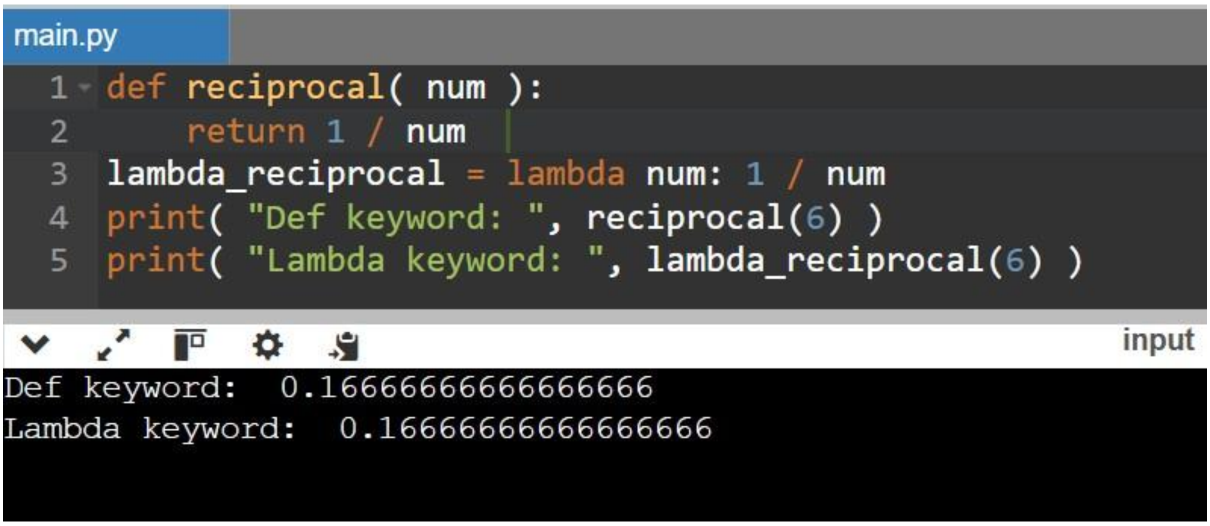
1. **Lambda function example**



```python
add = lambda num: num + 4
print( add(6) )
```

```
10
```

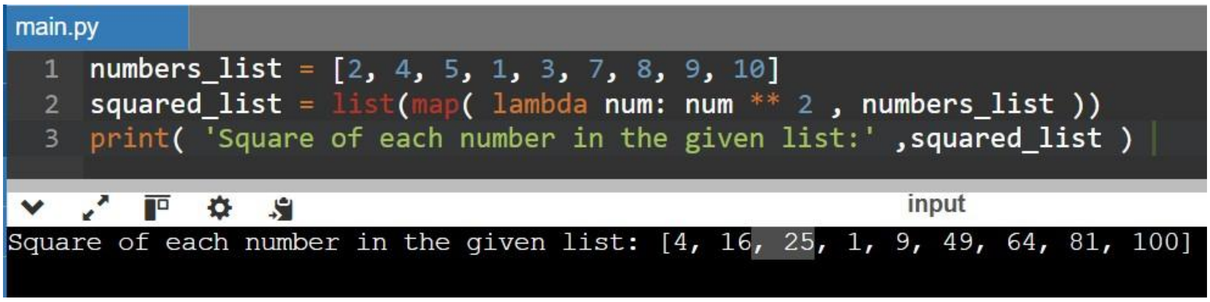
2. **Distinction between Lambda and Def Function**



```python
def reciprocal( num ):
    return 1 / num
lambda_reciprocal = lambda num: 1 / num
print( "Def keyword: ", reciprocal(6) )
print( "Lambda keyword: ", lambda_reciprocal(6) )
```

```
Def keyword:  0.16666666666666666
Lambda keyword:  0.16666666666666666
```

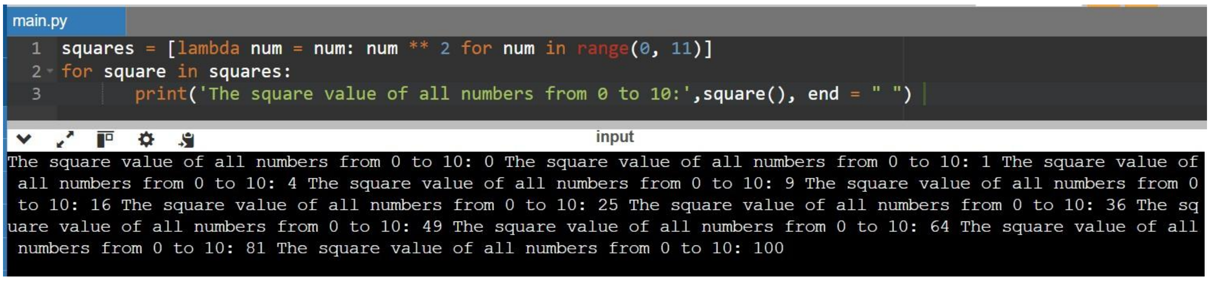
3. **Using Lambda Function with map ()**



```python
numbers_list = [2, 4, 5, 1, 3, 7, 8, 9, 10]
squared_list = list(map( lambda num: num ** 2 , numbers_list ))
print( 'Square of each number in the given list:' ,squared_list )
```

```
Square of each number in the given list: [4, 16, 25, 1, 9, 49, 64, 81, 100]
```

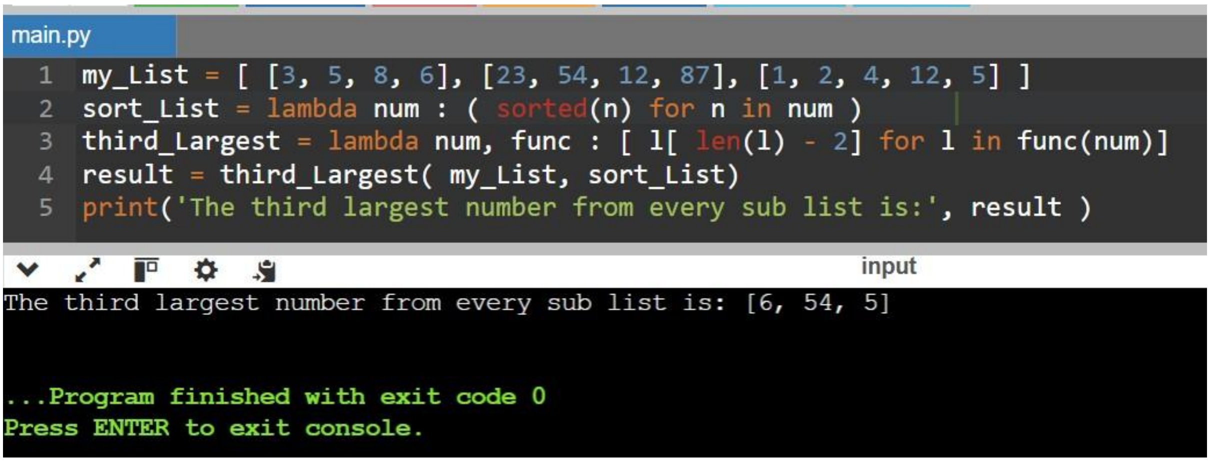
4. Using Lambda Function with List



```python
squares = [lambda num = num: num ** 2 for num in range(0, 11)]
for square in squares:
    print('The square value of all numbers from 0 to 10:',square(), end = " ")
```

```
The square value of all numbers from 0 to 10: 0 The square value of all numbers from 0 to 10: 1 The square value of
 all numbers from 0 to 10: 4 The square value of all numbers from 0 to 10: 9 The square value of all numbers from 0
 to 10: 16 The square value of all numbers from 0 to 10: 25 The square value of all numbers from 0 to 10: 36 The sq
uare value of all numbers from 0 to 10: 49 The square value of all numbers from 0 to 10: 64 The square value of all
 numbers from 0 to 10: 81 The square value of all numbers from 0 to 10: 100
```

5.Using Lambda Function with Multiple Statements

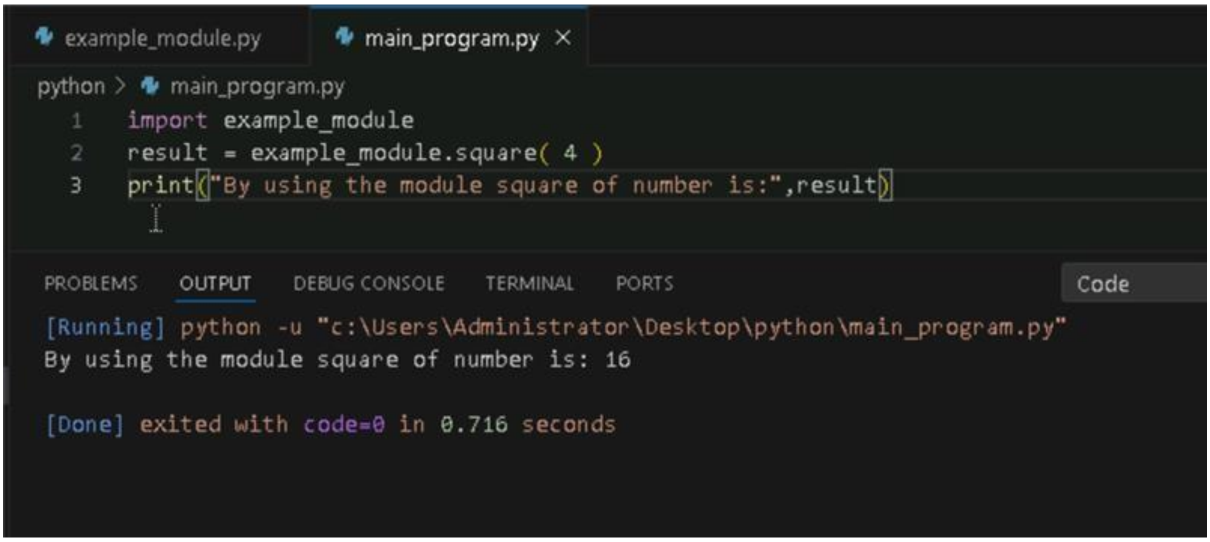


```python
my_List = [ [3, 5, 8, 6], [23, 54, 12, 87], [1, 2, 4, 12, 5] ]
sort_List = lambda num : ( sorted(n) for n in num )
third_Largest = lambda num, func : [ l[ len(l) - 2] for l in func(num)]
result = third_Largest( my_List, sort_List)
print('The third largest number from every sub list is:', result )
```

```
The third largest number from every sub list is: [6, 54, 5]


...Program finished with exit code 0
Press ENTER to exit console.
```

MODULES

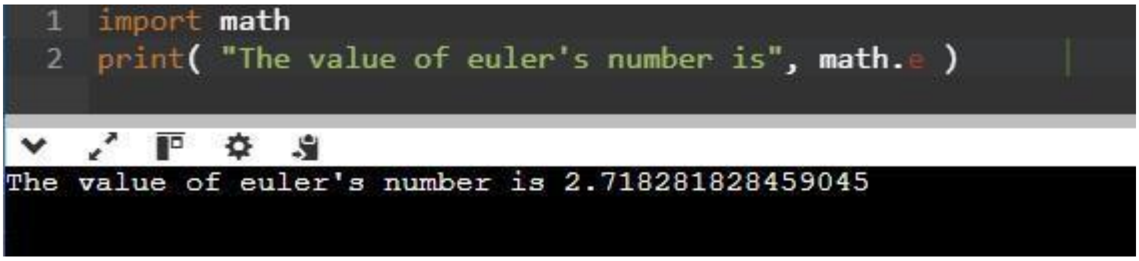
1.Python Modules



```
import example_module
result = example_module.square( 4 )
print("By using the module square of number is:",result)
```

```
[Running] python -u "c:\Users\Administrator\Desktop\python\main_program.py"
By using the module square of number is: 16

[Done] exited with code=0 in 0.716 seconds
```

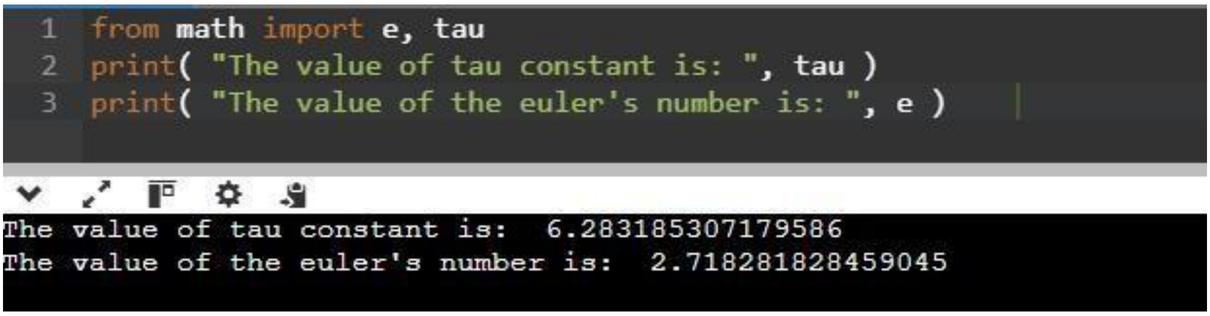
2.Importing and also Renaming



```
import math
print( "The value of euler's number is", math.e )
```

```
The value of euler's number is 2.718281828459045
```

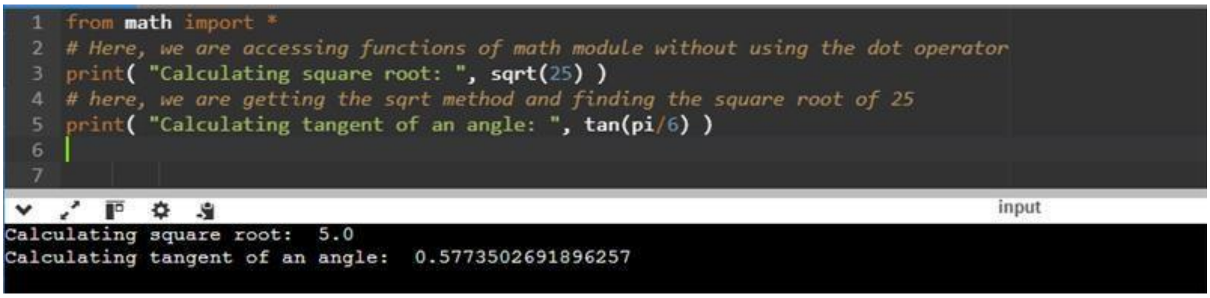
3.Python from...import Statement



```
from math import e, tau
print( "The value of tau constant is: ", tau )
print( "The value of the euler's number is: ", e )
```

```
The value of tau constant is:   6.283185307179586
The value of the euler's number is:   2.718281828459045
```

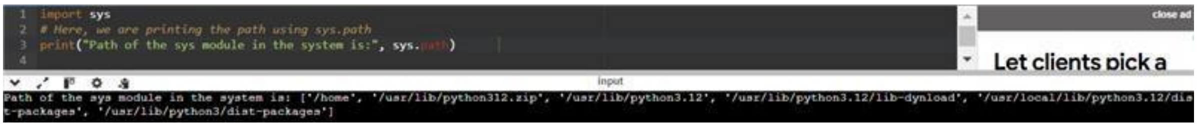
4.Import all Names - From import * Statement



```
from math import *
# Here, we are accessing functions of math module without using the dot operator
print( "Calculating square root: ", sqrt(25) )
# here, we are getting the sqrt method and finding the square root of 25
print( "Calculating tangent of an angle: ", tan(pi/6) )
```

```
Calculating square root:  5.0
Calculating tangent of an angle:  0.5773502691896257
```

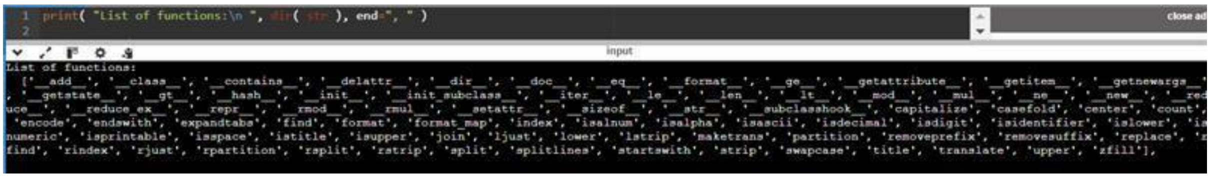
5.Locating Path of Modules



```
import sys
# Here, we are printing the path using sys.path
print("Path of the sys module in the system is:", sys.path)
```

```
Path of the sys module in the system is: ['/home', '/usr/lib/python312.zip', '/usr/lib/python3.12', '/usr/lib/python3.12/lib-dynload', '/usr/local/lib/python3.12/dis
t-packages', '/usr/lib/python3/dist-packages']
```

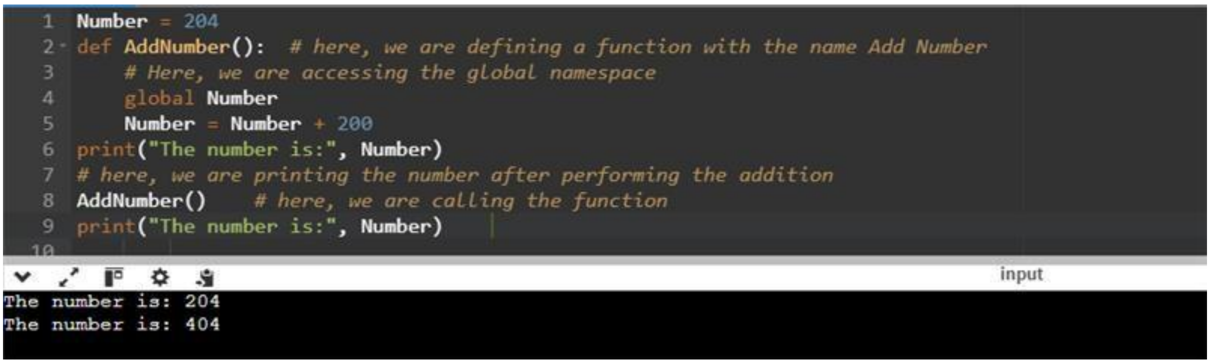
6.The dir() Built-in Function



```
print( "List of functions:\n ", dir( str ), end=", " )
```

```
List of functions:
  ['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__'
, '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__red
uce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count',
 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'is
numeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'r
find', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'],
```

7.Namespaces and Scoping

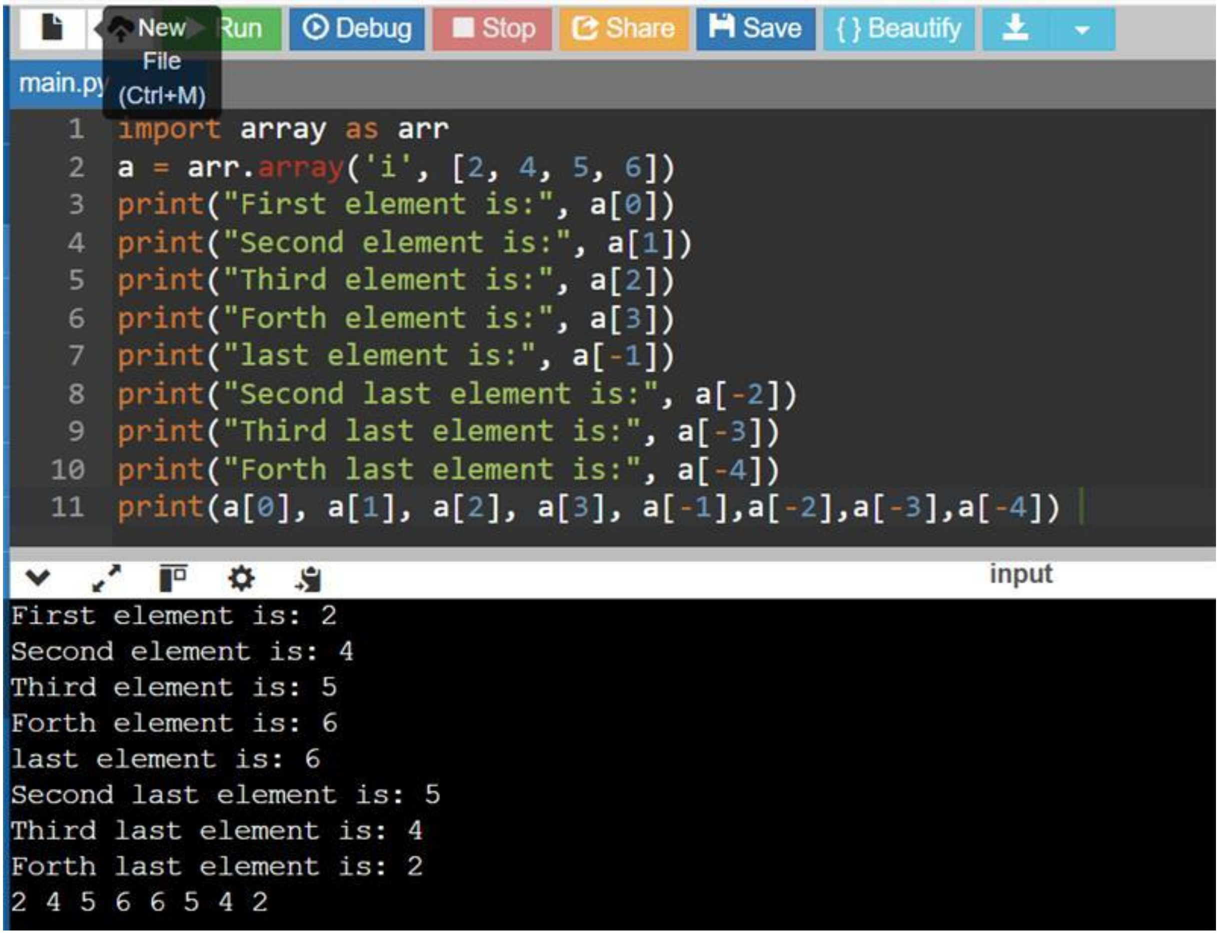


```
Number = 204
def AddNumber():  # here, we are defining a function with the name Add Number
    # Here, we are accessing the global namespace
    global Number
    Number = Number + 200
print("The number is:", Number)
# here, we are printing the number after performing the addition
AddNumber()    # here, we are calling the function
print("The number is:", Number)
```

```
The number is: 204
The number is: 404
```

PYTHON ARRAYS

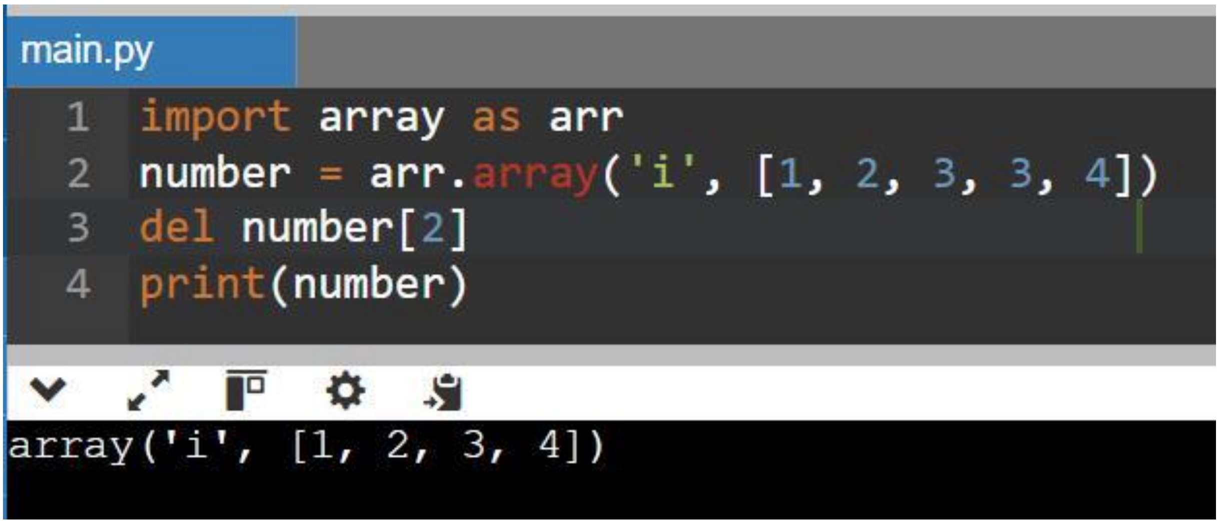
1. **Accessing array elements**



```
import array as arr
a = arr.array('i', [2, 4, 5, 6])
print("First element is:", a[0])
print("Second element is:", a[1])
print("Third element is:", a[2])
print("Forth element is:", a[3])
print("last element is:", a[-1])
print("Second last element is:", a[-2])
print("Third last element is:", a[-3])
print("Forth last element is:", a[-4])
print(a[0], a[1], a[2], a[3], a[-1],a[-2],a[-3],a[-4])
```

```
First element is: 2
Second element is: 4
Third element is: 5
Forth element is: 6
last element is: 6
Second last element is: 5
Third last element is: 4
Forth last element is: 2
2 4 5 6 6 5 4 2
```

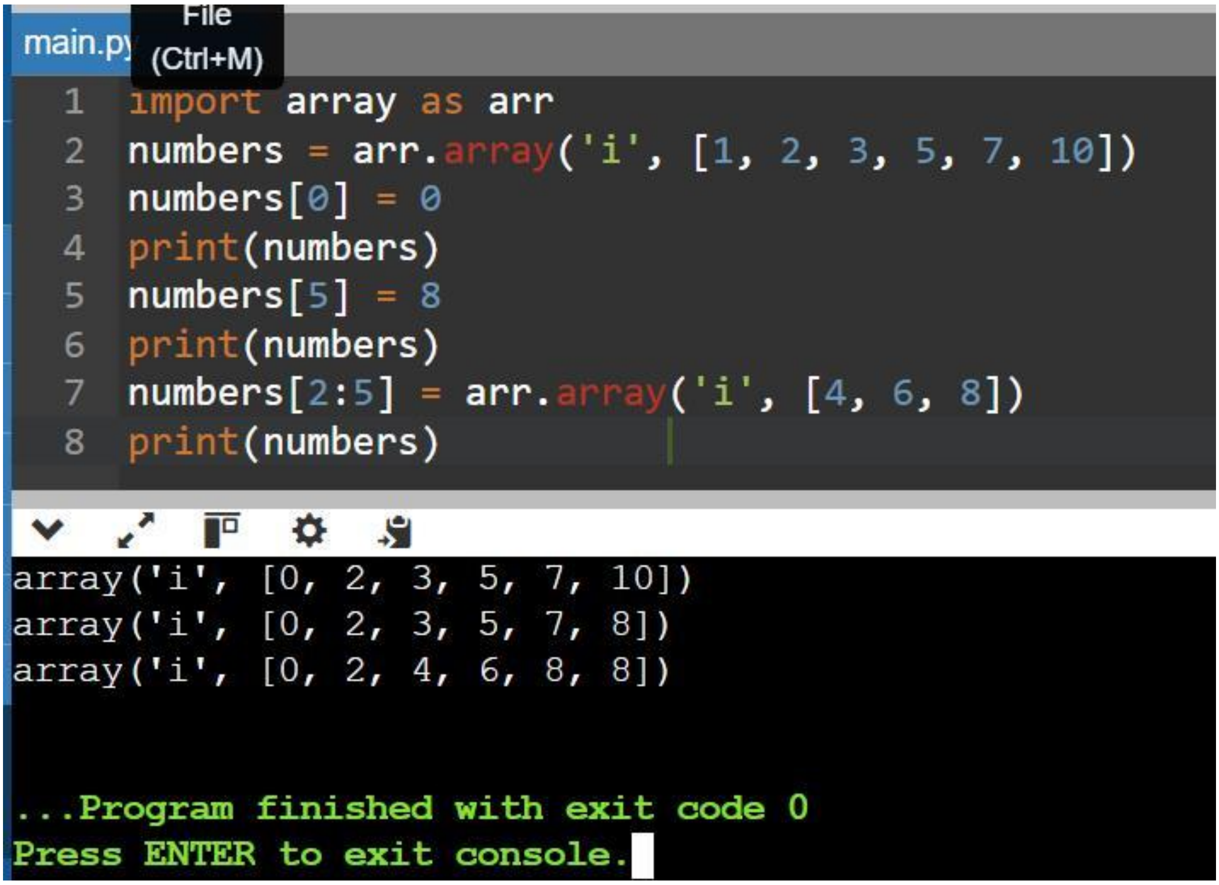
2. **Deleting the elements from Array**



```
import array as arr
number = arr.array('i', [1, 2, 3, 3, 4])
del number[2]
print(number)
```

```
array('i', [1, 2, 3, 4])
```

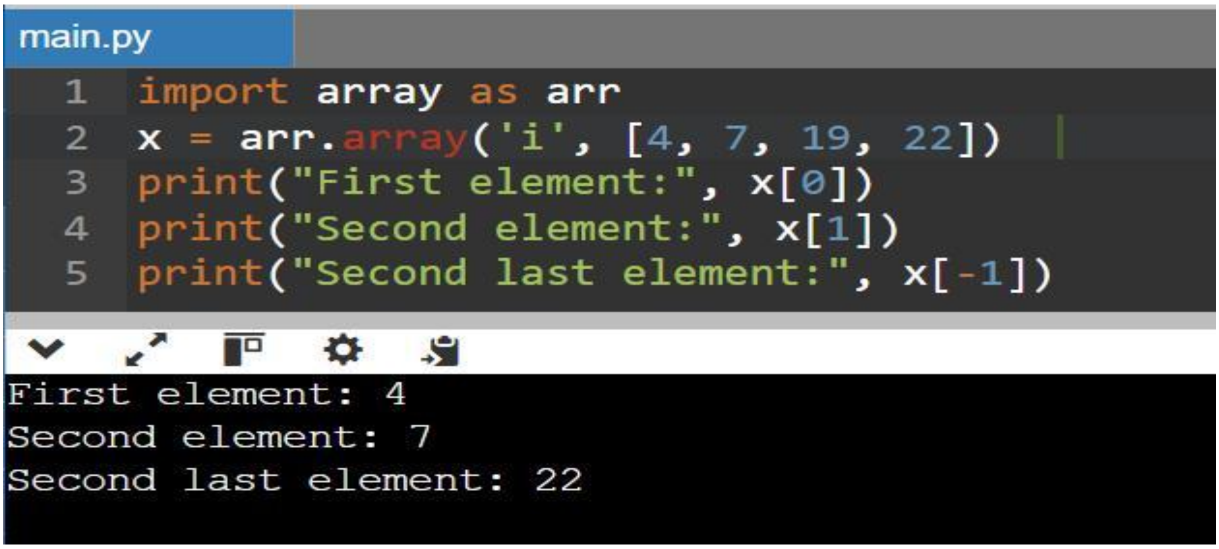
3.Adding or changing the elements in Array



```python
import array as arr
numbers = arr.array('i', [1, 2, 3, 5, 7, 10])
numbers[0] = 0
print(numbers)
numbers[5] = 8
print(numbers)
numbers[2:5] = arr.array('i', [4, 6, 8])
print(numbers)
```

```
array('i', [0, 2, 3, 5, 7, 10])
array('i', [0, 2, 3, 5, 7, 8])
array('i', [0, 2, 4, 6, 8, 8])


...Program finished with exit code 0
Press ENTER to exit console.
```

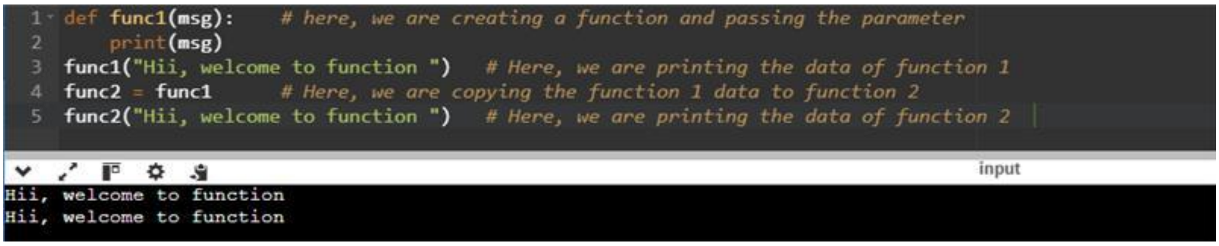
4.To find the length of array



```python
import array as arr
x = arr.array('i', [4, 7, 19, 22])
print("First element:", x[0])
print("Second element:", x[1])
print("Second last element:", x[-1])
```

```
First element: 4
Second element: 7
Second last element: 22
```

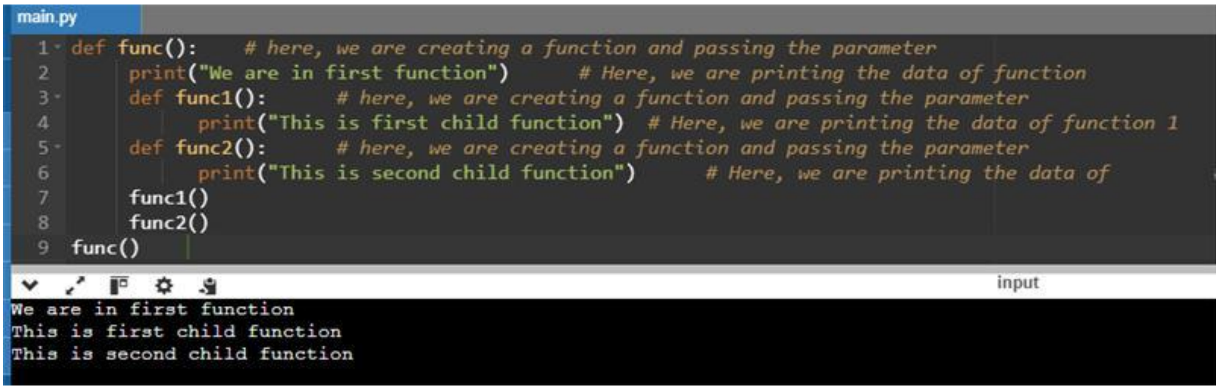
PYTHON DECORATOR



```python
def func1(msg):     # here, we are creating a function and passing the parameter
    print(msg)
func1("Hii, welcome to function ")   # Here, we are printing the data of function 1
func2 = func1      # Here, we are copying the function 1 data to function 2
func2("Hii, welcome to function ")   # Here, we are printing the data of function 2
```

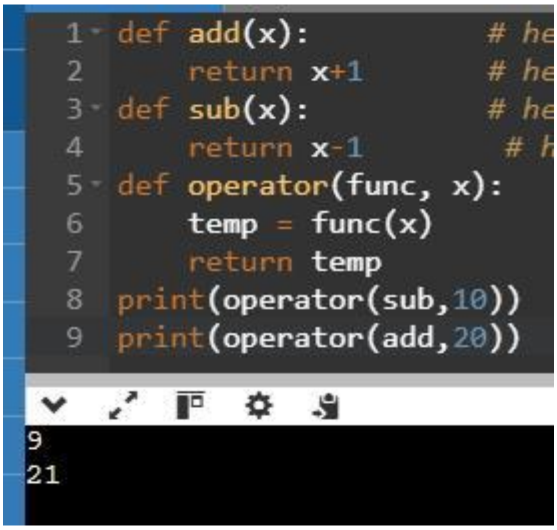
```
Hii, welcome to function
Hii, welcome to function
```

1.Inner Function



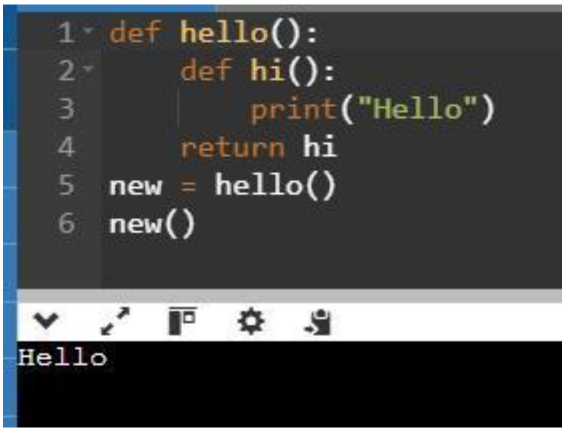
```python
def func():    # here, we are creating a function and passing the parameter
    print("We are in first function")      # Here, we are printing the data of function
    def func1():      # here, we are creating a function and passing the parameter
        print("This is first child function")  # Here, we are printing the data of function 1
    def func2():      # here, we are creating a function and passing the parameter
        print("This is second child function")      # Here, we are printing the data of
    func1()
    func2()
func()
```

```
We are in first function
This is first child function
This is second child function
```



```python
def add(x):              # he
    return x+1           # he
def sub(x):              # he
    return x-1            # h
def operator(func, x):
    temp = func(x)
    return temp
print(operator(sub,10))
print(operator(add,20))
```

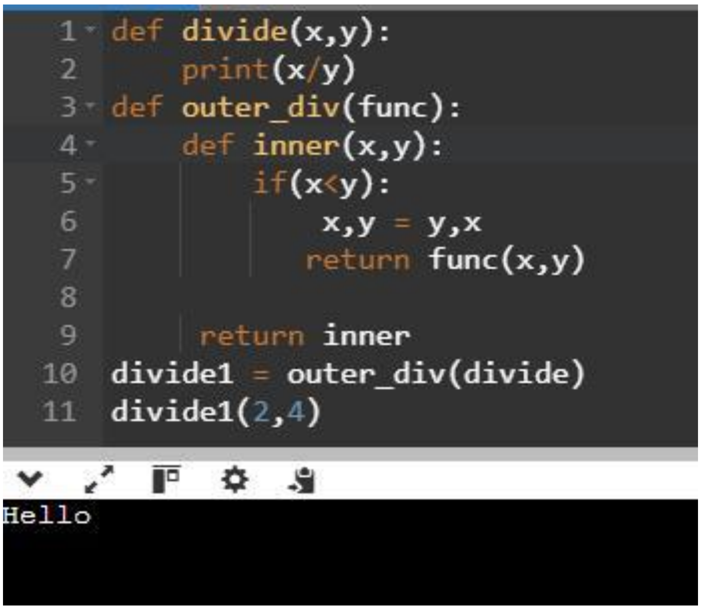
```
9
21
```



```
def hello():
    def hi():
        print("Hello")
    return hi
new = hello()
new()
```

```
Hello
```

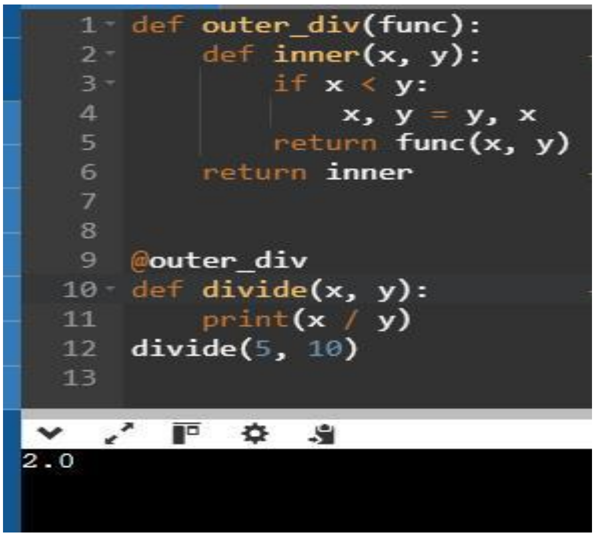
2.Decorating functions with parameters



```
def divide(x,y):
    print(x/y)
def outer_div(func):
    def inner(x,y):
        if(x<y):
            x,y = y,x
        return func(x,y)

     return inner
divide1 = outer_div(divide)
divide1(2,4)
```

```
Hello
```

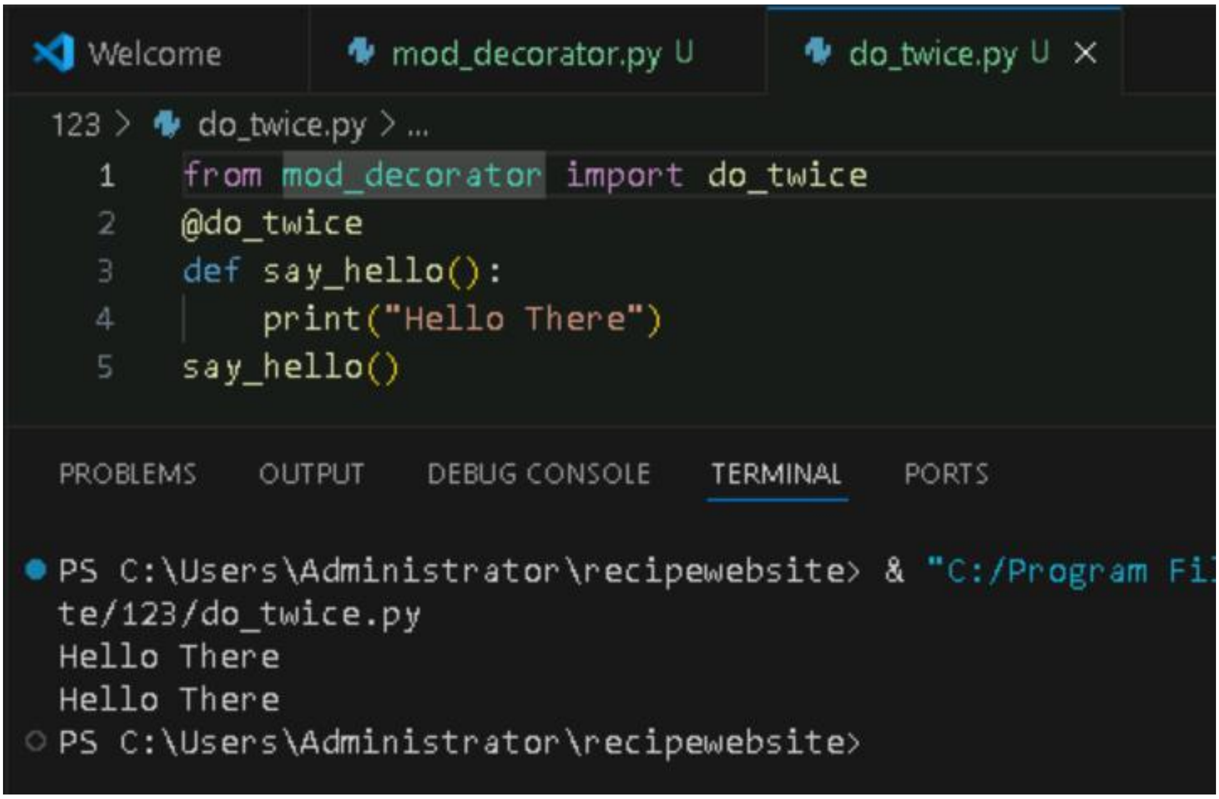
3.Syntactic Decorator



```python
def outer_div(func):
    def inner(x, y):
        if x < y:
            x, y = y, x
        return func(x, y)
    return inner


@outer_div
def divide(x, y):
    print(x / y)
divide(5, 10)
```

```
2.0
```

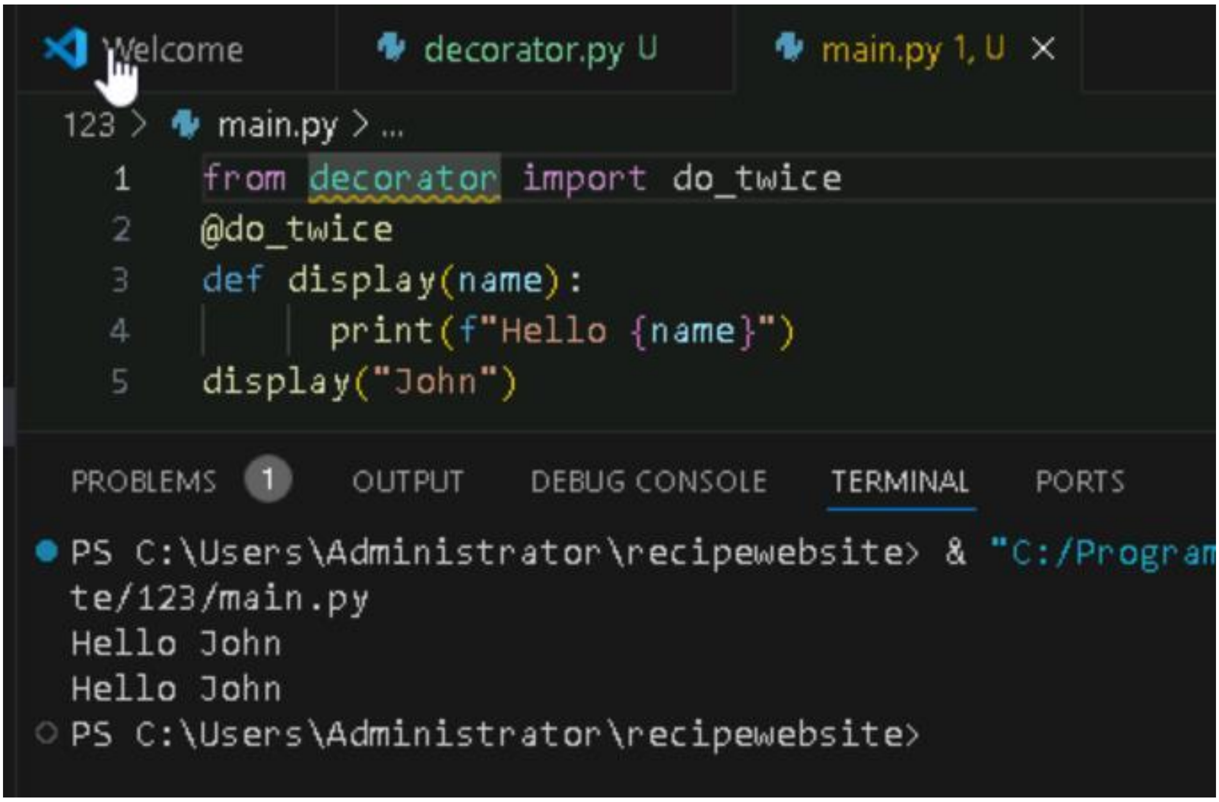
4.Reusing Decorator



```python
from mod_decorator import do_twice
@do_twice
def say_hello():
    print("Hello There")
say_hello()
```

```
PS C:\Users\Administrator\recipewebsite> & "C:/Program Fi
te/123/do_twice.py
Hello There
Hello There
PS C:\Users\Administrator\recipewebsite>
```

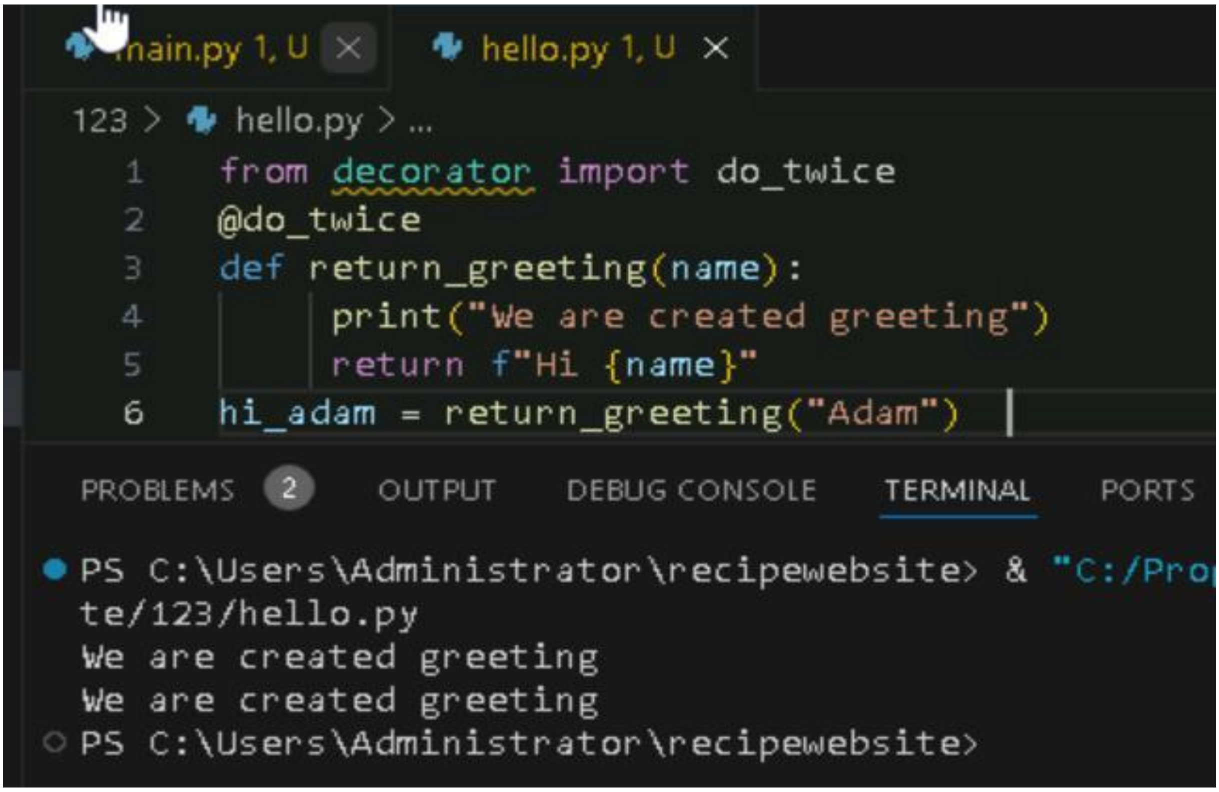
5.Python Decorator with Argument



```python
from decorator import do_twice
@do_twice
def display(name):
        print(f"Hello {name}")
display("John")
```

```
PS C:\Users\Administrator\recipewebsite> & "C:/Progra
te/123/main.py
Hello John
Hello John
PS C:\Users\Administrator\recipewebsite>
```

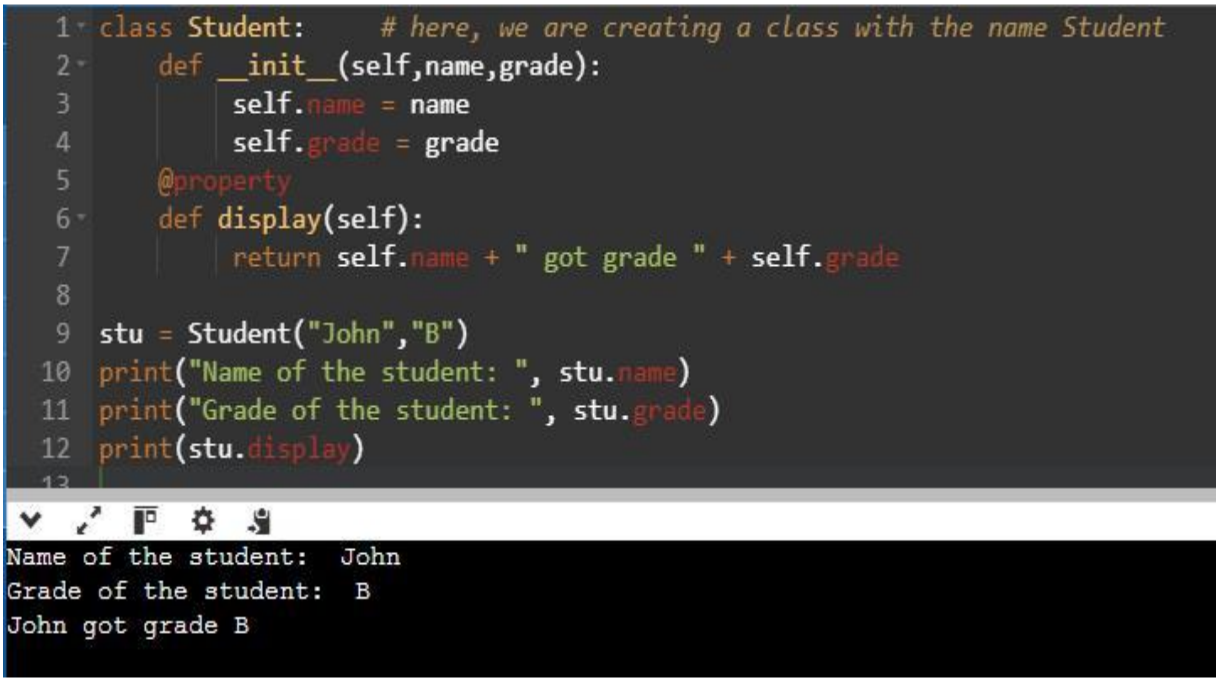
6.Returning Values from Decorated Functions



```python
from decorator import do_twice
@do_twice
def return_greeting(name):
        print("We are created greeting")
        return f"Hi {name}"
hi_adam = return_greeting("Adam")
```

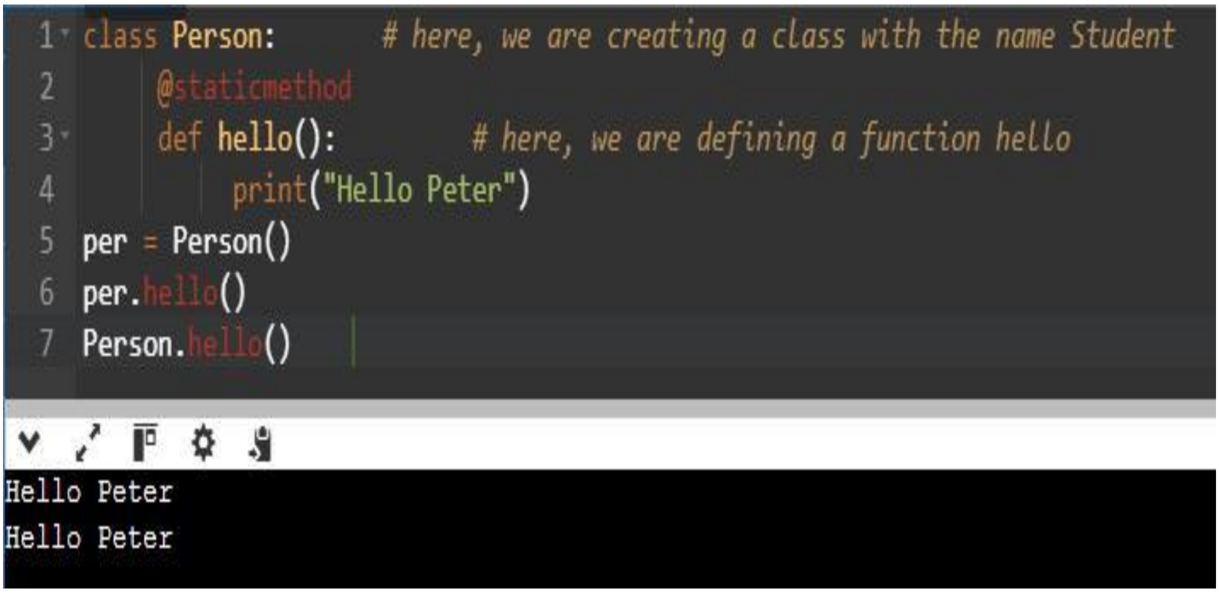
```
PS C:\Users\Administrator\recipewebsite> & "C:/Pro
te/123/hello.py
We are created greeting
We are created greeting
PS C:\Users\Administrator\recipewebsite>
```

7.Fancy Decorators



```python
class Student:     # here, we are creating a class with the name Student
    def __init__(self,name,grade):
        self.name = name
        self.grade = grade
    @property
    def display(self):
        return self.name + " got grade " + self.grade

stu = Student("John","B")
print("Name of the student: ", stu.name)
print("Grade of the student: ", stu.grade)
print(stu.display)
```

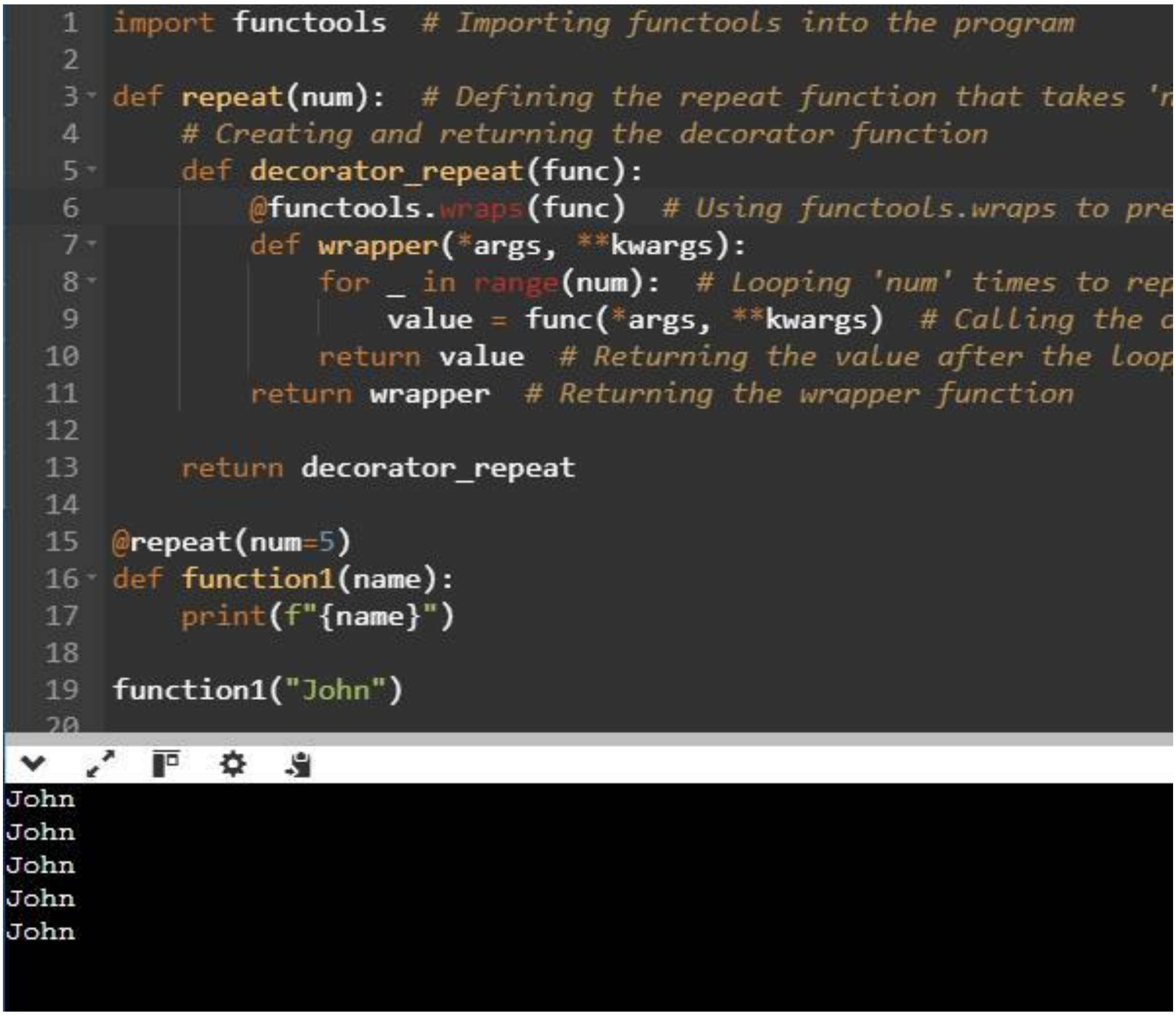
```
Name of the student:  John
Grade of the student:  B
John got grade B
```



```python
class Person:      # here, we are creating a class with the name Student
    @staticmethod
    def hello():        # here, we are defining a function hello
        print("Hello Peter")
per = Person()
per.hello()
Person.hello()
```

```
Hello Peter
Hello Peter
```

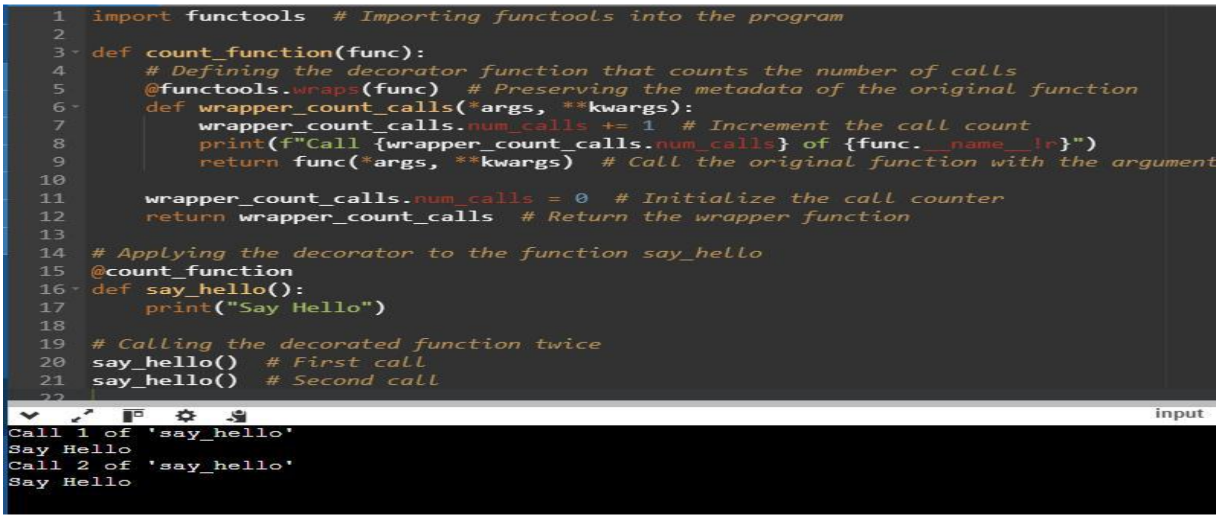
8.Decorator with Arguments



```python
import functools  # Importing functools into the program

def repeat(num):  # Defining the repeat function that takes 'n
    # Creating and returning the decorator function
    def decorator_repeat(func):
        @functools.wraps(func)  # Using functools.wraps to pre
        def wrapper(*args, **kwargs):
            for _ in range(num):  # Looping 'num' times to rep
                value = func(*args, **kwargs)  # Calling the c
            return value  # Returning the value after the loop
        return wrapper  # Returning the wrapper function

    return decorator_repeat

@repeat(num=5)
def function1(name):
    print(f"{name}")

function1("John")
```

```
John
John
John
John
John
```

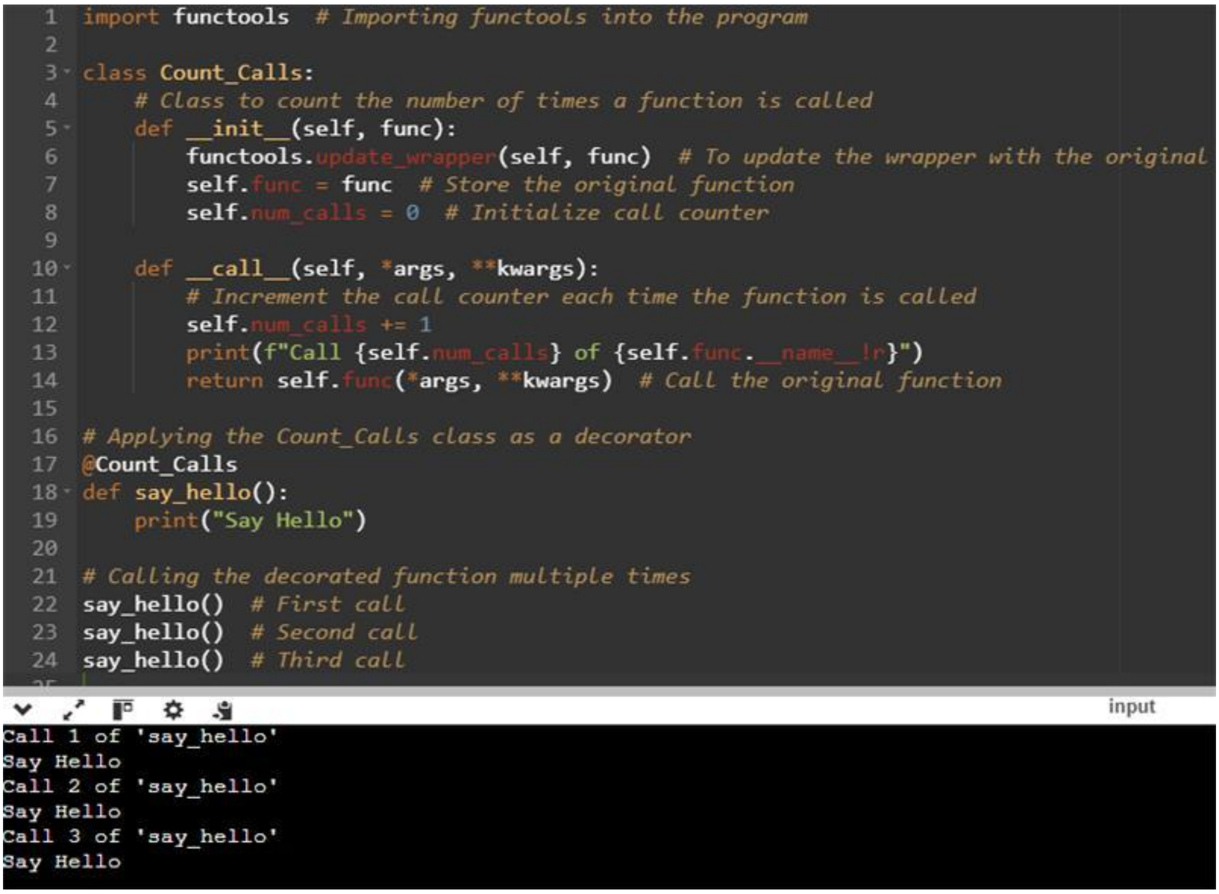
9.Stateful Decorators



```python
import functools  # Importing functools into the program

def count_function(func):
    # Defining the decorator function that counts the number of calls
    @functools.wraps(func)  # Preserving the metadata of the original function
    def wrapper_count_calls(*args, **kwargs):
        wrapper_count_calls.num_calls += 1  # Increment the call count
        print(f"Call {wrapper_count_calls.num_calls} of {func.__name__!r}")
        return func(*args, **kwargs)  # Call the original function with the argument

    wrapper_count_calls.num_calls = 0  # Initialize the call counter
    return wrapper_count_calls  # Return the wrapper function

# Applying the decorator to the function say_hello
@count_function
def say_hello():
    print("Say Hello")

# Calling the decorated function twice
say_hello()  # First call
say_hello()  # Second call
```

```
Call 1 of 'say_hello'
Say Hello
Call 2 of 'say_hello'
Say Hello
```

10.Classes as Decorators

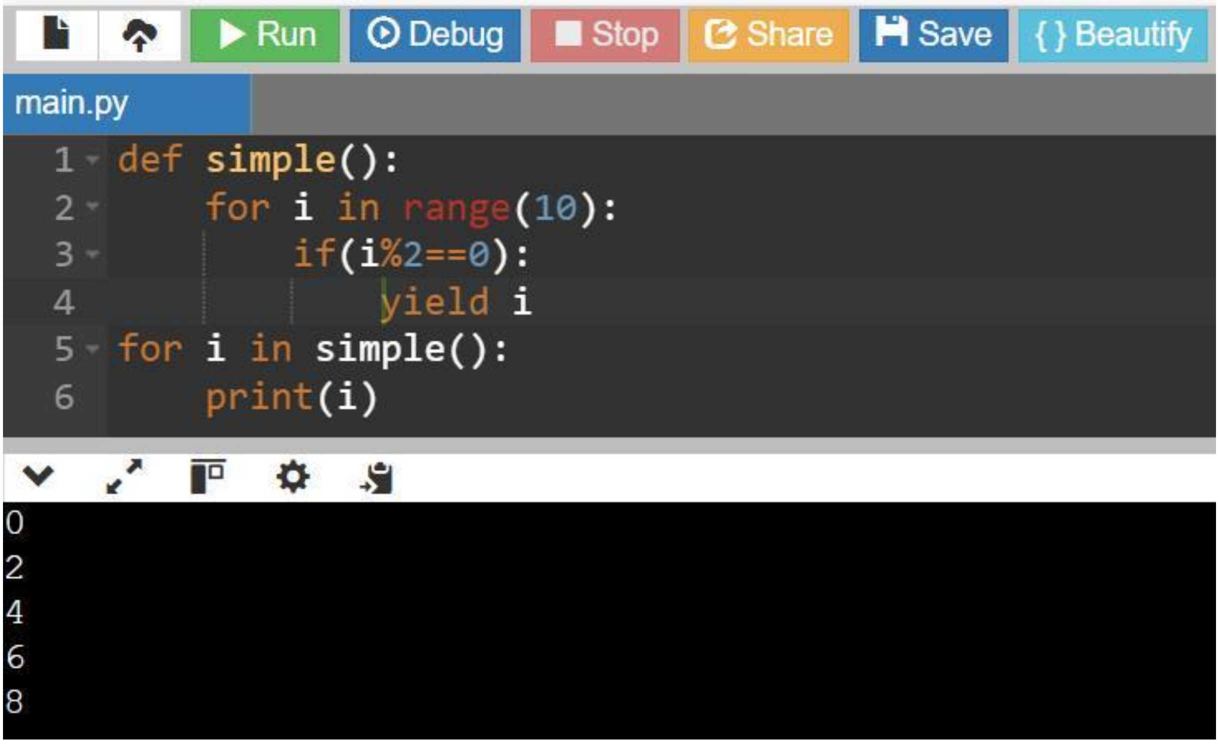


```python
import functools  # Importing functools into the program

class Count_Calls:
    # Class to count the number of times a function is called
    def __init__(self, func):
        functools.update_wrapper(self, func)  # To update the wrapper with the original
        self.func = func  # Store the original function
        self.num_calls = 0  # Initialize call counter

    def __call__(self, *args, **kwargs):
        # Increment the call counter each time the function is called
        self.num_calls += 1
        print(f"Call {self.num_calls} of {self.func.__name__!r}")
        return self.func(*args, **kwargs)  # Call the original function

# Applying the Count_Calls class as a decorator
@Count_Calls
def say_hello():
    print("Say Hello")

# Calling the decorated function multiple times
say_hello()  # First call
say_hello()  # Second call
say_hello()  # Third call
```

```
Call 1 of 'say_hello'
Say Hello
Call 2 of 'say_hello'
Say Hello
Call 3 of 'say_hello'
Say Hello
```

PYTHON GENERATORS

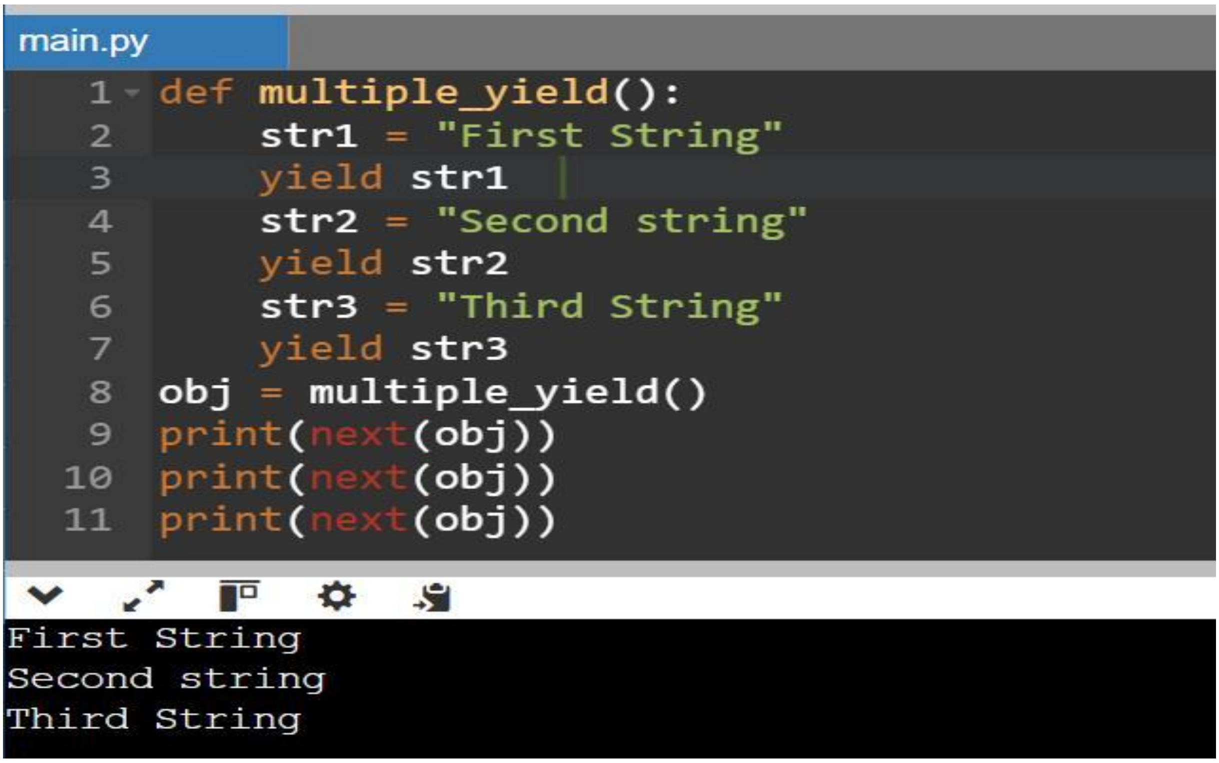
1. **To create Generator function in python**



```python
def simple():
    for i in range(10):
        if(i%2==0):
            yield i
for i in simple():
    print(i)
```

```
0
2
4
6
8
```

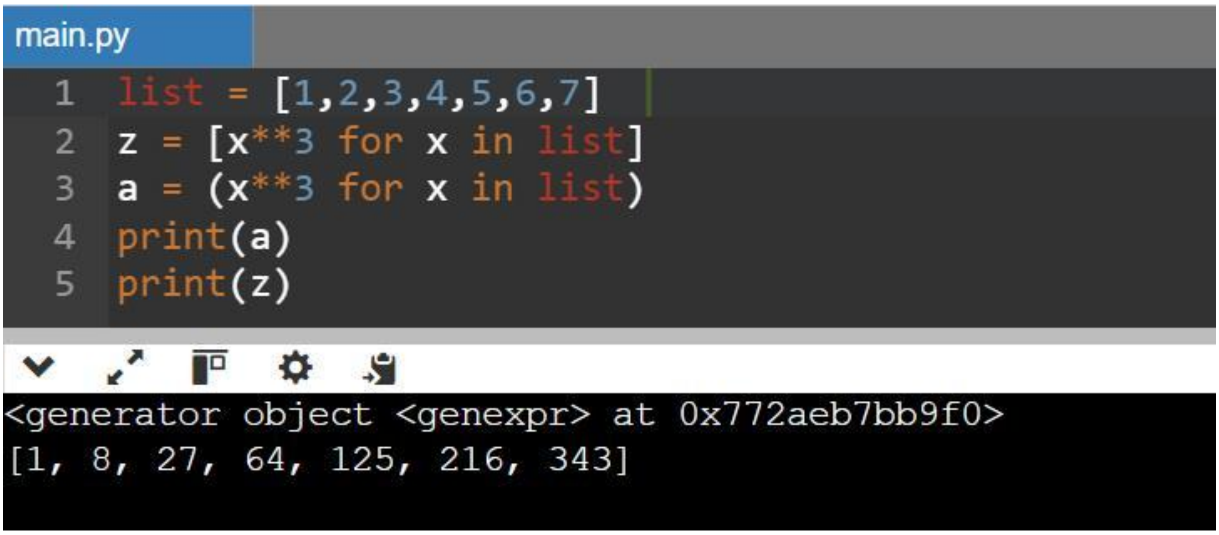
2.Using multiple Yield Statement



```python
def multiple_yield():
    str1 = "First String"
    yield str1
    str2 = "Second string"
    yield str2
    str3 = "Third String"
    yield str3
obj = multiple_yield()
print(next(obj))
print(next(obj))
print(next(obj))
```

```
First String
Second string
Third String
```

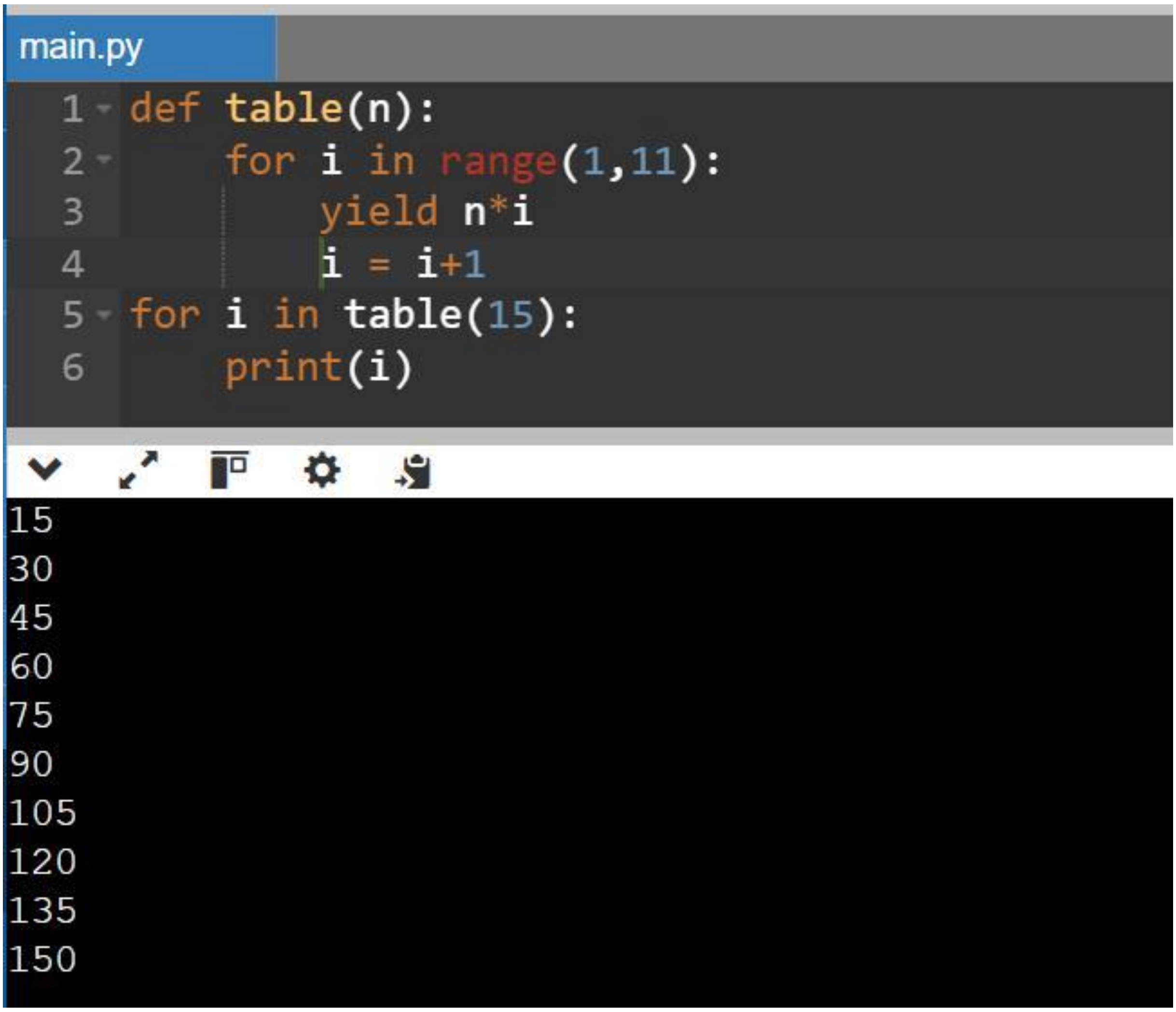
3.Generator Expression



```python
list = [1,2,3,4,5,6,7]
z = [x**3 for x in list]
a = (x**3 for x in list)
print(a)
print(z)
```

```
<generator object <genexpr> at 0x772aeb7bb9f0>
[1, 8, 27, 64, 125, 216, 343]
```

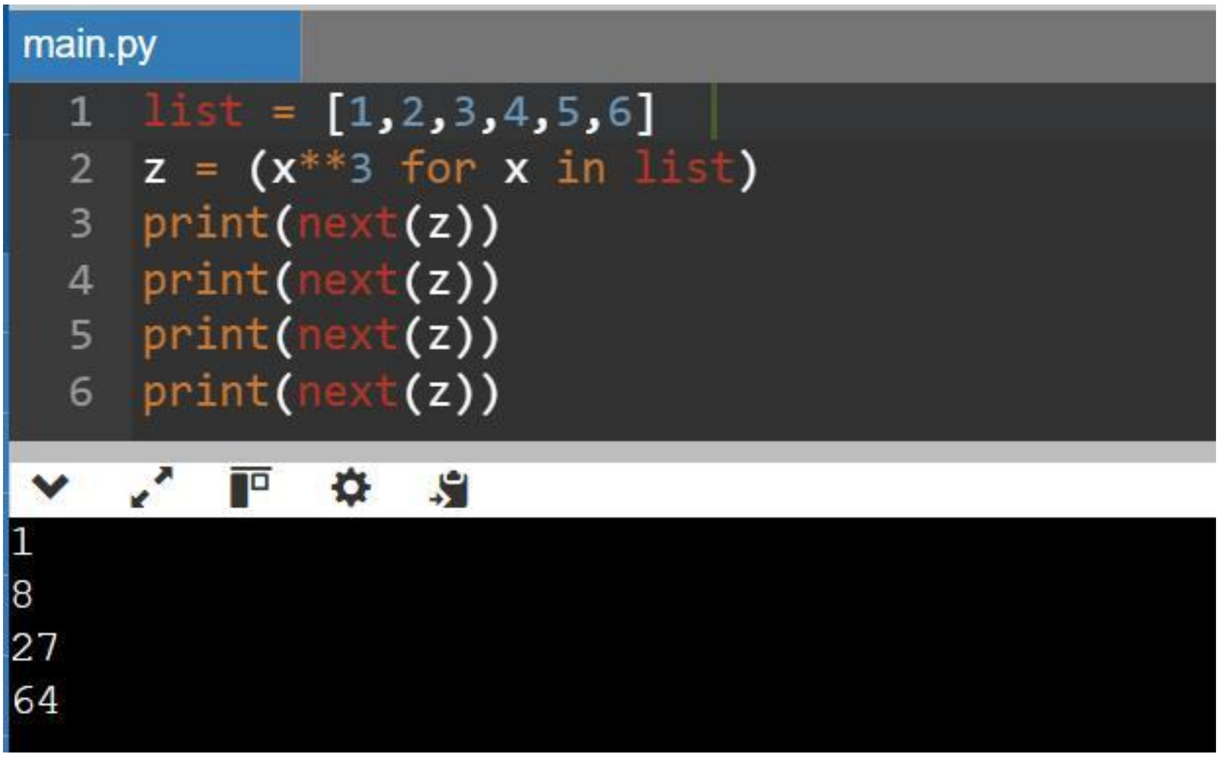
4.Multiplication table using Generators



```python
def table(n):
    for i in range(1,11):
        yield n*i
        i = i+1
for i in table(15):
    print(i)
```

```
15
30
45
60
75
90
105
120
135
150
```

5.Using next () on Generator Object

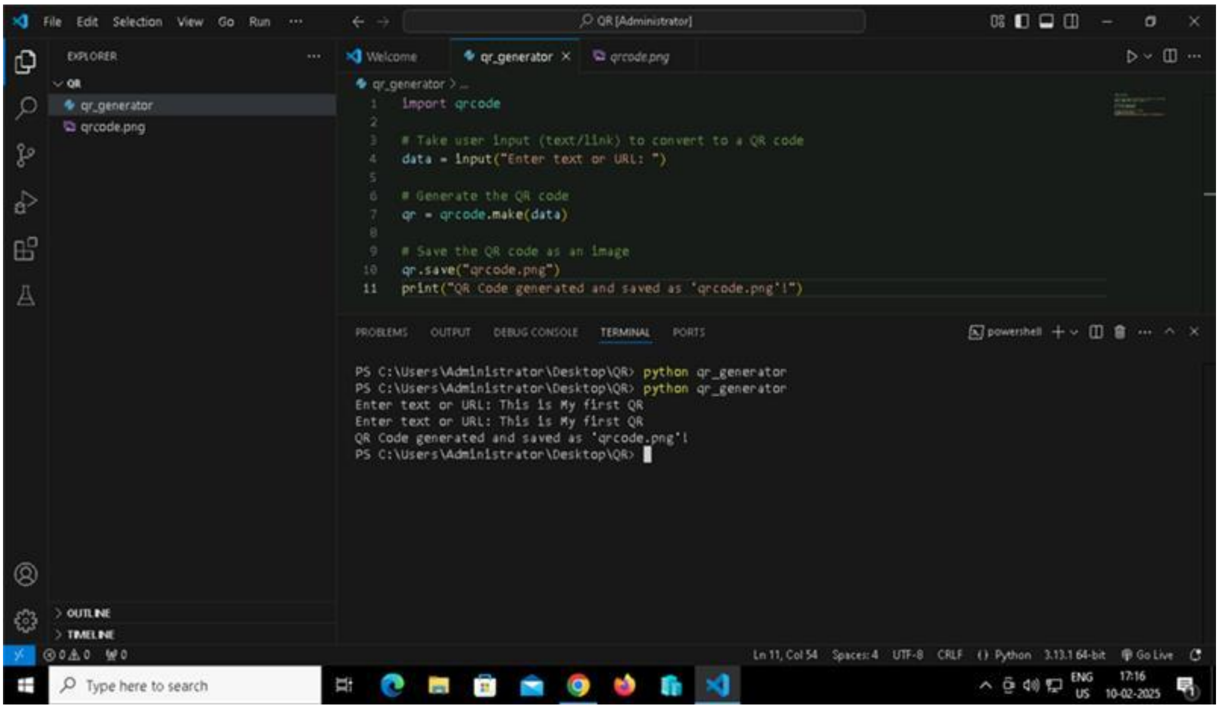


```python
list = [1,2,3,4,5,6]
z = (x**3 for x in list)
print(next(z))
print(next(z))
print(next(z))
print(next(z))
```

```
1
8
27
64
```

PYTHON BASIC PROJECT

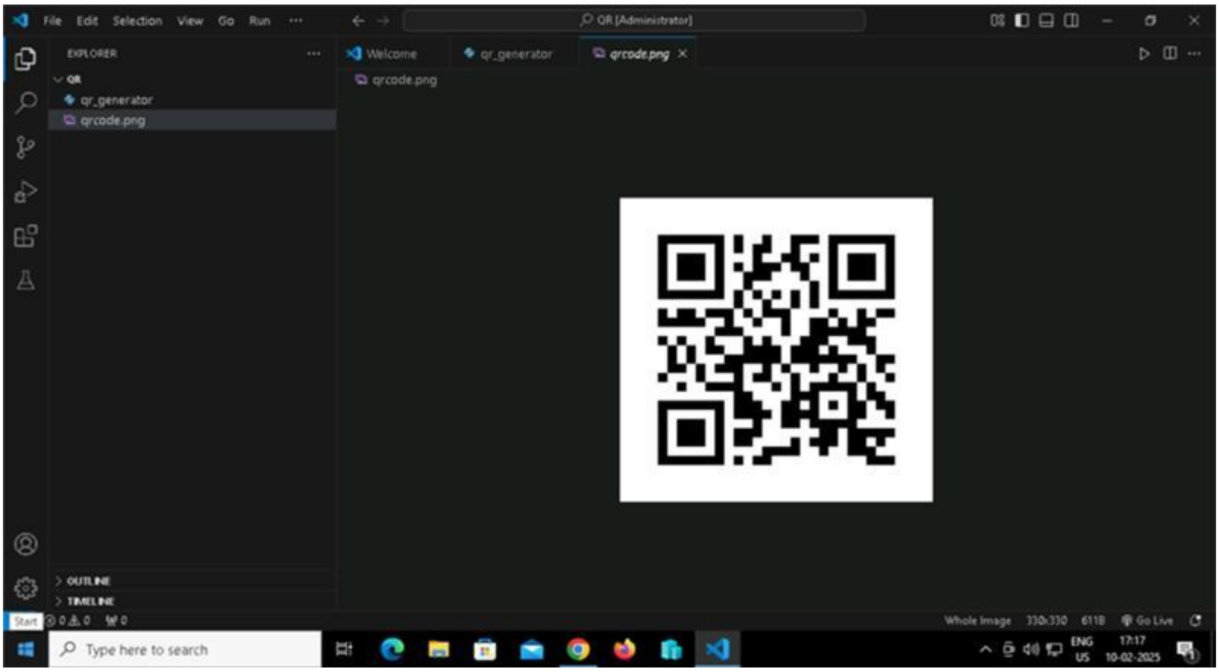
1.QR CODE GENERATOR



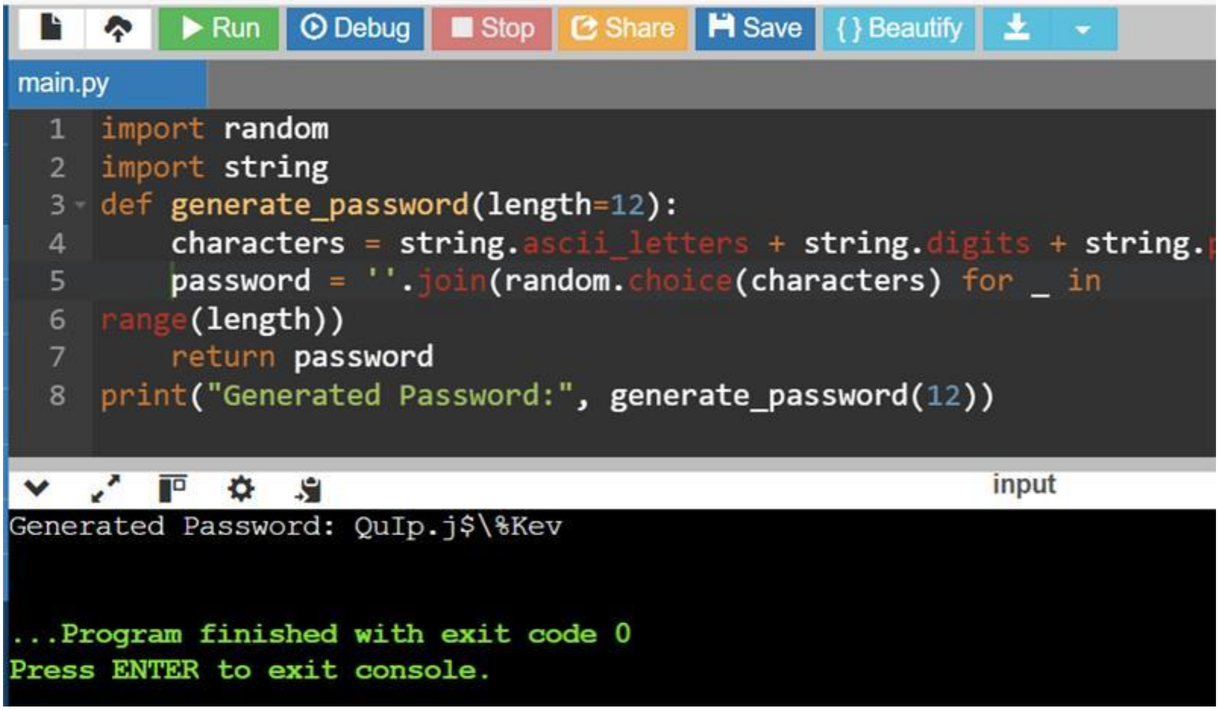
```python
import qrcode

# Take user input (text/link) to convert to a QR code
data = input("Enter text or URL: ")

# Generate the QR code
qr = qrcode.make(data)

# Save the QR code as an image
qr.save("qrcode.png")
print("QR Code generated and saved as 'qrcode.png'!")
```

```
PS C:\Users\Administrator\Desktop\QR> python qr_generator
PS C:\Users\Administrator\Desktop\QR> python qr_generator
Enter text or URL: This is My first QR
Enter text or URL: This is My first QR
QR Code generated and saved as 'qrcode.png'!
PS C:\Users\Administrator\Desktop\QR>
```

OUTPUT



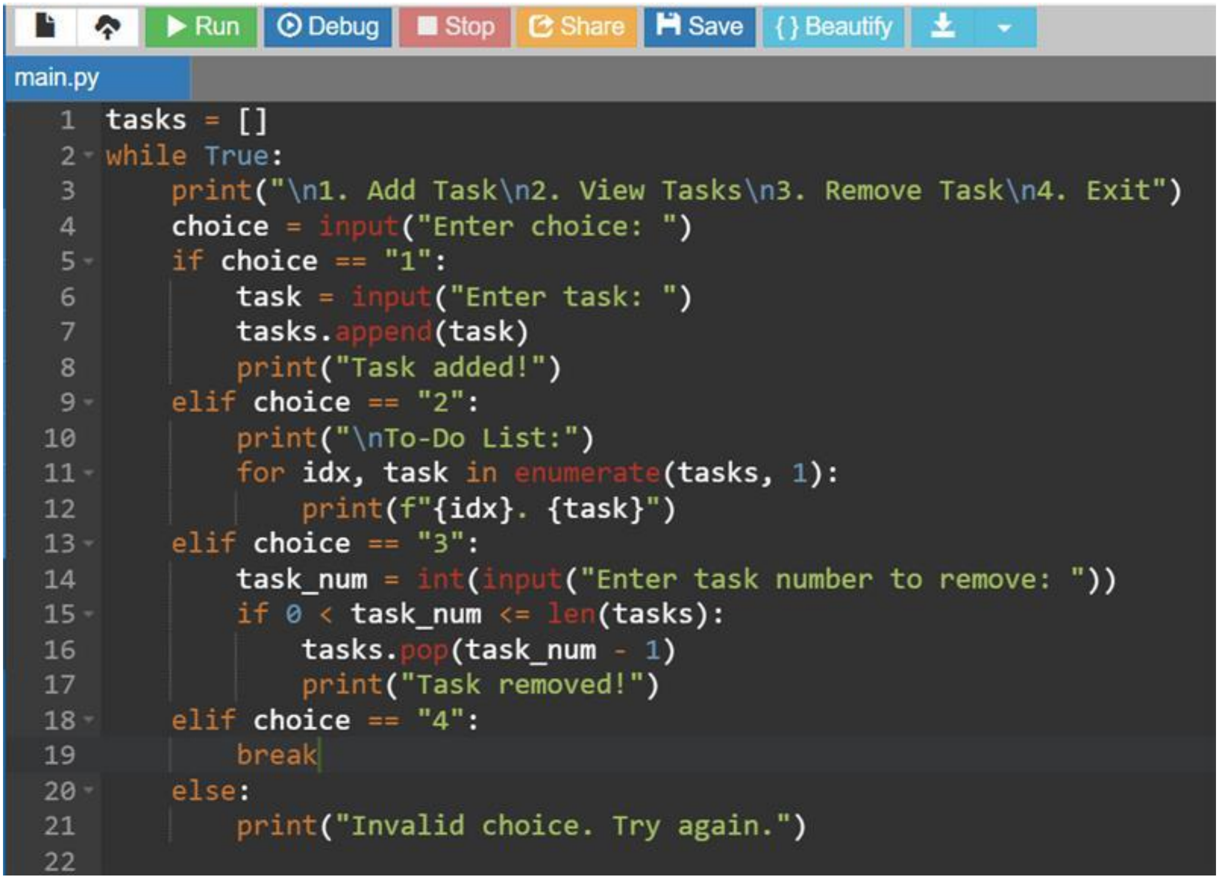
2. PASSWORD GENERATOR



```python
import random
import string
def generate_password(length=12):
    characters = string.ascii_letters + string.digits + string.
    password = ''.join(random.choice(characters) for _ in
range(length))
    return password
print("Generated Password:", generate_password(12))
```

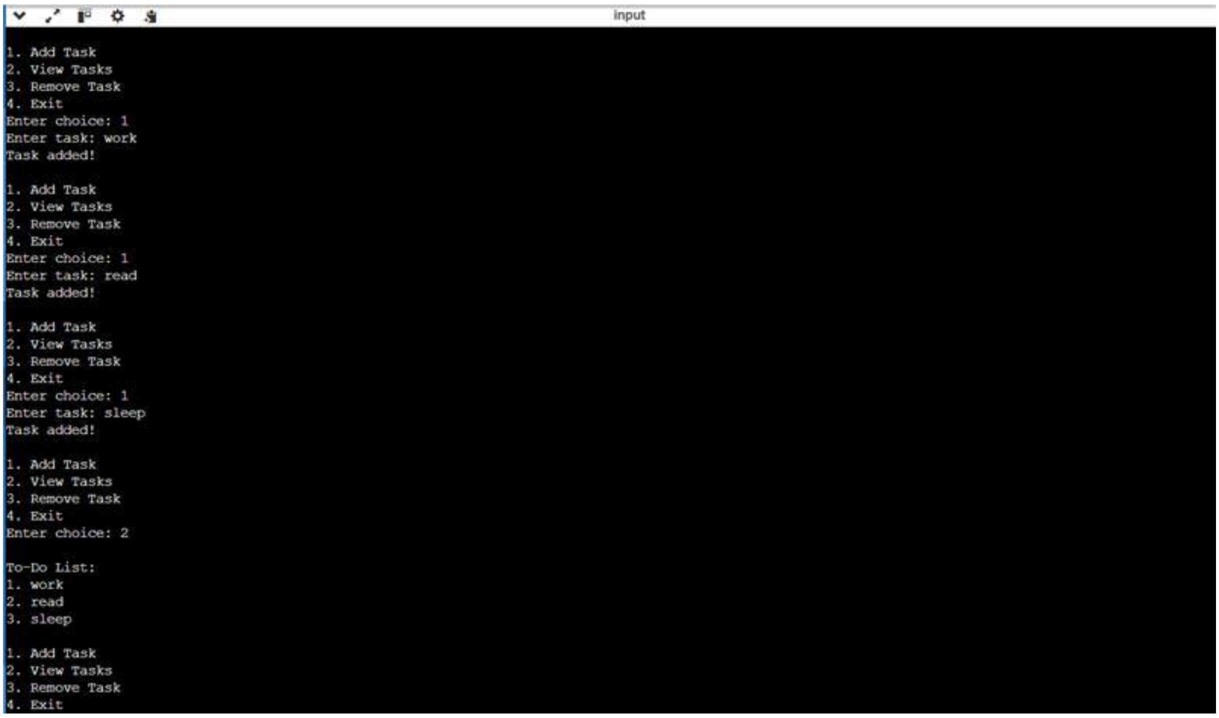
```
Generated Password: QuIp.j$\%Kev


...Program finished with exit code 0
Press ENTER to exit console.
```

3. TO-DO LIST



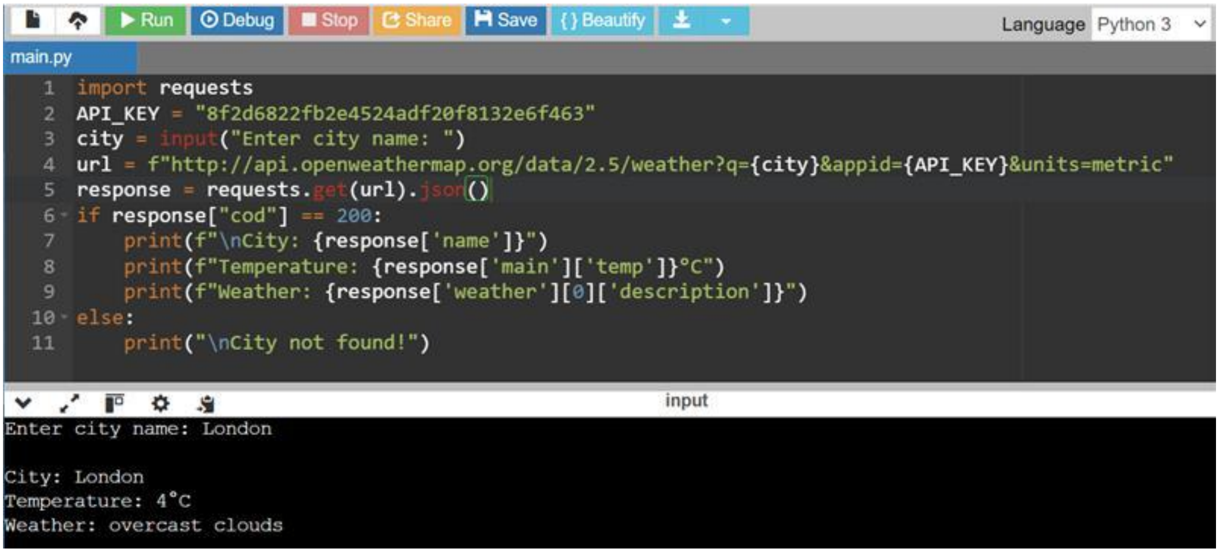
```python
tasks = []
while True:
    print("\n1. Add Task\n2. View Tasks\n3. Remove Task\n4. Exit")
    choice = input("Enter choice: ")
    if choice == "1":
        task = input("Enter task: ")
        tasks.append(task)
        print("Task added!")
    elif choice == "2":
        print("\nTo-Do List:")
        for idx, task in enumerate(tasks, 1):
            print(f"{idx}. {task}")
    elif choice == "3":
        task_num = int(input("Enter task number to remove: "))
        if 0 < task_num <= len(tasks):
            tasks.pop(task_num - 1)
            print("Task removed!")
    elif choice == "4":
        break
    else:
        print("Invalid choice. Try again.")

```

OUPUT



```
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 1
Enter task: work
Task added!

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 1
Enter task: read
Task added!

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 1
Enter task: sleep
Task added!

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 2

To-Do List:
1. work
2. read
3. sleep

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
```

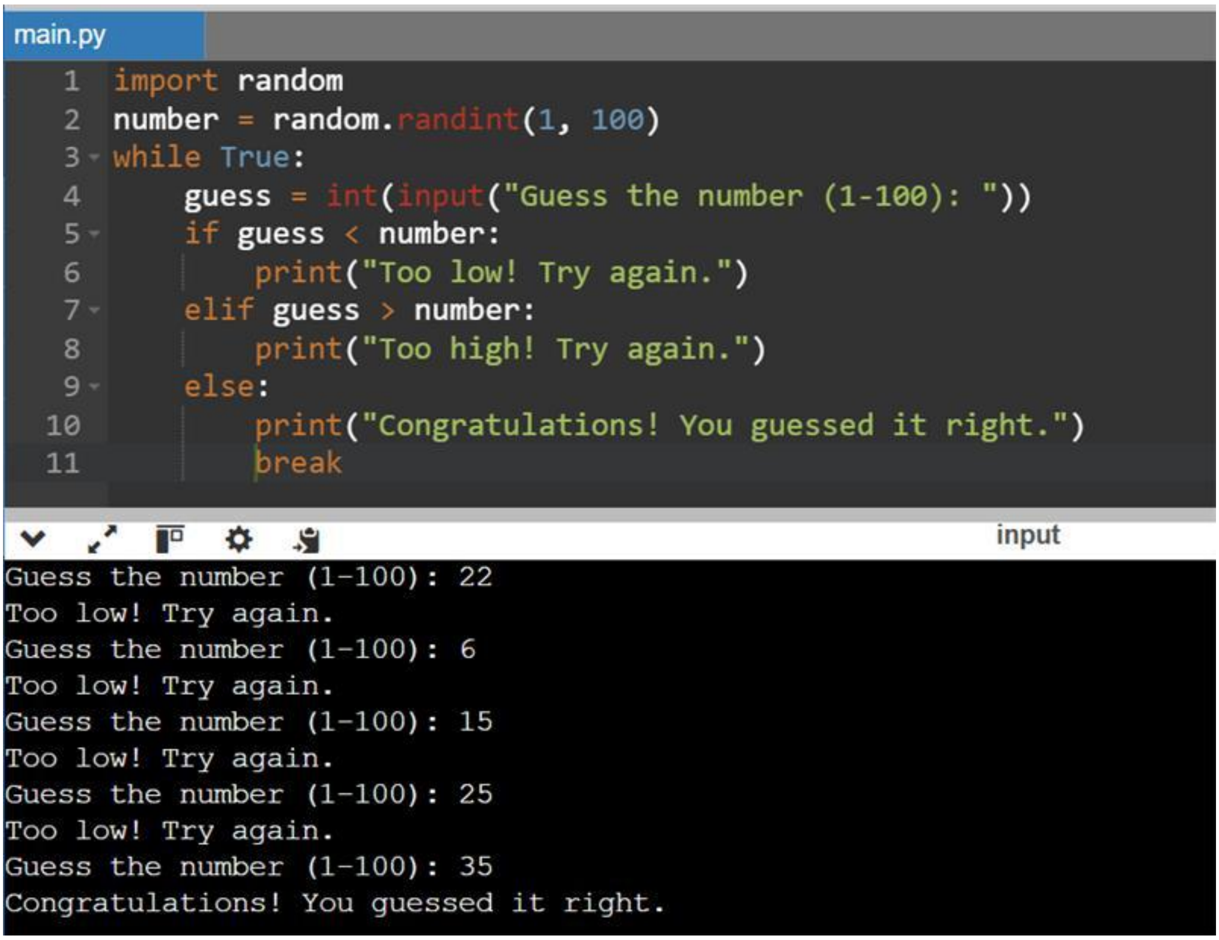
4. WEATHER APP (API Based)



```python
import requests
API_KEY = "8f2d6822fb2e4524adf20f8132e6f463"
city = input("Enter city name: ")
url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}&units=metric"
response = requests.get(url).json()
if response["cod"] == 200:
    print(f"\nCity: {response['name']}")
    print(f"Temperature: {response['main']['temp']}°C")
    print(f"Weather: {response['weather'][0]['description']}")
else:
    print("\nCity not found!")
```

```
Enter city name: London

City: London
Temperature: 4°C
Weather: overcast clouds
```

5. NUMBER GUESSING GAME



```python
import random
number = random.randint(1, 100)
while True:
    guess = int(input("Guess the number (1-100): "))
    if guess < number:
        print("Too low! Try again.")
    elif guess > number:
        print("Too high! Try again.")
    else:
        print("Congratulations! You guessed it right.")
        break
```

```
Guess the number (1-100): 22
Too low! Try again.
Guess the number (1-100): 6
Too low! Try again.
Guess the number (1-100): 15
Too low! Try again.
Guess the number (1-100): 25
Too low! Try again.
Guess the number (1-100): 35
Congratulations! You guessed it right.
```