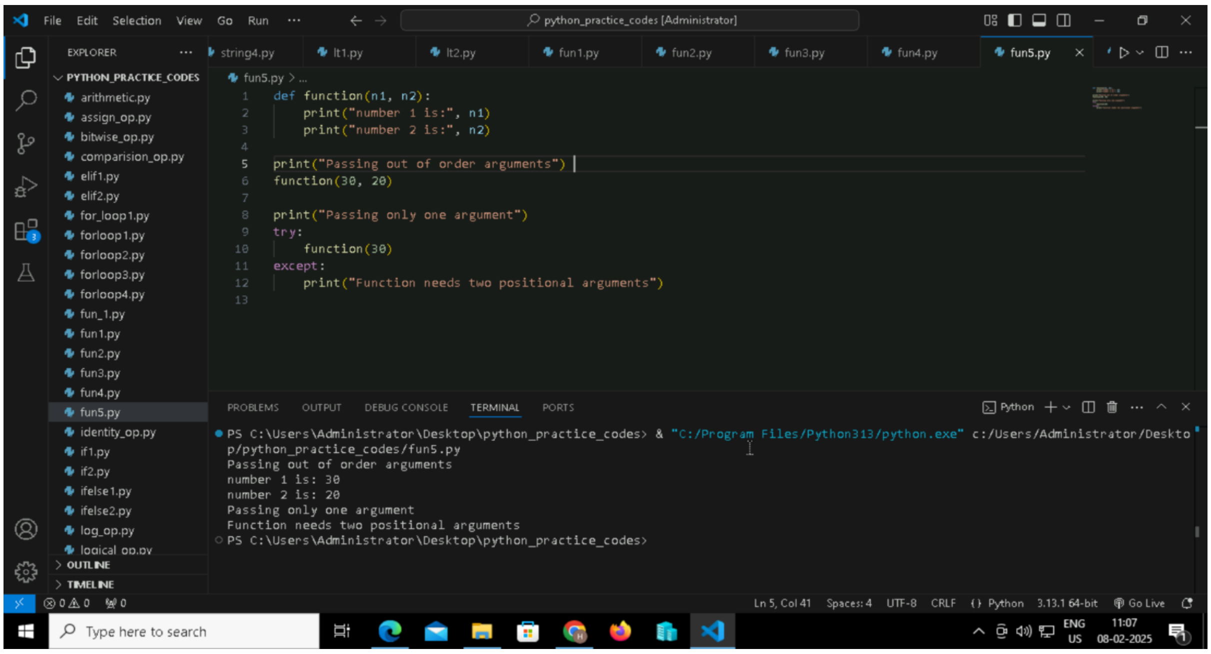
Python Functions

Python code to demonstrate the use of default arguments

Code and output:

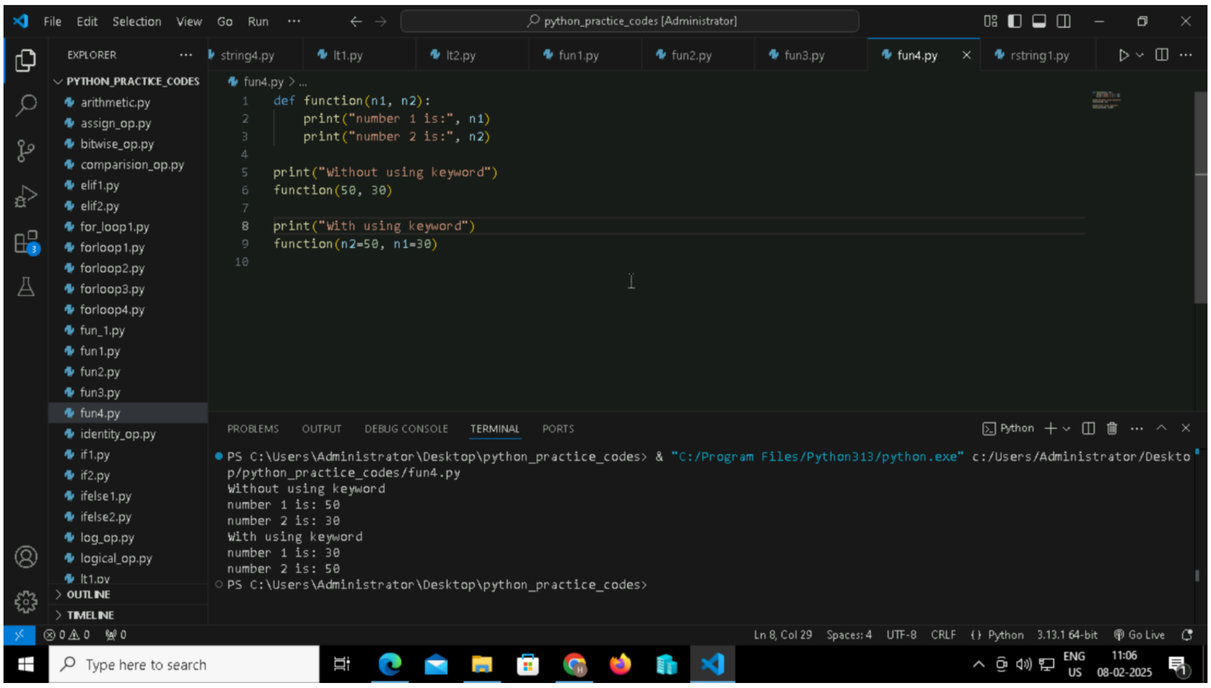


```python
def function(n1, n2):
    print("number 1 is:", n1)
    print("number 2 is:", n2)

print("Passing out of order arguments")
function(30, 20)

print("Passing only one argument")
try:
    function(30)
except:
    print("Function needs two positional arguments")
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Deskto
p/python_practice_codes/fun5.py
Passing out of order arguments
number 1 is: 30
number 2 is: 20
Passing only one argument
Function needs two positional arguments
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python code to demonstrate the use of keyword arguments

Code and output:

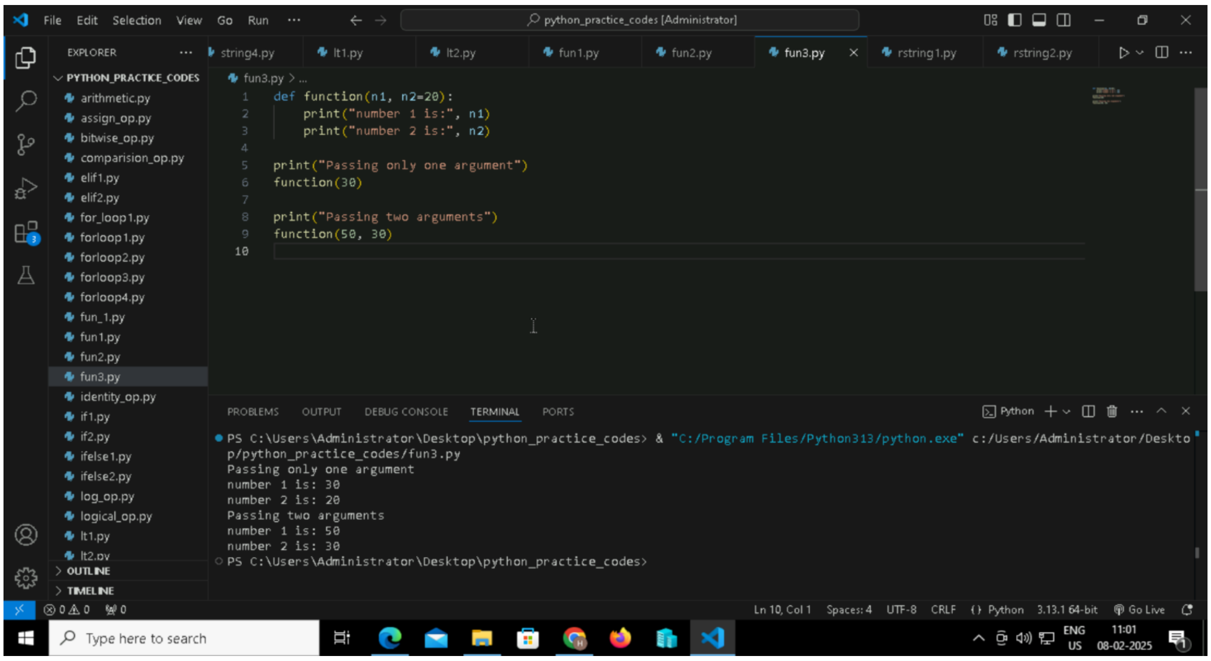


```python
def function(n1, n2):
    print("number 1 is:", n1)
    print("number 2 is:", n2)

print("Without using keyword")
function(50, 30)

print("With using keyword")
function(n2=50, n1=30)
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Deskto
p/python_practice_codes/fun4.py
Without using keyword
number 1 is: 50
number 2 is: 30
With using keyword
number 1 is: 30
number 2 is: 50
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python code to demonstrate the use of default arguments

Code and output:

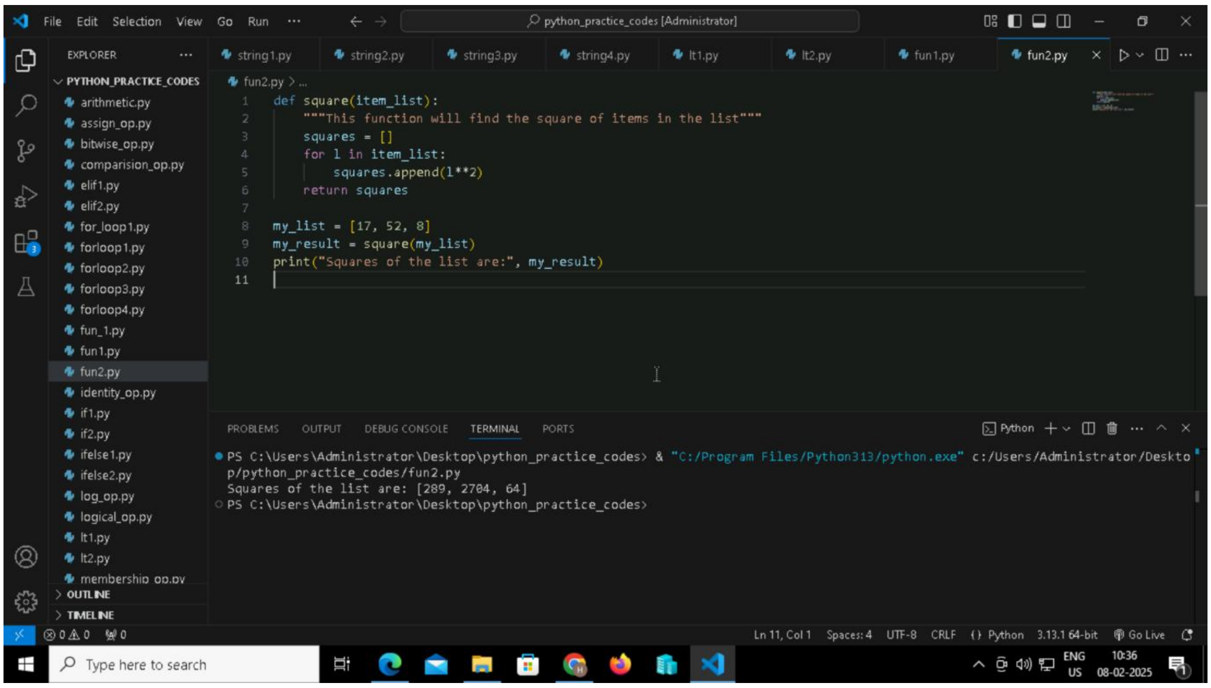


```python
def function(n1, n2=20):
    print("number 1 is:", n1)
    print("number 2 is:", n2)

print("Passing only one argument")
function(30)

print("Passing two arguments")
function(50, 30)
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/fun3.py
Passing only one argument
number 1 is: 30
number 2 is: 20
Passing two arguments
number 1 is: 50
number 2 is: 30
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Example Python Code for Pass by Reference vs. Value

Code and output:

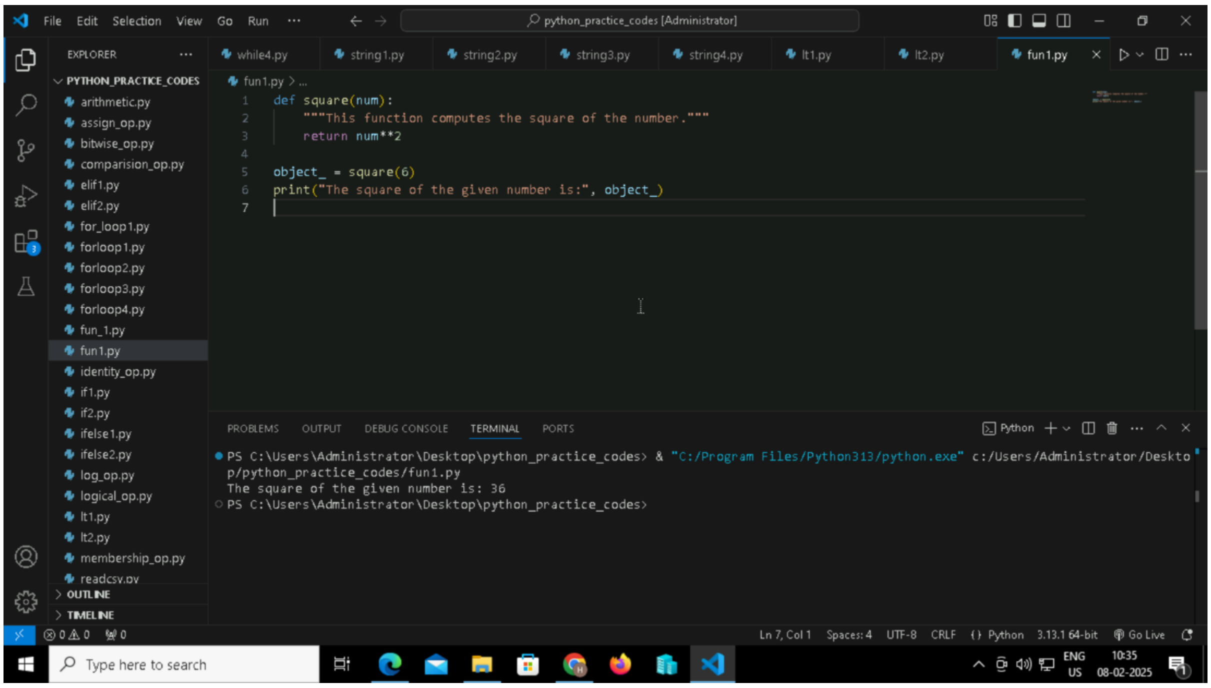


```python
def square(item_list):
    """This function will find the square of items in the list"""
    squares = []
    for l in item_list:
        squares.append(l**2)
    return squares

my_list = [17, 52, 8]
my_result = square(my_list)
print("Squares of the list are:", my_result)
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/fun2.py
Squares of the list are: [289, 2704, 64]
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Example Python Code for User-Defined function

Code and output:

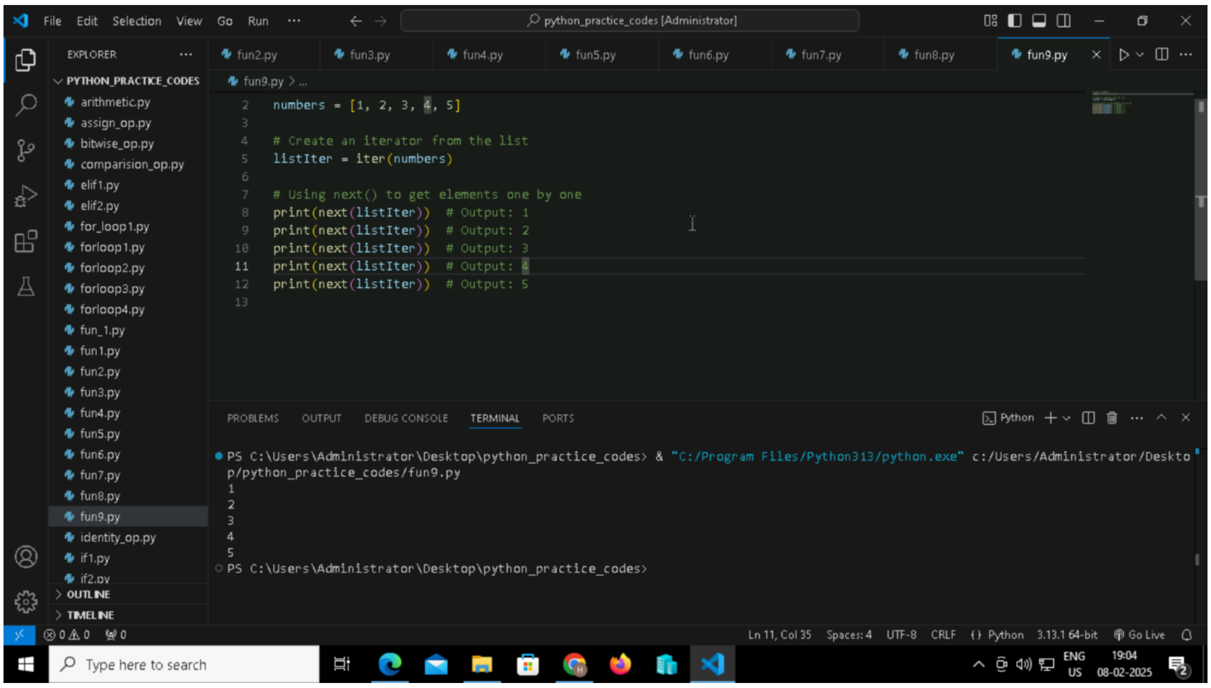


```python
def square(num):
    """This function computes the square of the number."""
    return num**2

object_ = square(6)
print("The square of the given number is:", object_)
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Deskto
p/python_practice_codes/fun1.py
The square of the given number is: 36
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Creating iterator and using next()

Code and output:

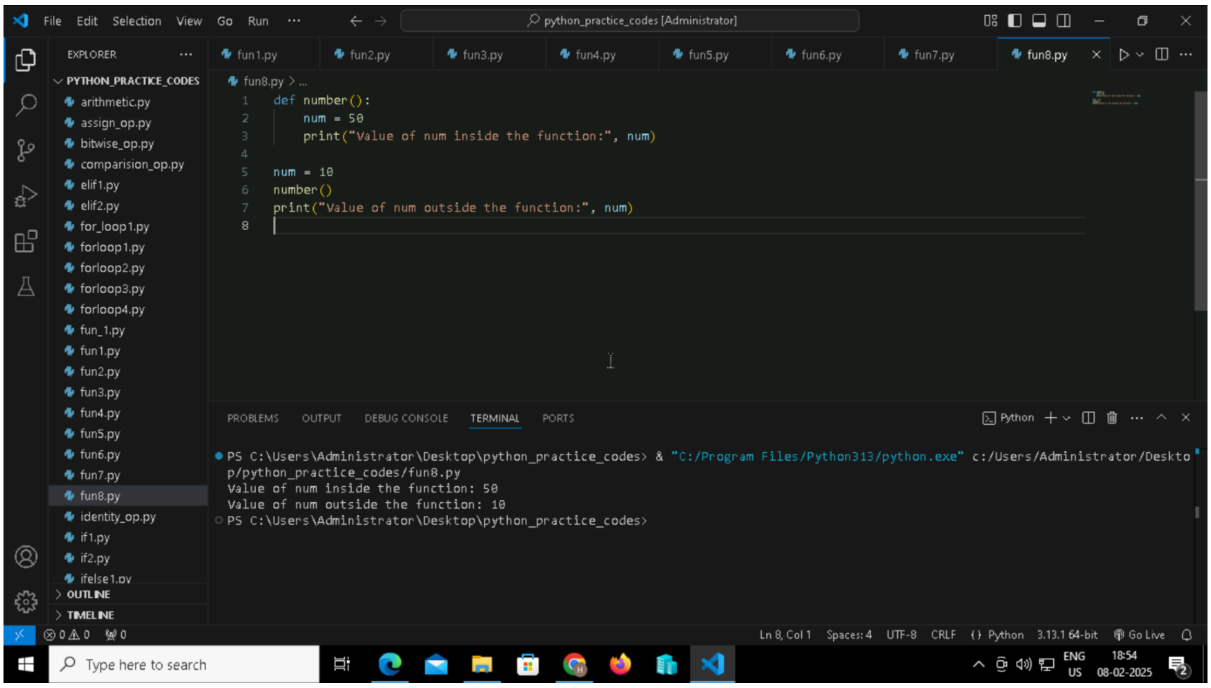


```python
numbers = [1, 2, 3, 4, 5]

# Create an iterator from the list
listIter = iter(numbers)

# Using next() to get elements one by one
print(next(listIter))  # Output: 1
print(next(listIter))  # Output: 2
print(next(listIter))  # Output: 3
print(next(listIter))  # Output: 4
print(next(listIter))  # Output: 5
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Deskto
p/python_practice_codes/fun9.py
1
2
3
4
5
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python code to demonstrate scope and lifetime of variables

Code and output:

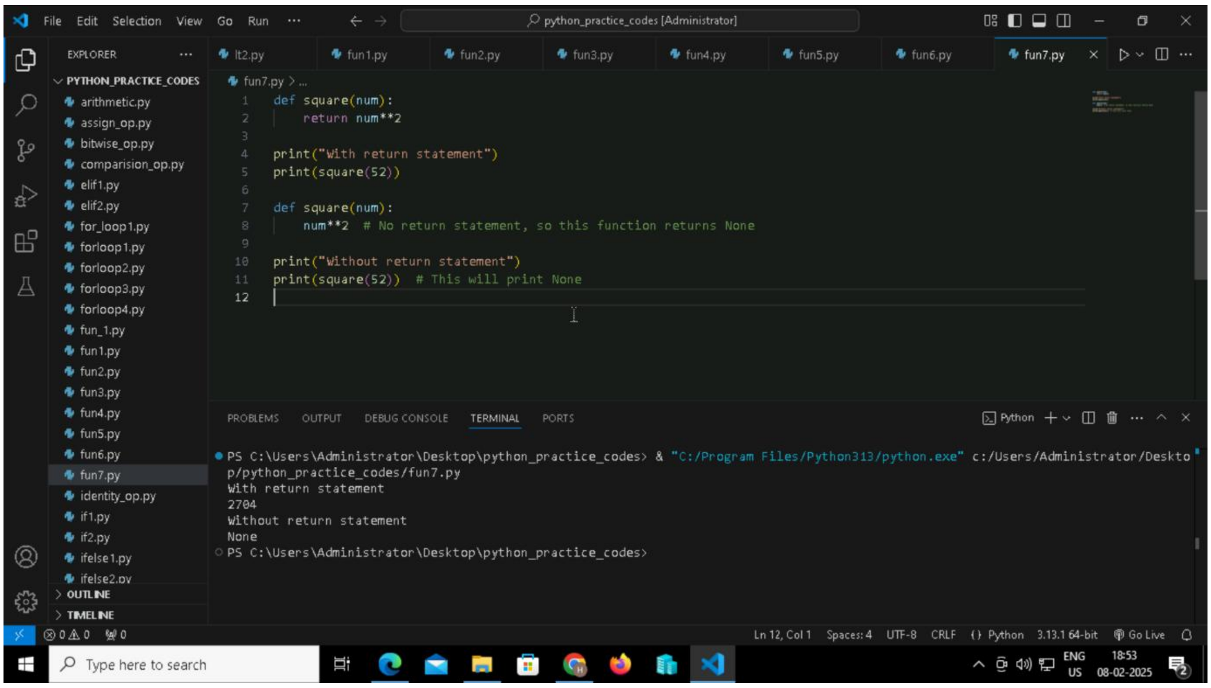


```python
def number():
    num = 50
    print("Value of num inside the function:", num)

num = 10
number()
print("Value of num outside the function:", num)
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/fun8.py
Value of num inside the function: 50
Value of num outside the function: 10
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python code to demonstrate the use of return statements

Code and output:

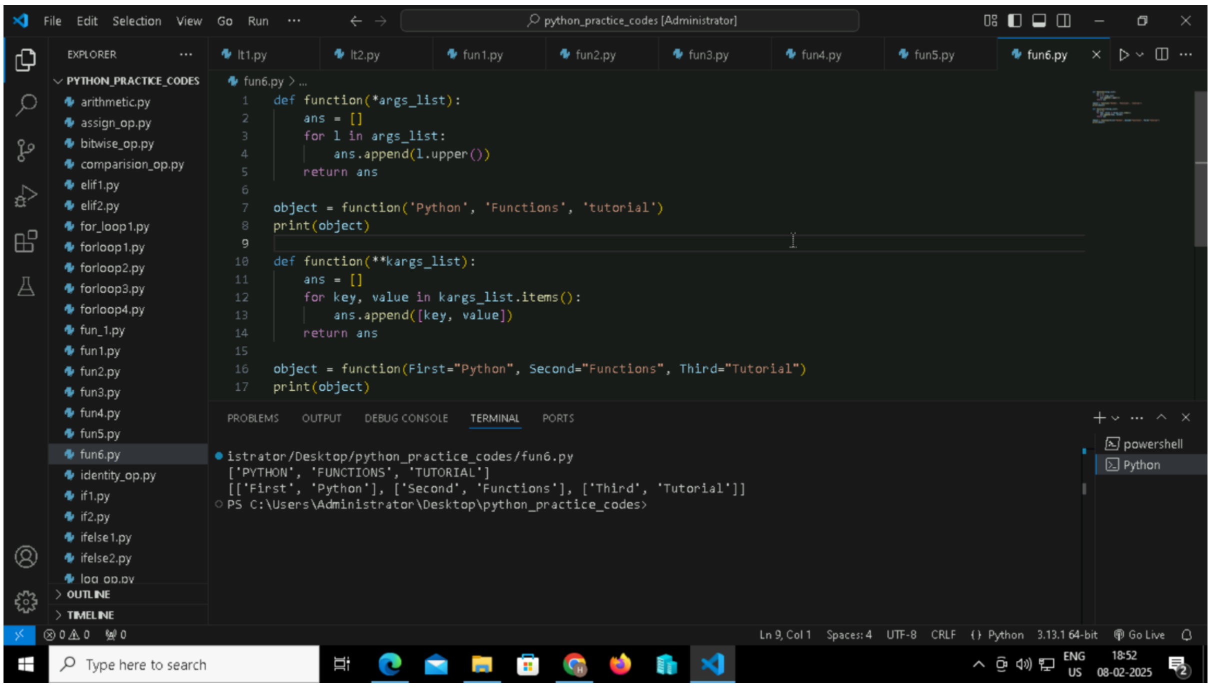


```python
def square(num):
    return num**2

print("With return statement")
print(square(52))

def square(num):
    num**2  # No return statement, so this function returns None

print("Without return statement")
print(square(52))  # This will print None
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/fun7.py
With return statement
2704
Without return statement
None
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python code to demonstrate the use of variable-length arguments

Code and output:

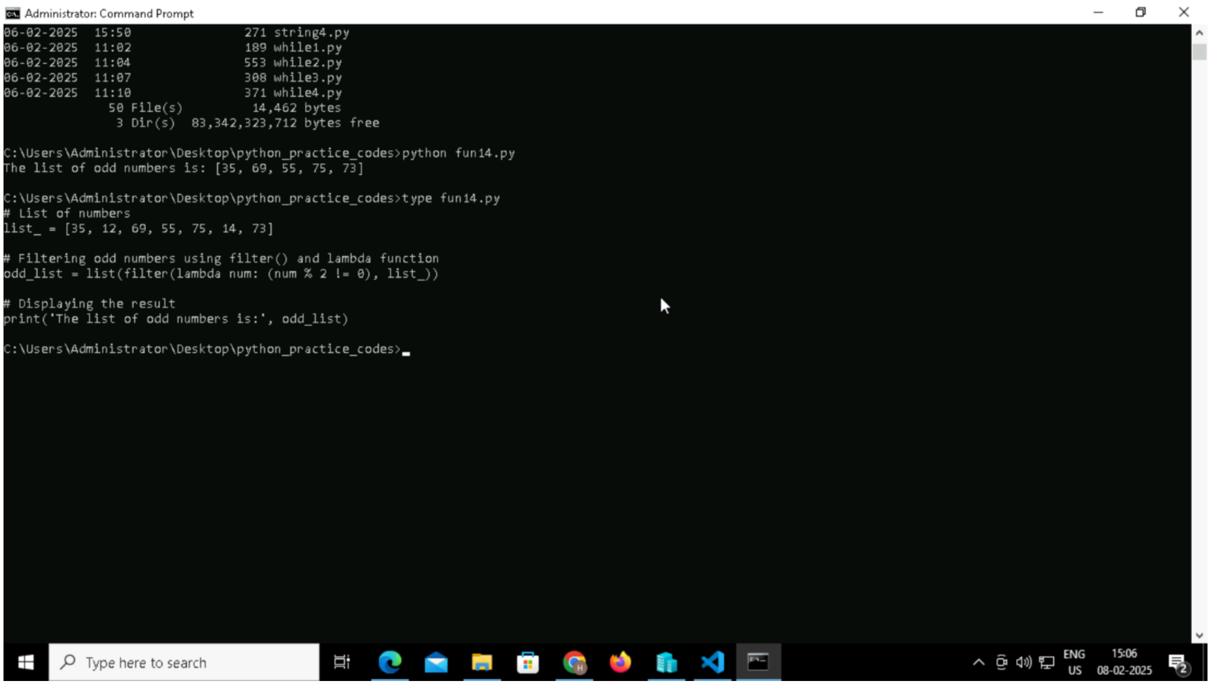


```python
def function(*args_list):
    ans = []
    for l in args_list:
        ans.append(l.upper())
    return ans

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)
```

```
istrator/Desktop/python_practice_codes/fun6.py
['PYTHON', 'FUNCTIONS', 'TUTORIAL']
[['First', 'Python'], ['Second', 'Functions'], ['Third', 'Tutorial']]
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

example of lambda function with filter() in Python

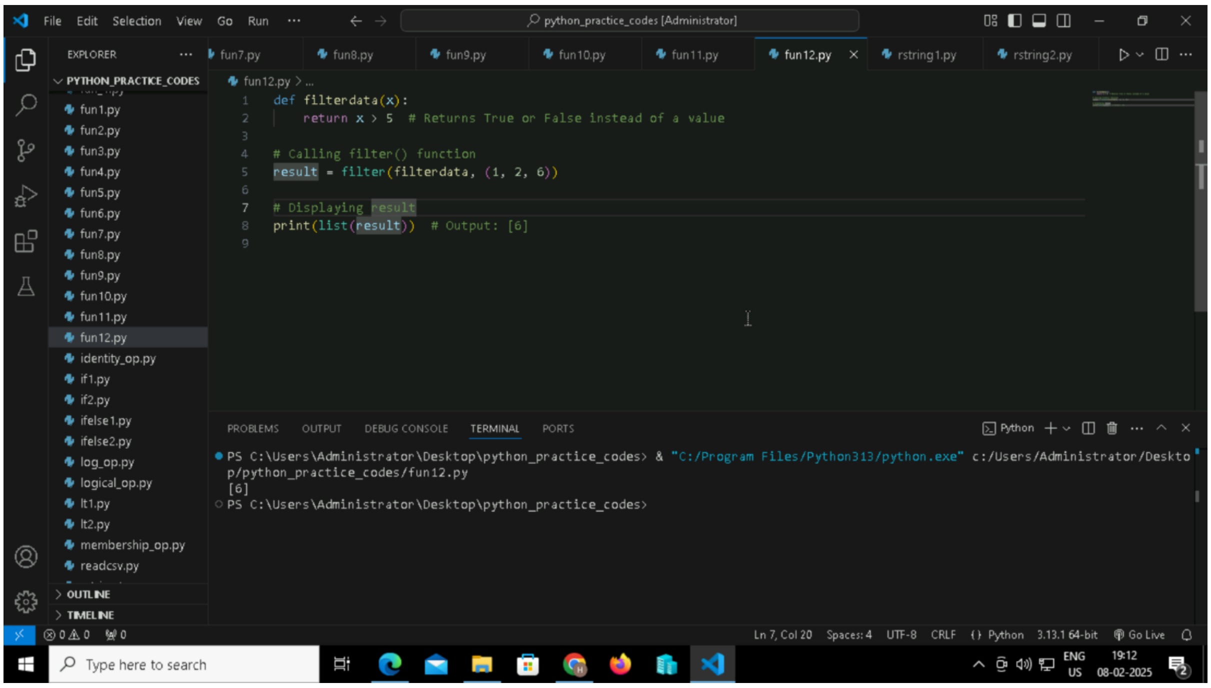
code and output:



```
06-02-2025  15:50               271 string4.py
06-02-2025  11:02               189 while1.py
06-02-2025  11:04               553 while2.py
06-02-2025  11:07               308 while3.py
06-02-2025  11:10               371 while4.py
              50 File(s)         14,462 bytes
               3 Dir(s)  83,342,323,712 bytes free

C:\Users\Administrator\Desktop\python_practice_codes>python fun14.py
The list of odd numbers is: [35, 69, 55, 75, 73]

C:\Users\Administrator\Desktop\python_practice_codes>type fun14.py
# List of numbers
list_ = [35, 12, 69, 55, 75, 14, 73]

# Filtering odd numbers using filter() and lambda function
odd_list = list(filter(lambda num: (num % 2 != 0), list_))

# Displaying the result
print('The list of odd numbers is:', odd_list)

C:\Users\Administrator\Desktop\python_practice_codes>
```

Python filter() function example

Code and example:

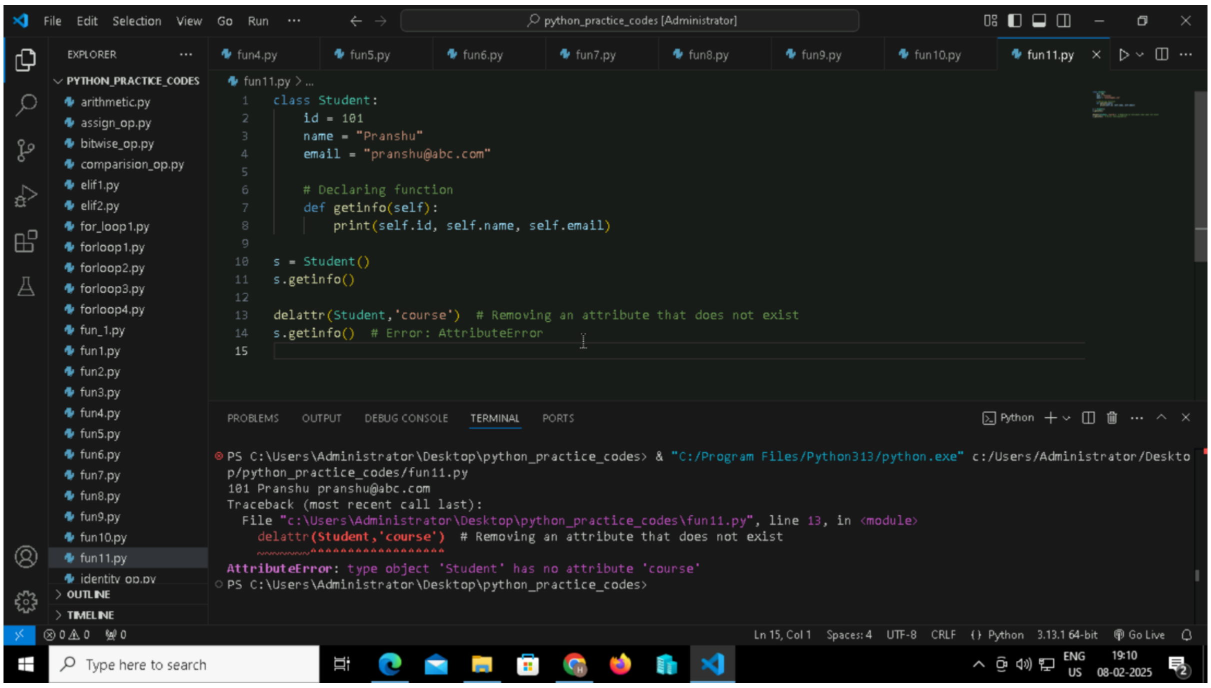


```python
def filterdata(x):
    return x > 5  # Returns True or False instead of a value

# Calling filter() function
result = filter(filterdata, (1, 2, 6))

# Displaying result
print(list(result))  # Output: [6]
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Deskto
p/python_practice_codes/fun12.py
[6]
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python delattr() Function Example

Code and output:

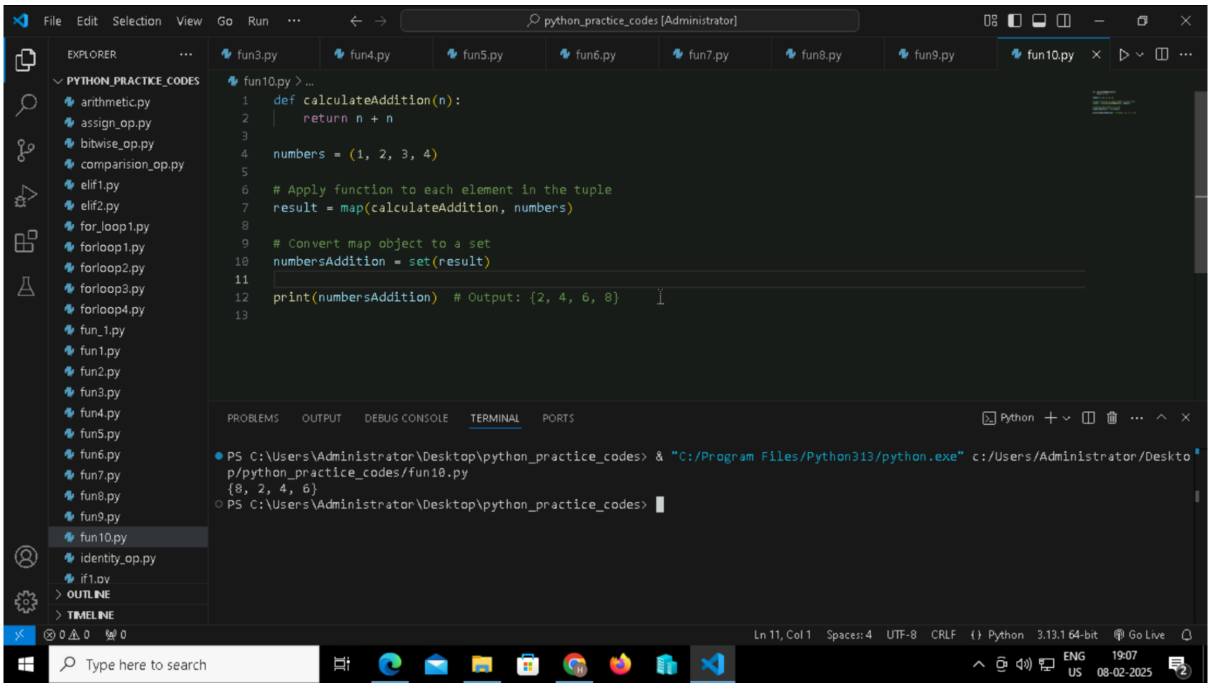


```python
class Student:
    id = 101
    name = "Pranshu"
    email = "pranshu@abc.com"

    # Declaring function
    def getinfo(self):
        print(self.id, self.name, self.email)

s = Student()
s.getinfo()

delattr(Student,'course')  # Removing an attribute that does not exist
s.getinfo()  # Error: AttributeError
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Deskto
p/python_practice_codes/fun11.py
101 Pranshu pranshu@abc.com
Traceback (most recent call last):
  File "c:\Users\Administrator\Desktop\python_practice_codes\fun11.py", line 13, in <module>
    delattr(Student,'course')  # Removing an attribute that does not exist
    ~~~~~~~^^^^^^^^^^^^^^^^^^
AttributeError: type object 'Student' has no attribute 'course'
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

Python map() Function Example

Code and output:

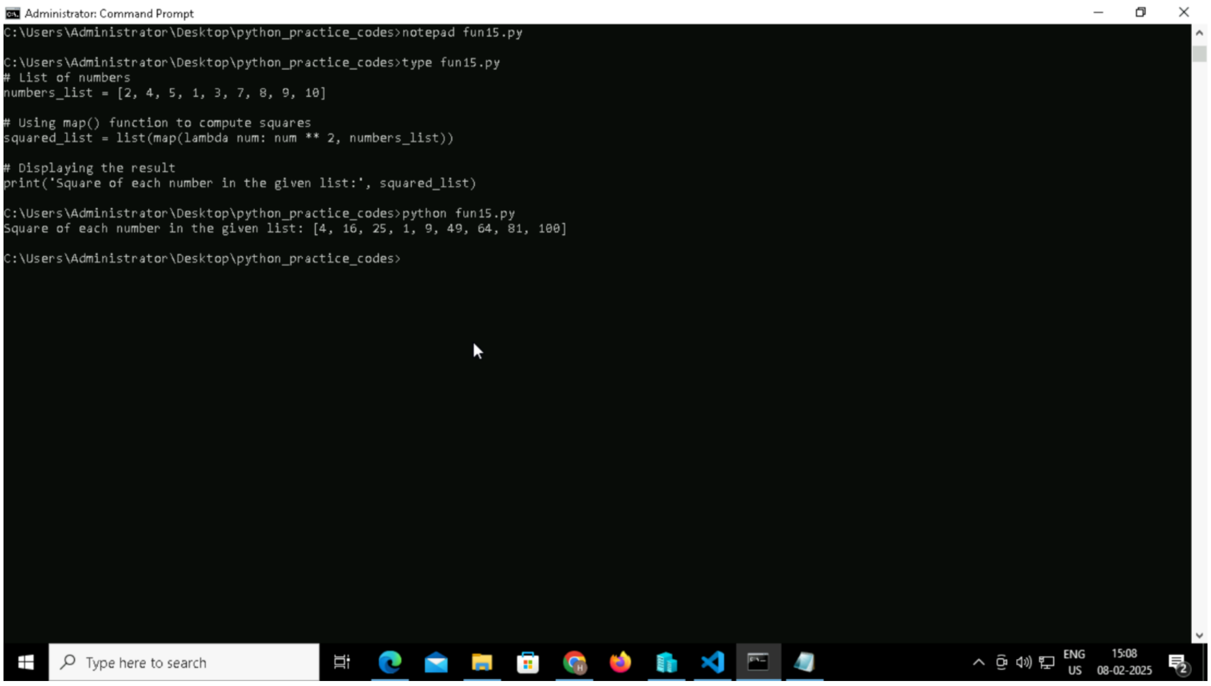


```python
def calculateAddition(n):
    return n + n

numbers = (1, 2, 3, 4)

# Apply function to each element in the tuple
result = map(calculateAddition, numbers)

# Convert map object to a set
numbersAddition = set(result)

print(numbersAddition)  # Output: {2, 4, 6, 8}
```

```
PS C:\Users\Administrator\Desktop\python_practice_codes> & "C:/Program Files/Python313/python.exe" c:/Users/Administrator/Desktop/python_practice_codes/fun10.py
{8, 2, 4, 6}
PS C:\Users\Administrator\Desktop\python_practice_codes>
```

example of lambda function with map() in Python

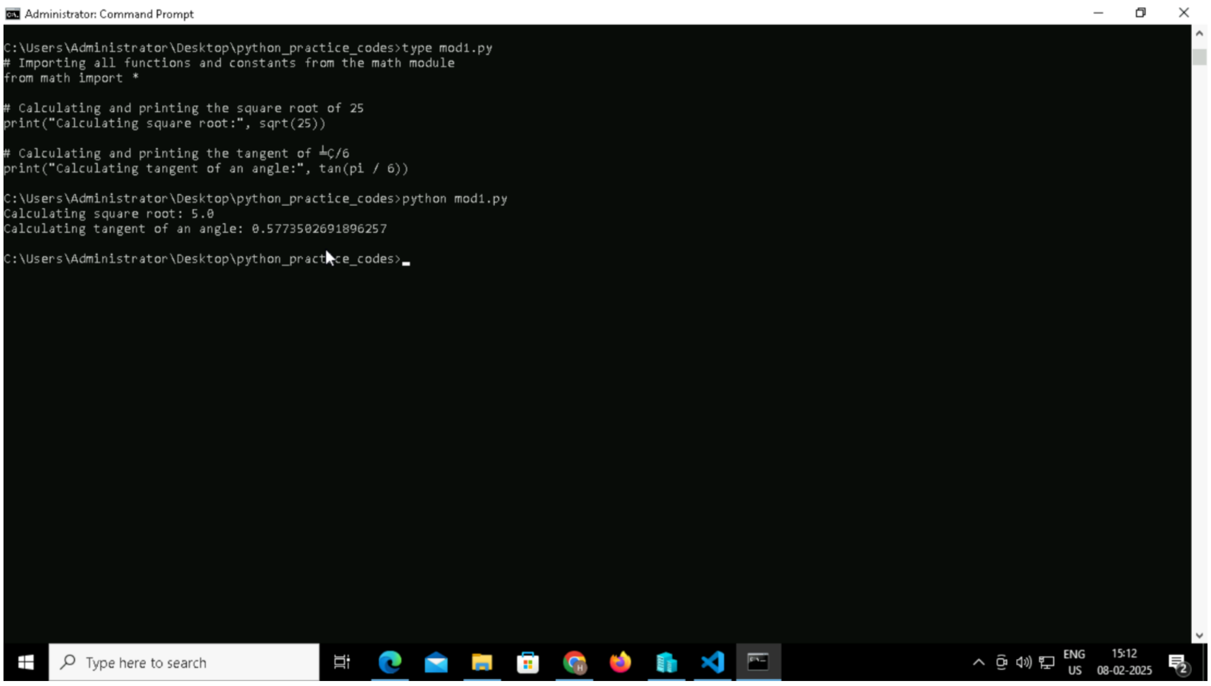
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>notepad fun15.py

C:\Users\Administrator\Desktop\python_practice_codes>type fun15.py
# List of numbers
numbers_list = [2, 4, 5, 1, 3, 7, 8, 9, 10]

# Using map() function to compute squares
squared_list = list(map(lambda num: num ** 2, numbers_list))

# Displaying the result
print('Square of each number in the given list:', squared_list)

C:\Users\Administrator\Desktop\python_practice_codes>python fun15.py
Square of each number in the given list: [4, 16, 25, 1, 9, 49, 64, 81, 100]

C:\Users\Administrator\Desktop\python_practice_codes>
```

importing the complete math module using *

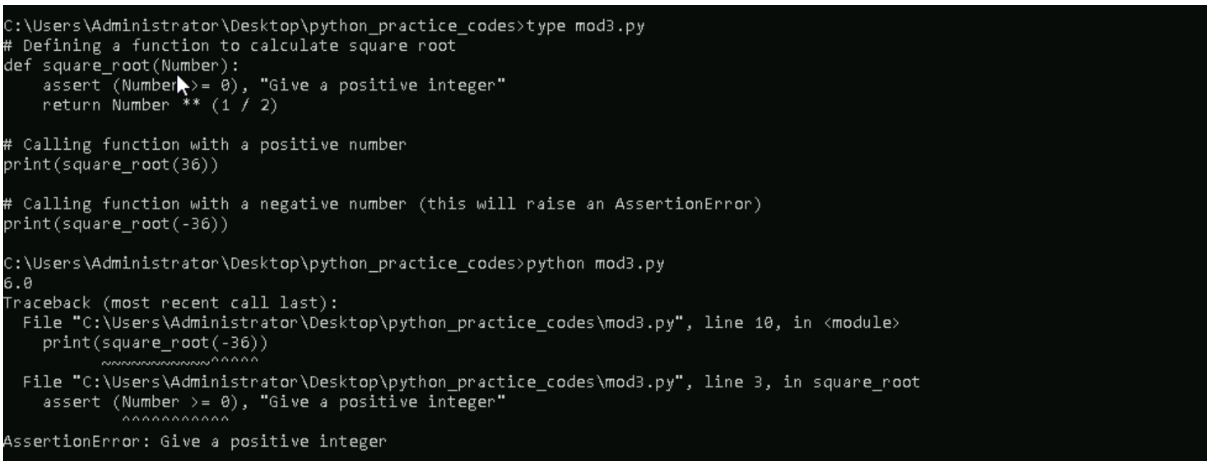
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type mod1.py
# Importing all functions and constants from the math module
from math import *

# Calculating and printing the square root of 25
print("Calculating square root:", sqrt(25))

# Calculating and printing the tangent of ╧Ç/6
print("Calculating tangent of an angle:", tan(pi / 6))

C:\Users\Administrator\Desktop\python_practice_codes>python mod1.py
Calculating square root: 5.0
Calculating tangent of an angle: 0.5773502691896257

C:\Users\Administrator\Desktop\python_practice_codes>_
```

Python program to show how to use assert keyword

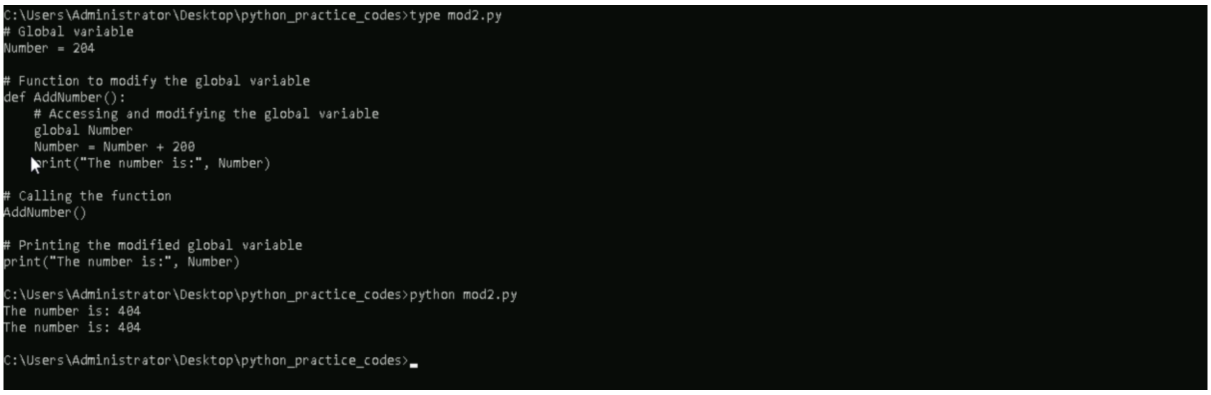
Code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type mod3.py
# Defining a function to calculate square root
def square_root(Number):
    assert (Number >= 0), "Give a positive integer"
    return Number ** (1 / 2)

# Calling function with a positive number
print(square_root(36))

# Calling function with a negative number (this will raise an AssertionError)
print(square_root(-36))

C:\Users\Administrator\Desktop\python_practice_codes>python mod3.py
6.0
Traceback (most recent call last):
  File "C:\Users\Administrator\Desktop\python_practice_codes\mod3.py", line 10, in <module>
    print(square_root(-36))
          ~~~~~~~~~~~^^^^^
  File "C:\Users\Administrator\Desktop\python_practice_codes\mod3.py", line 3, in square_root
    assert (Number >= 0), "Give a positive integer"
            ^^^^^^^^^^^^
AssertionError: Give a positive integer
```

defining a function with the name Add Number

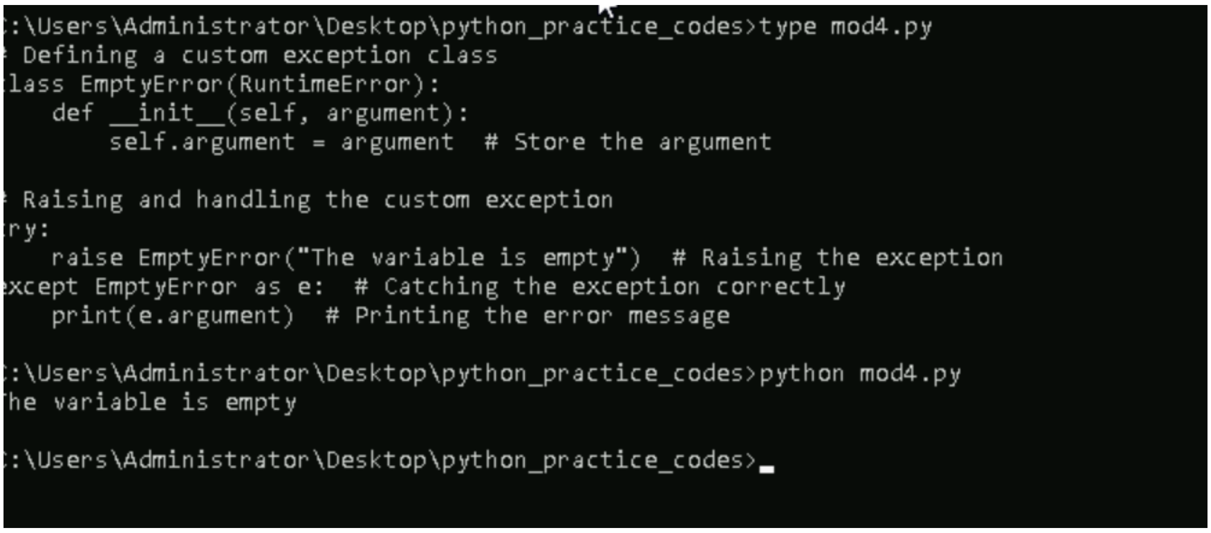
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type mod2.py
# Global variable
Number = 204

# Function to modify the global variable
def AddNumber():
    # Accessing and modifying the global variable
    global Number
    Number = Number + 200
    print("The number is:", Number)

# Calling the function
AddNumber()

# Printing the modified global variable
print("The number is:", Number)

C:\Users\Administrator\Desktop\python_practice_codes>python mod2.py
The number is: 404
The number is: 404

C:\Users\Administrator\Desktop\python_practice_codes>
```

illustration of a Runtime Error:

code and output:

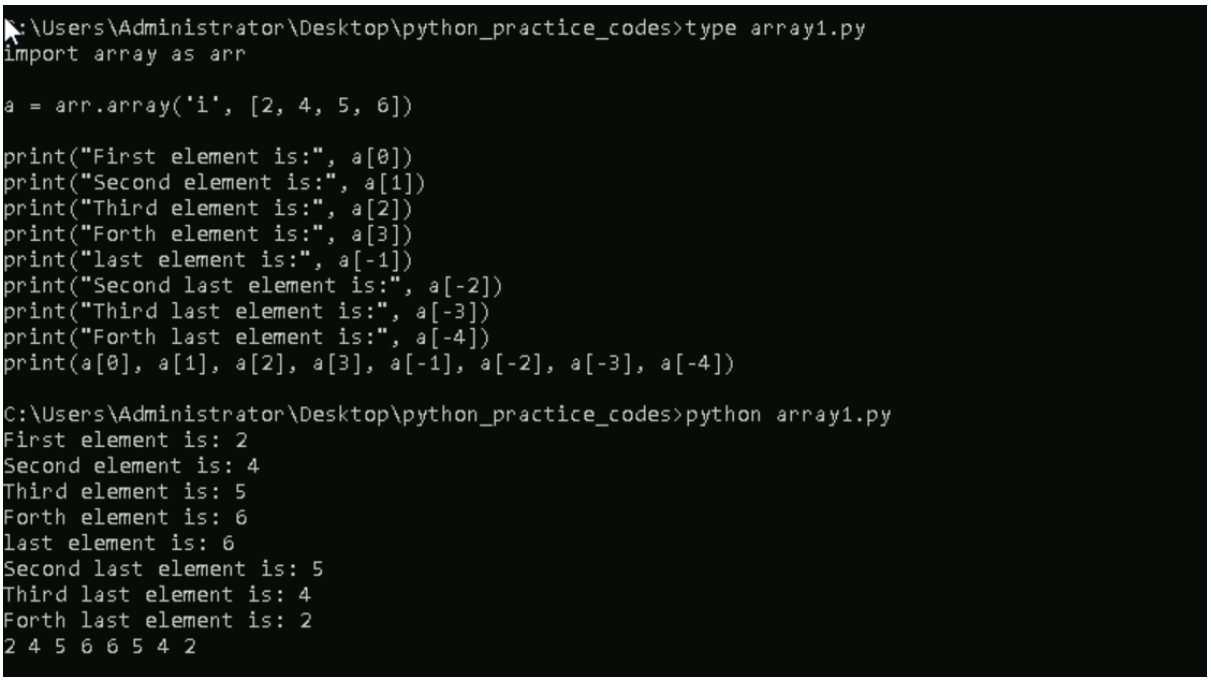


```
:\Users\Administrator\Desktop\python_practice_codes>type mod4.py
 Defining a custom exception class
lass EmptyError(RuntimeError):
   def __init__(self, argument):
       self.argument = argument  # Store the argument

 Raising and handling the custom exception
ry:
   raise EmptyError("The variable is empty")  # Raising the exception
xcept EmptyError as e:  # Catching the exception correctly
   print(e.argument)  # Printing the error message

:\Users\Administrator\Desktop\python_practice_codes>python mod4.py
he variable is empty

:\Users\Administrator\Desktop\python_practice_codes>
```

Arrays:

example of how we access the elements of an array using its index value in Python:

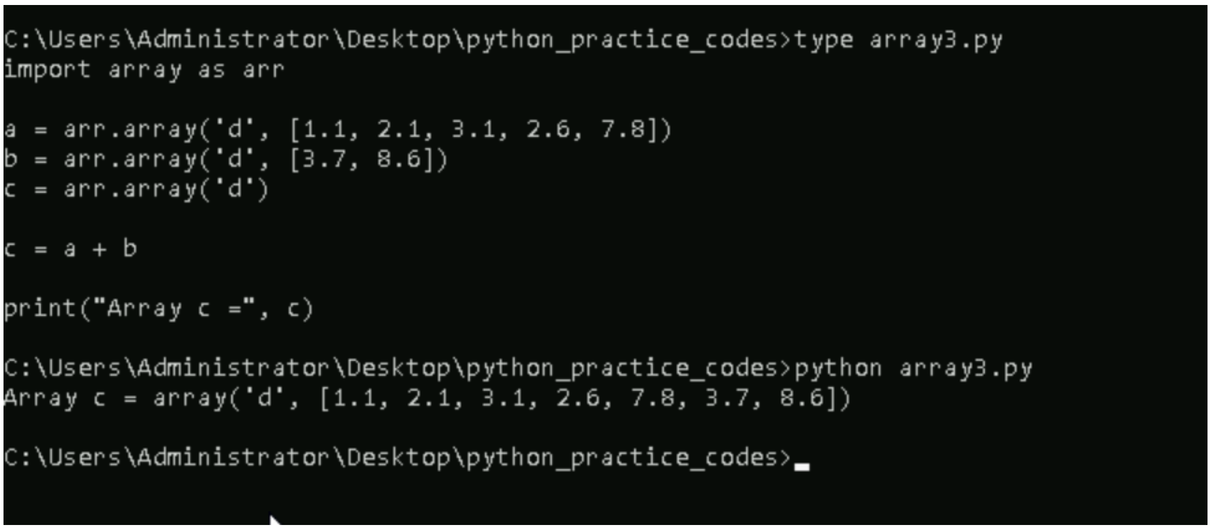
code and output:



```
:\Users\Administrator\Desktop\python_practice_codes>type array1.py
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])

C:\Users\Administrator\Desktop\python_practice_codes>python array1.py
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
```

concatenate any two arrays using the + symbol:

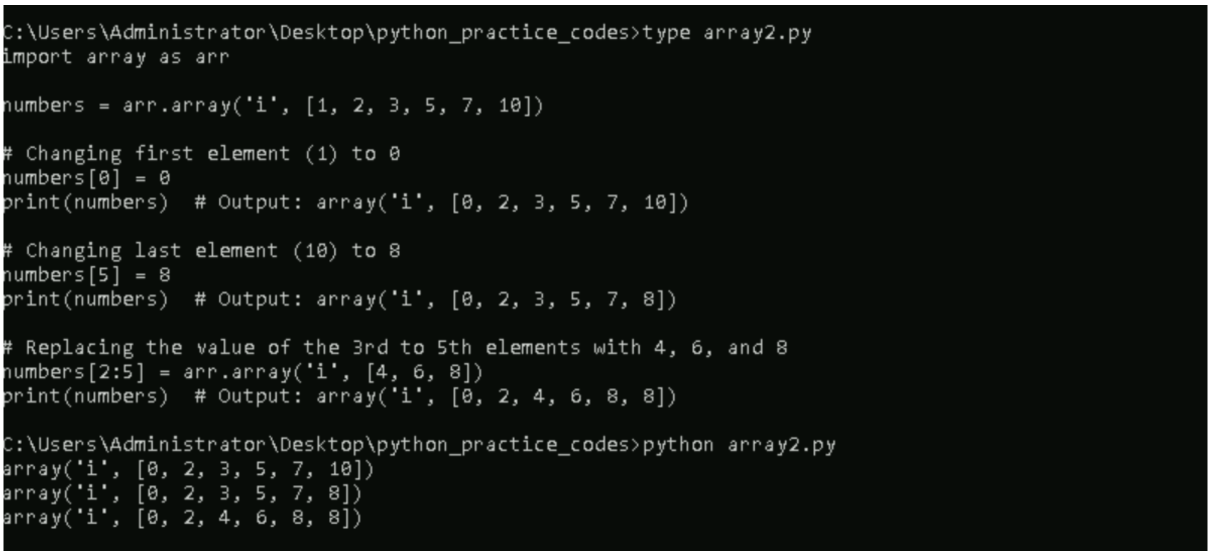
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type array3.py
import array as arr

a = arr.array('d', [1.1, 2.1, 3.1, 2.6, 7.8])
b = arr.array('d', [3.7, 8.6])
c = arr.array('d')

c = a + b

print("Array c =", c)

C:\Users\Administrator\Desktop\python_practice_codes>python array3.py
Array c = array('d', [1.1, 2.1, 3.1, 2.6, 7.8, 3.7, 8.6])

C:\Users\Administrator\Desktop\python_practice_codes>_
```

example, we can change or add or replace any element from the Array in Python:

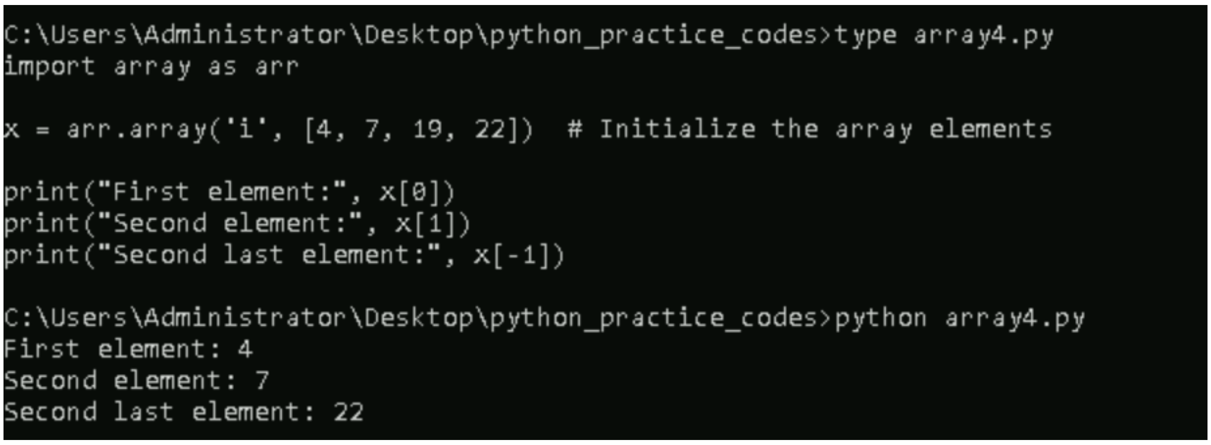
code and example:



```
:\Users\Administrator\Desktop\python_practice_codes>type array2.py
import array as arr

numbers = arr.array('i', [1, 2, 3, 5, 7, 10])

# Changing first element (1) to 0
numbers[0] = 0
print(numbers)  # Output: array('i', [0, 2, 3, 5, 7, 10])

# Changing last element (10) to 8
numbers[5] = 8
print(numbers)  # Output: array('i', [0, 2, 3, 5, 7, 8])

# Replacing the value of the 3rd to 5th elements with 4, 6, and 8
numbers[2:5] = arr.array('i', [4, 6, 8])
print(numbers)  # Output: array('i', [0, 2, 4, 6, 8, 8])

:\Users\Administrator\Desktop\python_practice_codes>python array2.py
array('i', [0, 2, 3, 5, 7, 10])
array('i', [0, 2, 3, 5, 7, 8])
array('i', [0, 2, 4, 6, 8, 8])
```

e example, first, we imported an array and defined a variable named "x," which holds the value of an array

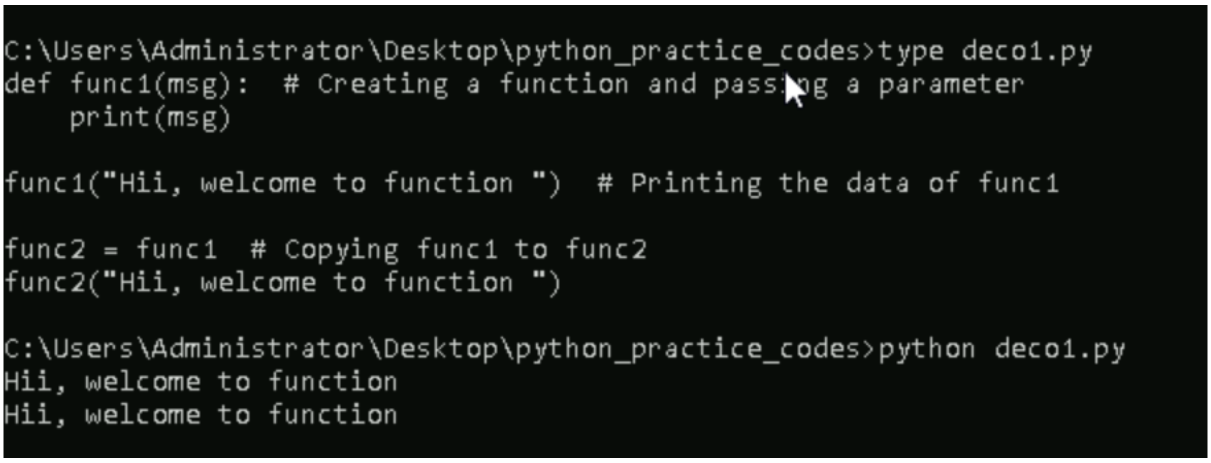
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type array4.py
import array as arr

x = arr.array('i', [4, 7, 19, 22])  # Initialize the array elements

print("First element:", x[0])
print("Second element:", x[1])
print("Second last element:", x[-1])

C:\Users\Administrator\Desktop\python_practice_codes>python array4.py
First element: 4
Second element: 7
Second last element: 22
```

creating a function and passing the parameter

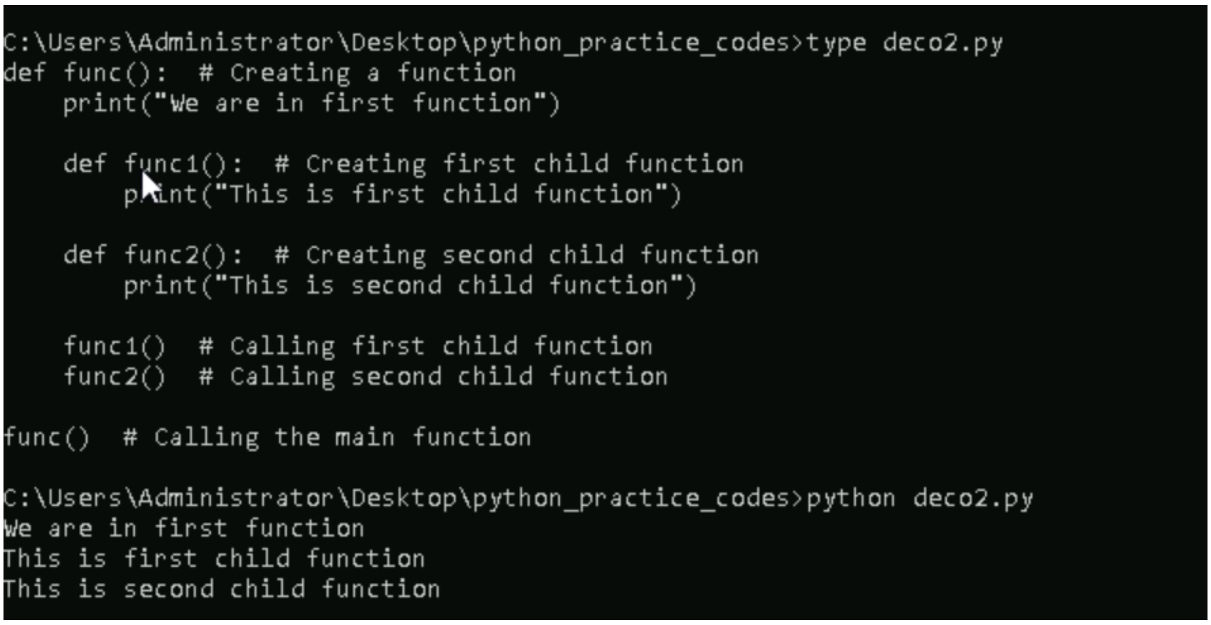
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type deco1.py
def func1(msg):  # Creating a function and passing a parameter
    print(msg)

func1("Hii, welcome to function ")  # Printing the data of func1

func2 = func1  # Copying func1 to func2
func2("Hii, welcome to function ")

C:\Users\Administrator\Desktop\python_practice_codes>python deco1.py
Hii, welcome to function
Hii, welcome to function
```

Inner Function:

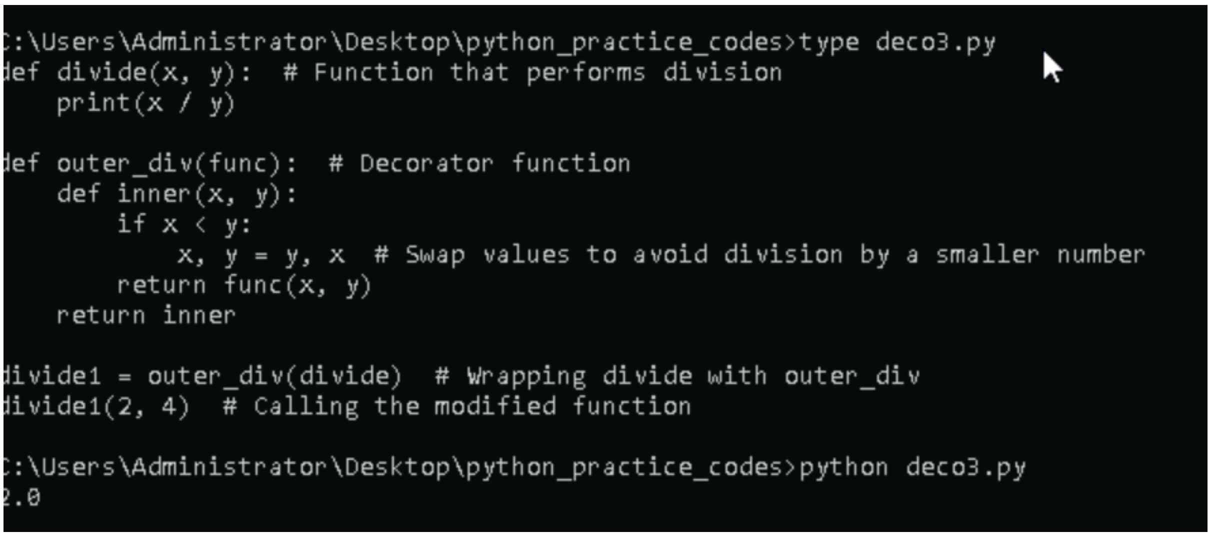
Code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type deco2.py
def func():  # Creating a function
    print("We are in first function")

    def func1():  # Creating first child function
        print("This is first child function")

    def func2():  # Creating second child function
        print("This is second child function")

    func1()  # Calling first child function
    func2()  # Calling second child function

func()  # Calling the main function

C:\Users\Administrator\Desktop\python_practice_codes>python deco2.py
We are in first function
This is first child function
This is second child function
```

example to understand the parameterized decorator function

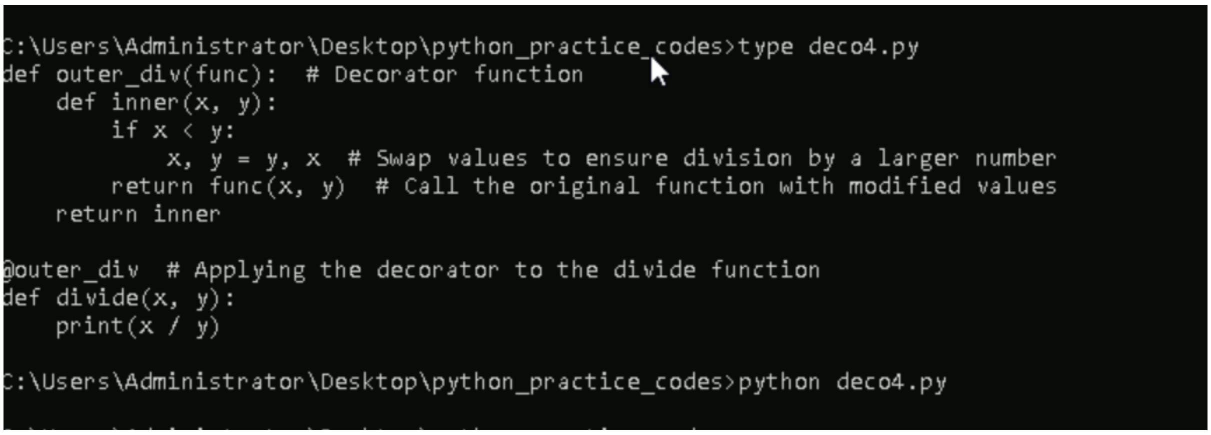
code and output:



```
:\Users\Administrator\Desktop\python_practice_codes>type deco3.py
lef divide(x, y):  # Function that performs division
    print(x / y)

lef outer_div(func):  # Decorator function
    def inner(x, y):
        if x < y:
            x, y = y, x  # Swap values to avoid division by a smaller number
        return func(x, y)
    return inner

livide1 = outer_div(divide)  # Wrapping divide with outer_div
livide1(2, 4)  # Calling the modified function

:\Users\Administrator\Desktop\python_practice_codes>python deco3.py
.0
```

Python allows to use decorator in easy way with @symbol

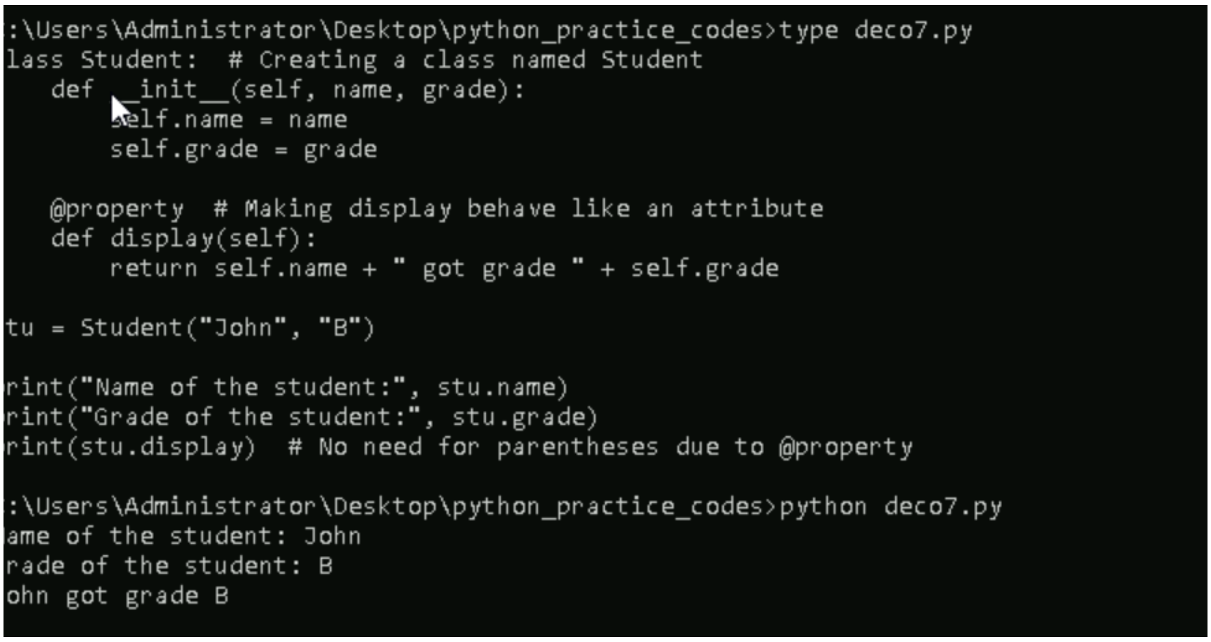
Code and output:



```
:\Users\Administrator\Desktop\python_practice_codes>type deco4.py
lef outer_div(func):  # Decorator function
    def inner(x, y):
        if x < y:
            x, y = y, x  # Swap values to ensure division by a larger number
        return func(x, y)  # Call the original function with modified values
    return inner

outer_div  # Applying the decorator to the divide function
lef divide(x, y):
    print(x / y)

:\Users\Administrator\Desktop\python_practice_codes>python deco4.py
```

@property decorator - By using it, we can use the class function as an attribute

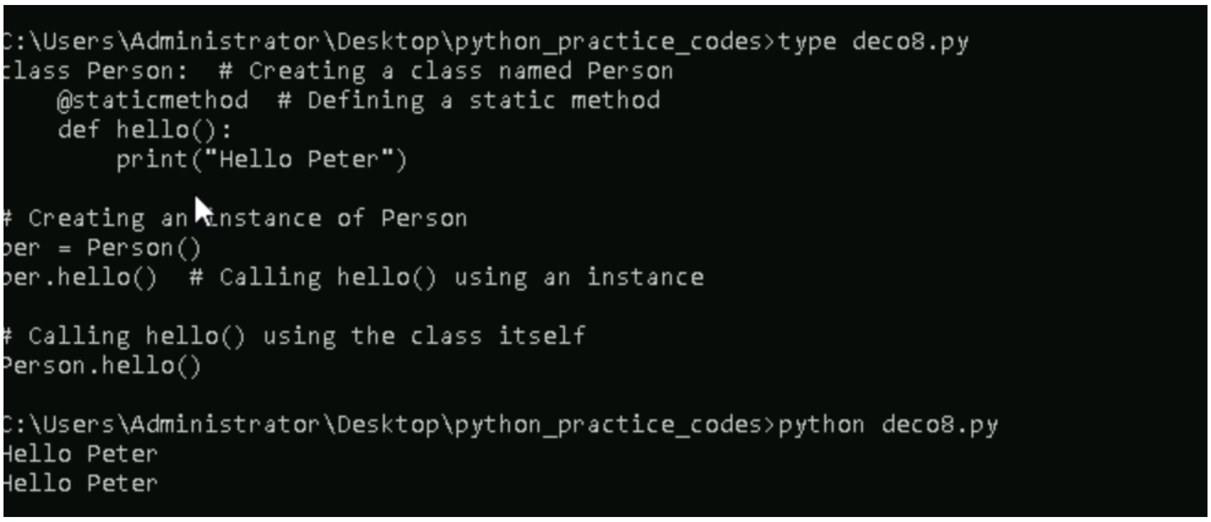
Code and output:



```
:\Users\Administrator\Desktop\python_practice_codes>type deco7.py
lass Student:  # Creating a class named Student
   def __init__(self, name, grade):
       self.name = name
       self.grade = grade

   @property  # Making display behave like an attribute
   def display(self):
       return self.name + " got grade " + self.grade

tu = Student("John", "B")

rint("Name of the student:", stu.name)
rint("Grade of the student:", stu.grade)
rint(stu.display)  # No need for parentheses due to @property

:\Users\Administrator\Desktop\python_practice_codes>python deco7.py
ame of the student: John
rade of the student: B
ohn got grade B
```

@staticmethod decorator- The @staticmethod is used to define a static method in the class

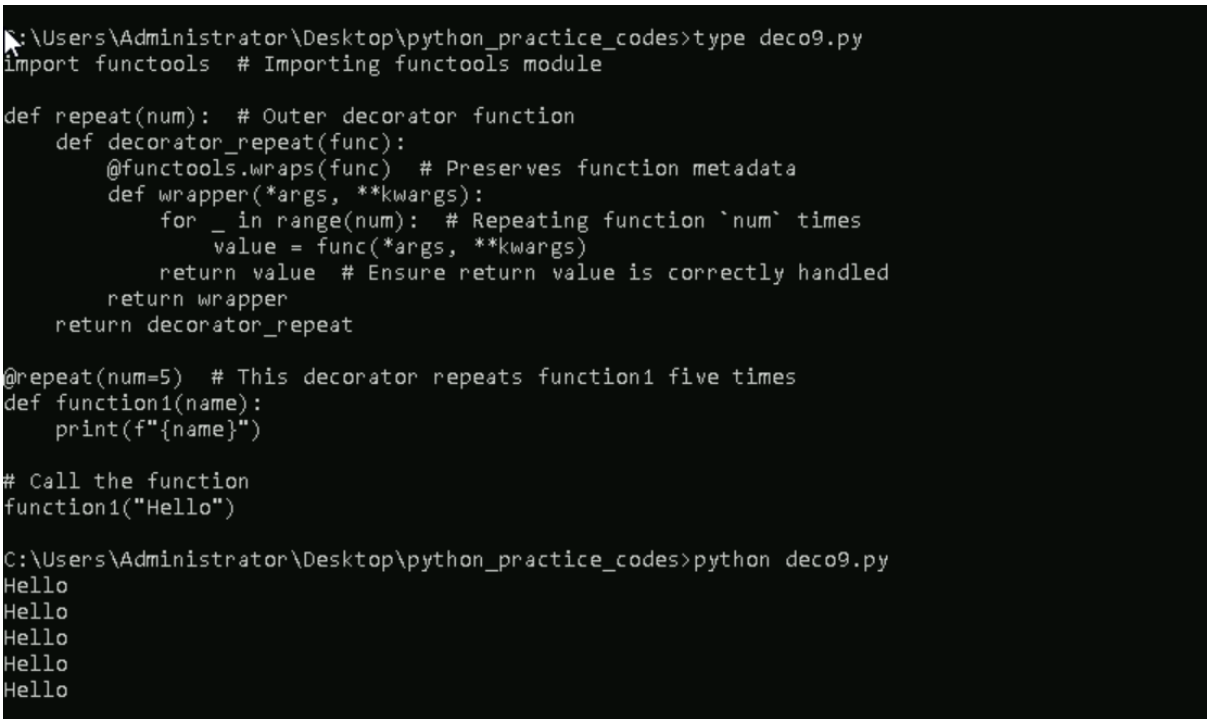
Code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type deco8.py
class Person:  # Creating a class named Person
    @staticmethod  # Defining a static method
    def hello():
        print("Hello Peter")

# Creating an instance of Person
per = Person()
per.hello()  # Calling hello() using an instance

# Calling hello() using the class itself
Person.hello()

C:\Users\Administrator\Desktop\python_practice_codes>python deco8.py
Hello Peter
Hello Peter
```

we are importing the functools into our program

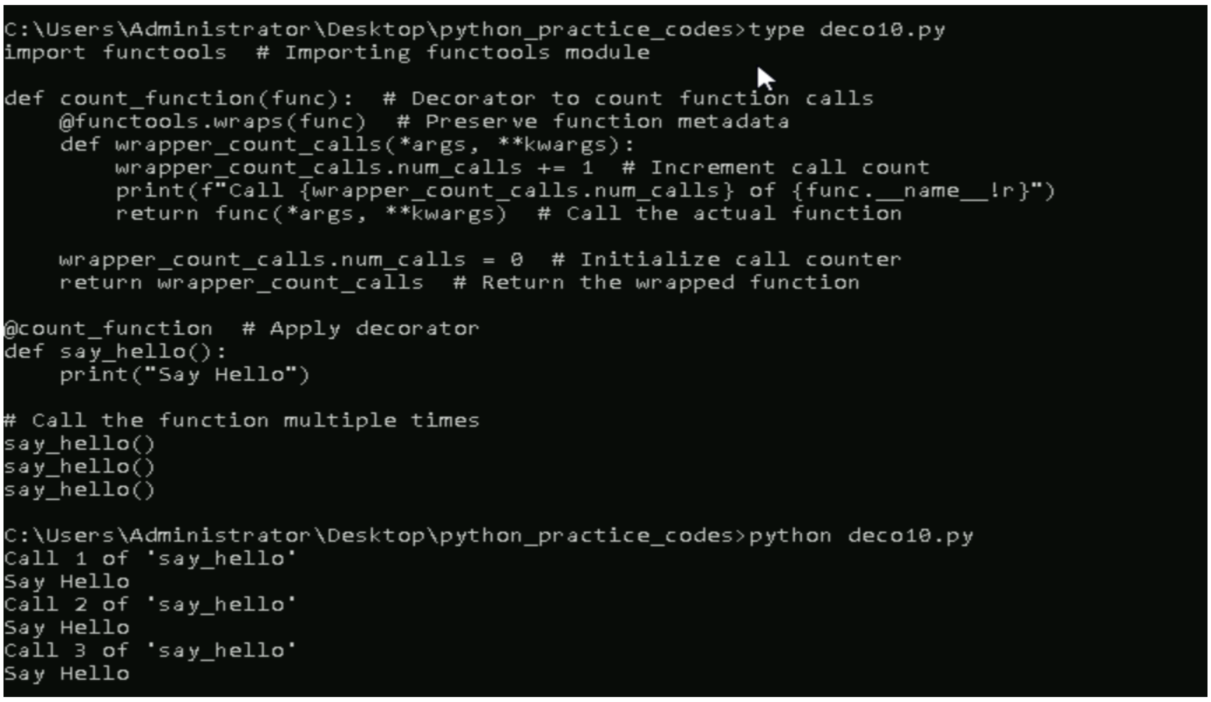
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type deco9.py
import functools  # Importing functools module

def repeat(num):  # Outer decorator function
    def decorator_repeat(func):
        @functools.wraps(func)  # Preserves function metadata
        def wrapper(*args, **kwargs):
            for _ in range(num):  # Repeating function `num` times
                value = func(*args, **kwargs)
            return value  # Ensure return value is correctly handled
        return wrapper
    return decorator_repeat

@repeat(num=5)  # This decorator repeats function1 five times
def function1(name):
    print(f"{name}")

# Call the function
function1("Hello")

C:\Users\Administrator\Desktop\python_practice_codes>python deco9.py
Hello
Hello
Hello
Hello
Hello
```

e example where we are creating a decorator that counts how many times the function has been called

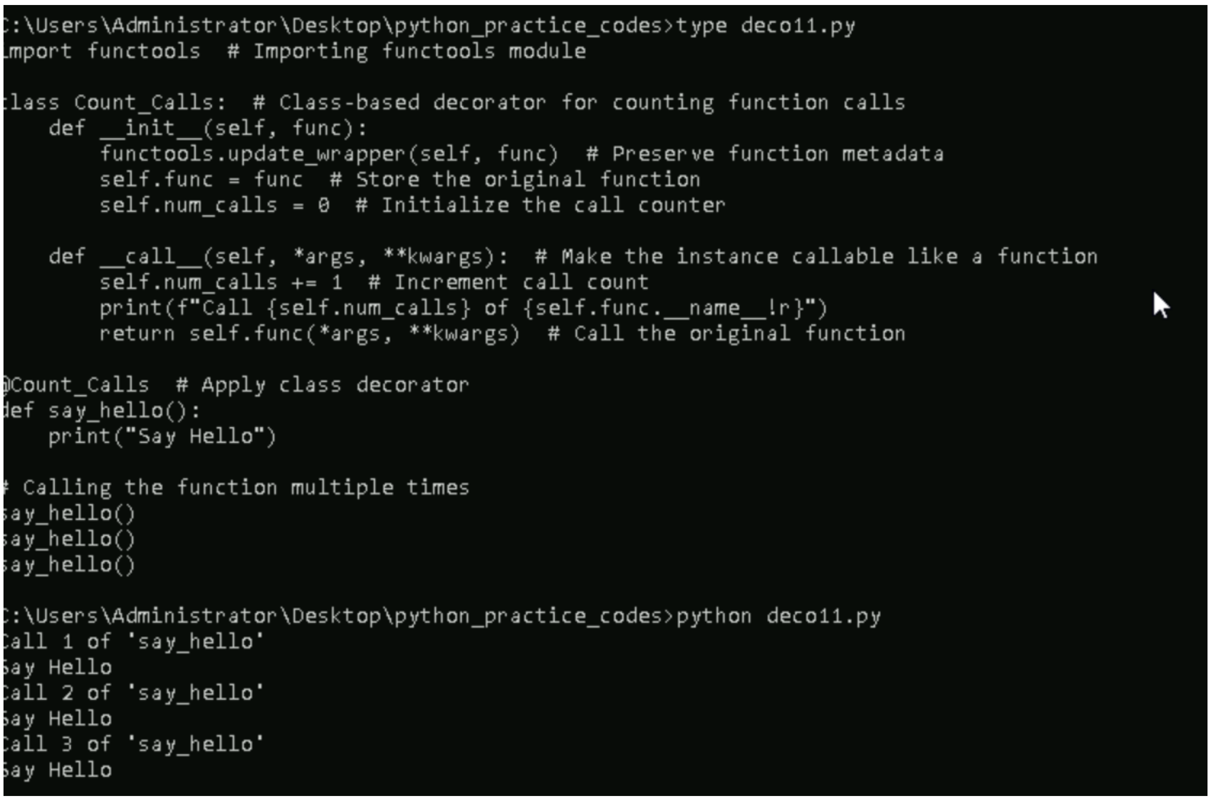
code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type deco10.py
import functools  # Importing functools module

def count_function(func):  # Decorator to count function calls
    @functools.wraps(func)  # Preserve function metadata
    def wrapper_count_calls(*args, **kwargs):
        wrapper_count_calls.num_calls += 1  # Increment call count
        print(f"Call {wrapper_count_calls.num_calls} of {func.__name__!r}")
        return func(*args, **kwargs)  # Call the actual function

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

@count_function  # Apply decorator
def say_hello():
    print("Say Hello")

# Call the function multiple times
say_hello()
say_hello()
say_hello()

C:\Users\Administrator\Desktop\python_practice_codes>python deco10.py
Call 1 of 'say_hello'
Say Hello
Call 2 of 'say_hello'
Say Hello
Call 3 of 'say_hello'
Say Hello
```

create a class that contains __init__() and take func as an argument

code and output:

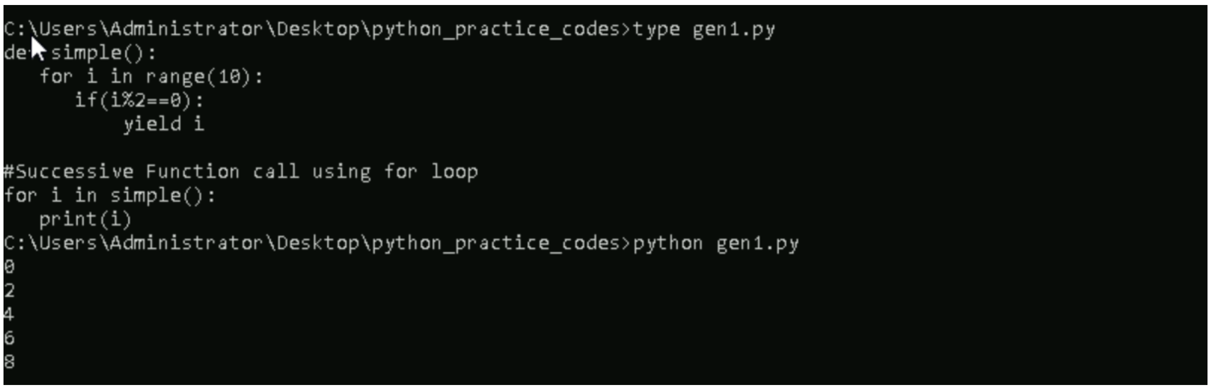


```
:\Users\Administrator\Desktop\python_practice_codes>type deco11.py
mport functools  # Importing functools module

lass Count_Calls:  # Class-based decorator for counting function calls
    def __init__(self, func):
        functools.update_wrapper(self, func)  # Preserve function metadata
        self.func = func  # Store the original function
        self.num_calls = 0  # Initialize the call counter

    def __call__(self, *args, **kwargs):  # Make the instance callable like a function
        self.num_calls += 1  # Increment call count
        print(f"Call {self.num_calls} of {self.func.__name__!r}")
        return self.func(*args, **kwargs)  # Call the original function

Count_Calls  # Apply class decorator
ef say_hello():
    print("Say Hello")

 Calling the function multiple times
ay_hello()
ay_hello()
ay_hello()

:\Users\Administrator\Desktop\python_practice_codes>python deco11.py
all 1 of 'say_hello'
ay Hello
all 2 of 'say_hello'
ay Hello
all 3 of 'say_hello'
ay Hello
```

Python generators:

Create Generator function in Python

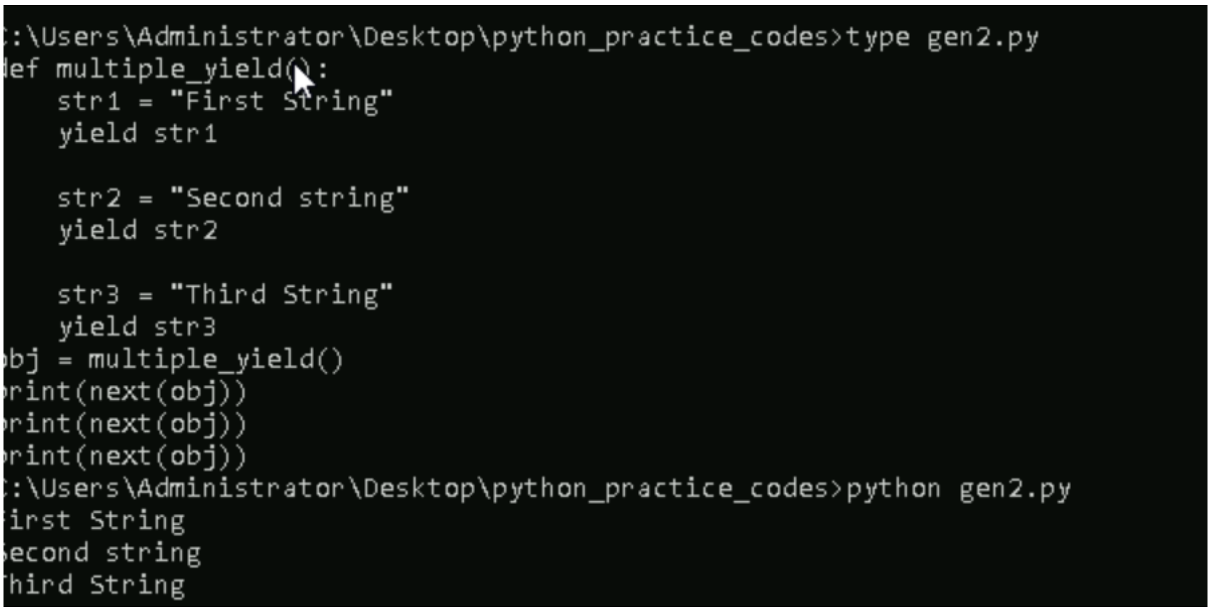
Code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type gen1.py
def simple():
   for i in range(10):
      if(i%2==0):
          yield i

#Successive Function call using for loop
for i in simple():
   print(i)
C:\Users\Administrator\Desktop\python_practice_codes>python gen1.py
0
2
4
6
8
```

Using multiple yield Statement

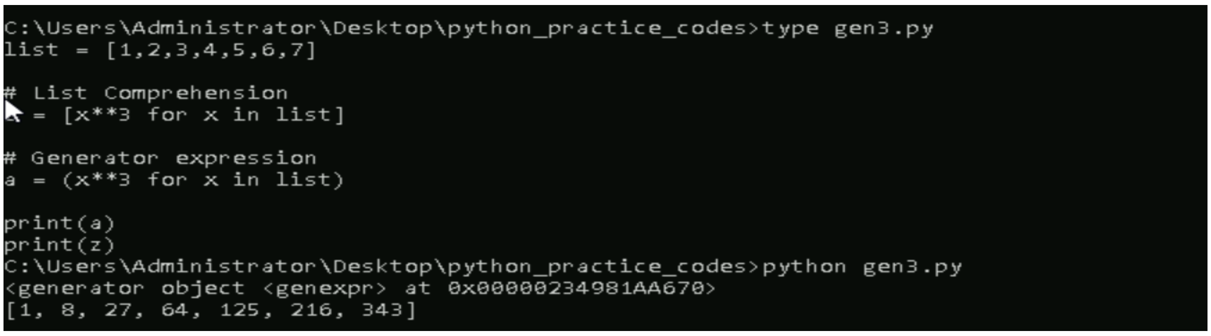
Code and output:



```
:\Users\Administrator\Desktop\python_practice_codes>type gen2.py
ef multiple_yield():
   str1 = "First String"
   yield str1

   str2 = "Second string"
   yield str2

   str3 = "Third String"
   yield str3
bj = multiple_yield()
rint(next(obj))
rint(next(obj))
rint(next(obj))
:\Users\Administrator\Desktop\python_practice_codes>python gen2.py
irst String
econd string
hird String
```

Difference between Generator function and Normal function

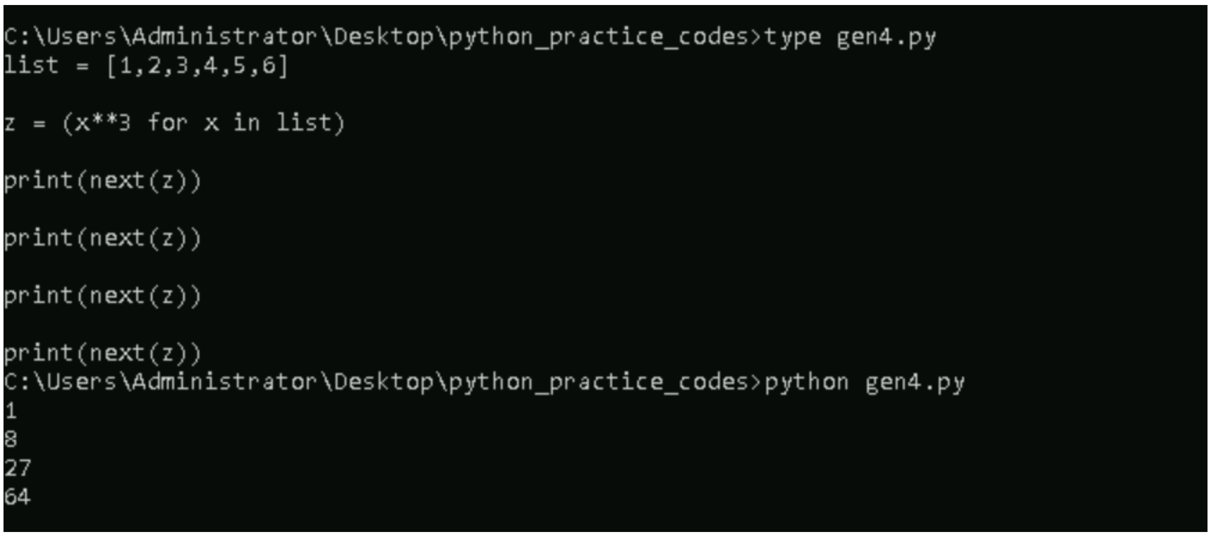
Code and output:



```
C:\Users\Administrator\Desktop\python_practice_codes>type gen3.py
list = [1,2,3,4,5,6,7]

# List Comprehension
z = [x**3 for x in list]

# Generator expression
a = (x**3 for x in list)

print(a)
print(z)
C:\Users\Administrator\Desktop\python_practice_codes>python gen3.py
<generator object <genexpr> at 0x00000234981AA670>
[1, 8, 27, 64, 125, 216, 343]
```

Using next()

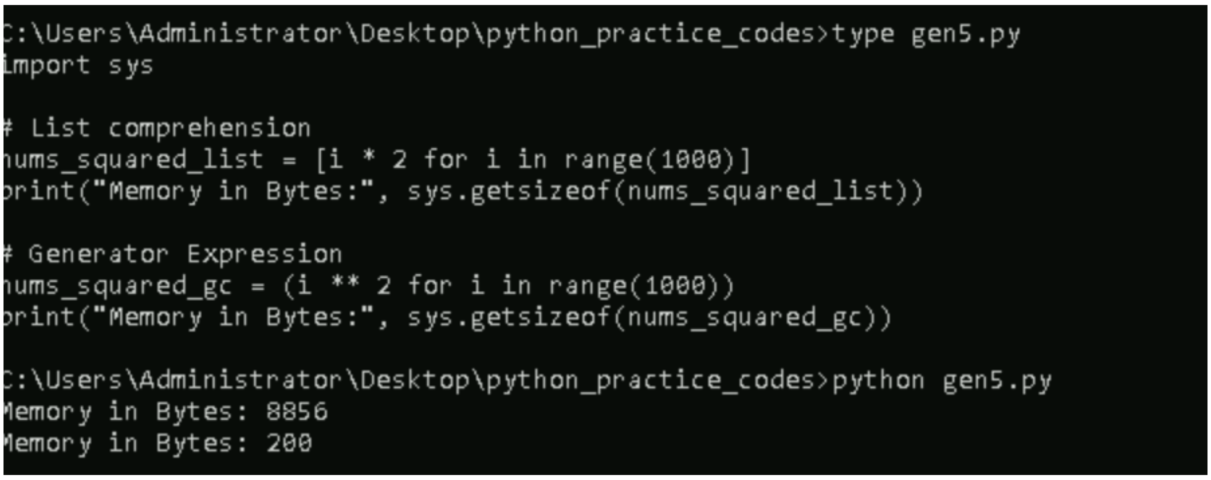
Code and output:



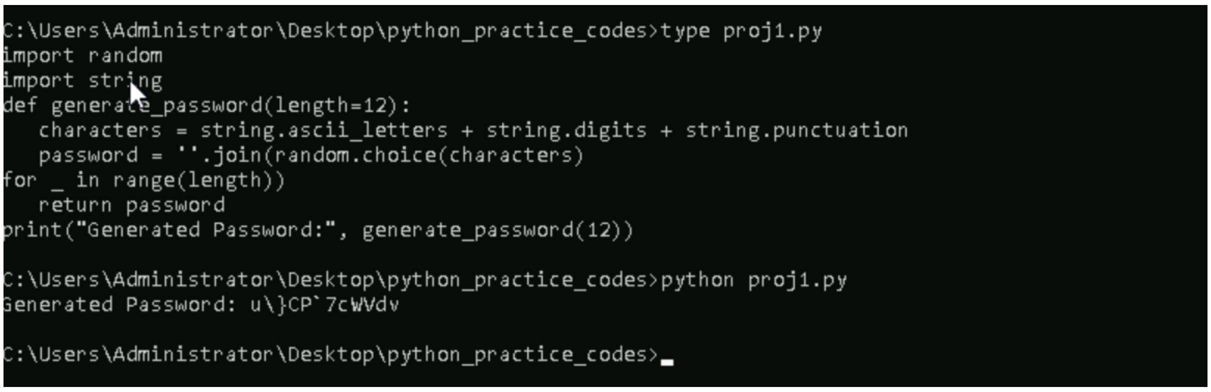
```
C:\Users\Administrator\Desktop\python_practice_codes>type gen4.py
list = [1,2,3,4,5,6]

z = (x**3 for x in list)

print(next(z))

print(next(z))

print(next(z))

print(next(z))
C:\Users\Administrator\Desktop\python_practice_codes>python gen4.py
1
8
27
64
```

Using sys.getsizeof() with generators to get the memory

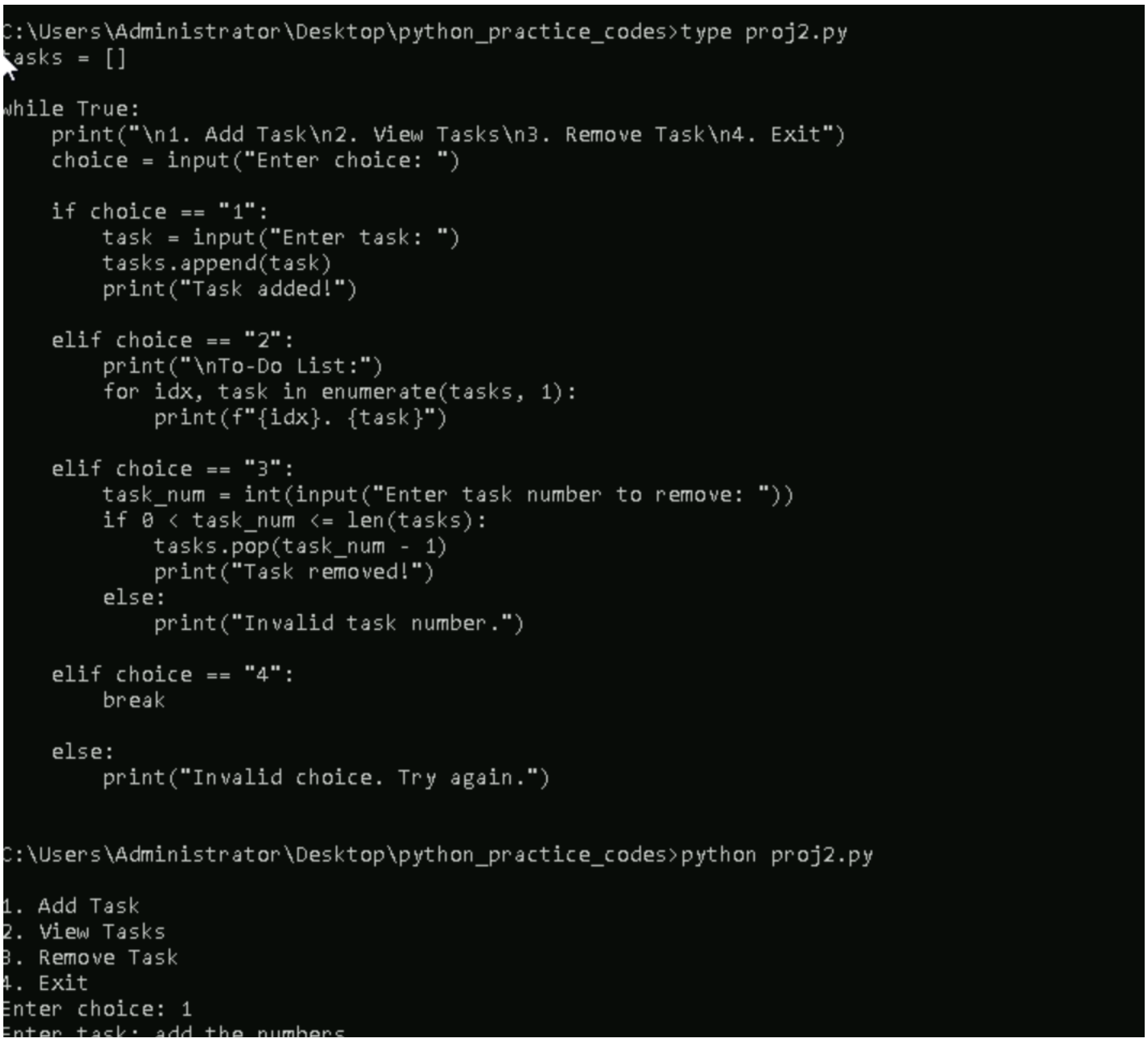
Code and output:



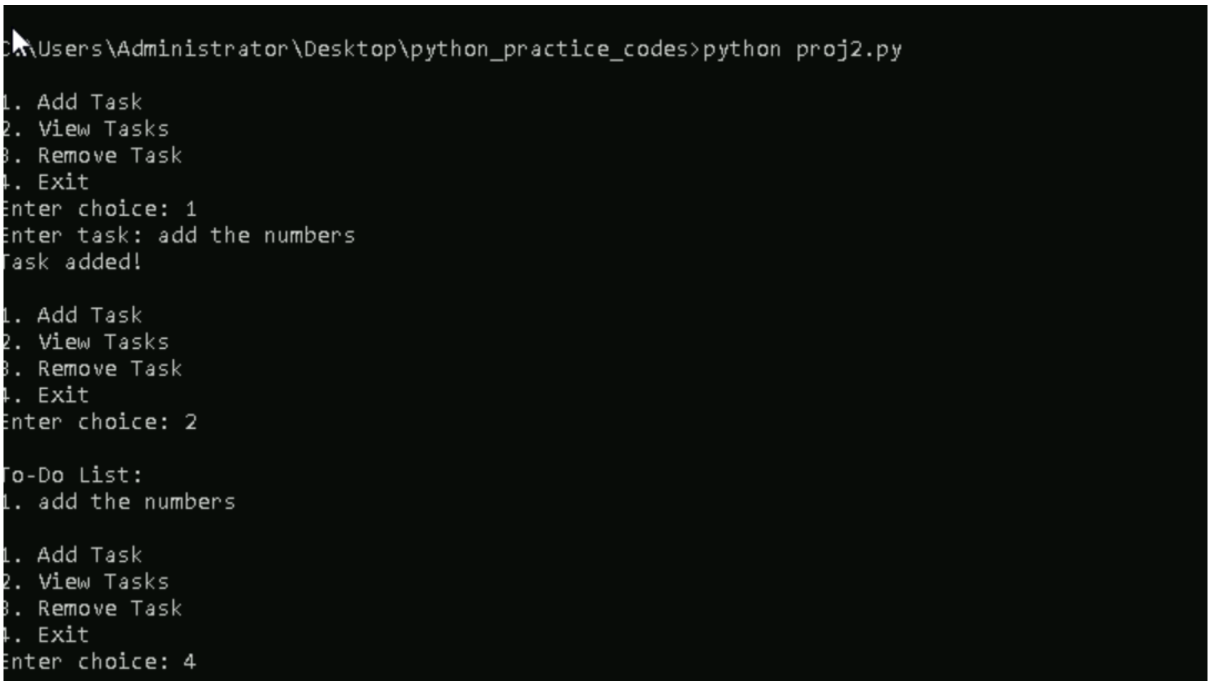
```
C:\Users\Administrator\Desktop\python_practice_codes>type gen5.py
import sys

# List comprehension
nums_squared_list = [i * 2 for i in range(1000)]
print("Memory in Bytes:", sys.getsizeof(nums_squared_list))

# Generator Expression
nums_squared_gc = (i ** 2 for i in range(1000))
print("Memory in Bytes:", sys.getsizeof(nums_squared_gc))

C:\Users\Administrator\Desktop\python_practice_codes>python gen5.py
Memory in Bytes: 8856
Memory in Bytes: 200
```



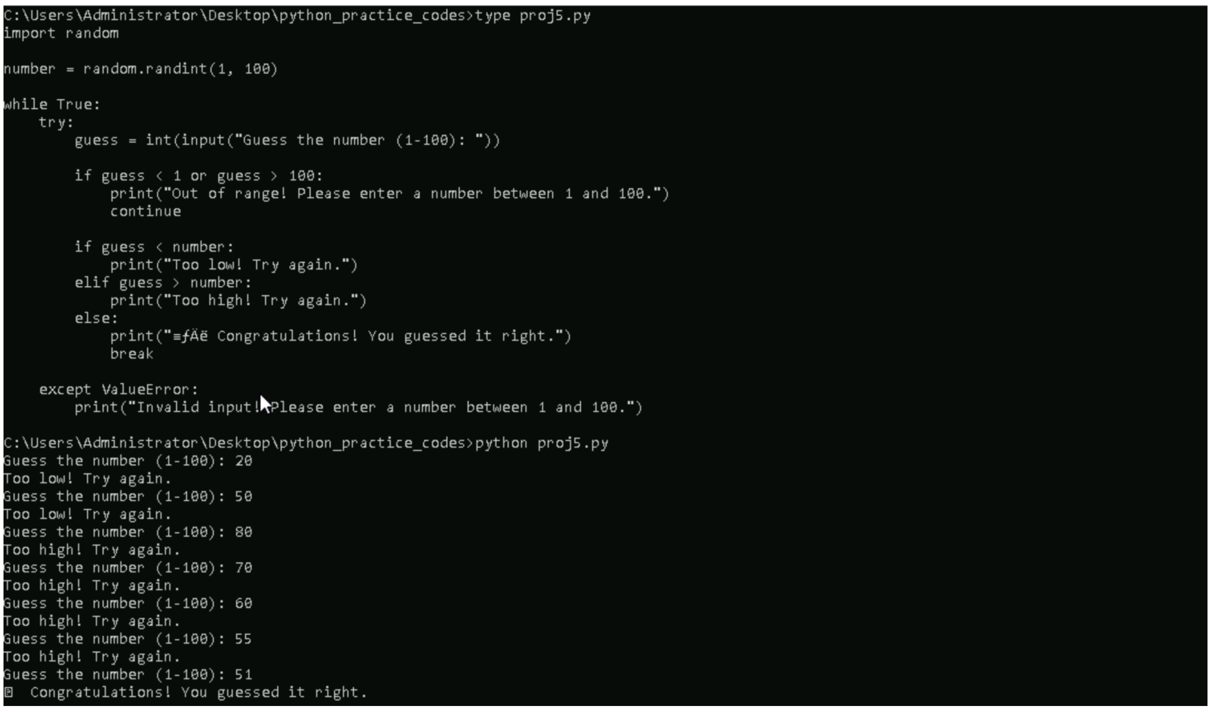
```
C:\Users\Administrator\Desktop\python_practice_codes>type proj1.py
import random
import string
def generate_password(length=12):
   characters = string.ascii_letters + string.digits + string.punctuation
   password = ''.join(random.choice(characters)
for _ in range(length))
   return password
print("Generated Password:", generate_password(12))

C:\Users\Administrator\Desktop\python_practice_codes>python proj1.py
Generated Password: u\}CP`7cWVdv

C:\Users\Administrator\Desktop\python_practice_codes>_
```



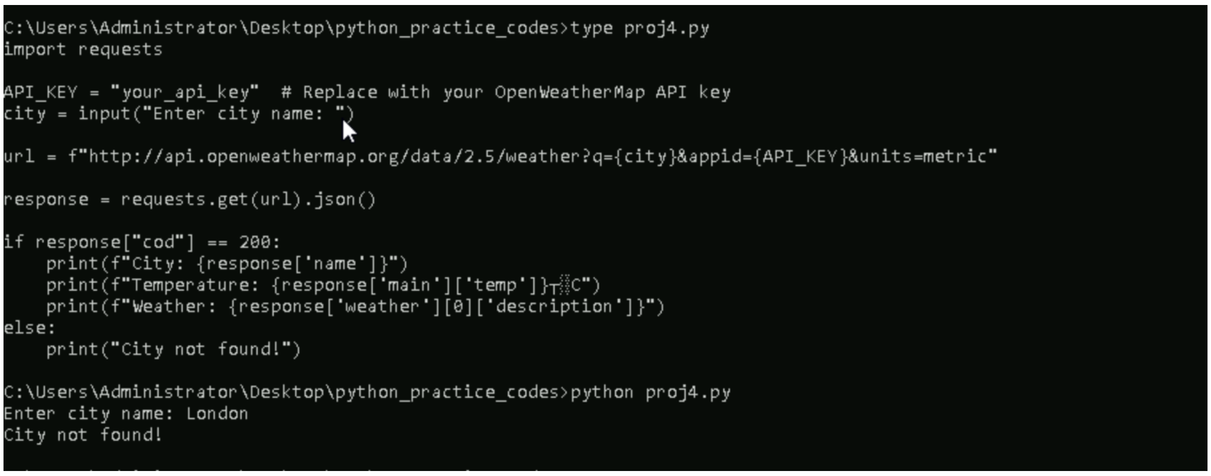
```
C:\Users\Administrator\Desktop\python_practice_codes>type proj2.py
tasks = []

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!")
        else:
            print("Invalid task number.")

    elif choice == "4":
        break

    else:
        print("Invalid choice. Try again.")


C:\Users\Administrator\Desktop\python_practice_codes>python proj2.py

1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Enter choice: 1
Enter task: add the numbers
```



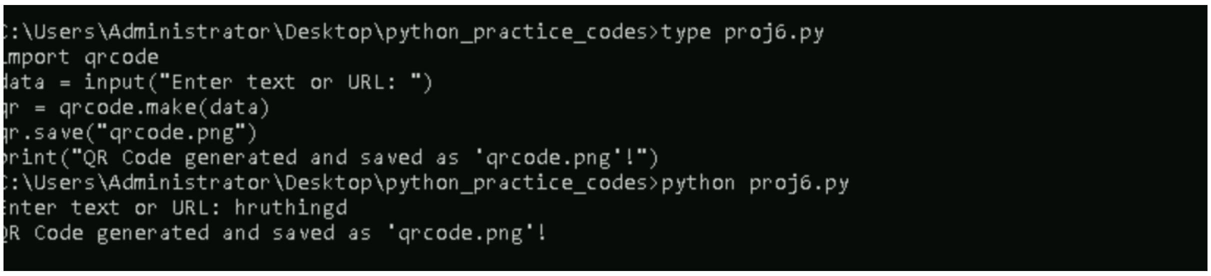
```
C:\Users\Administrator\Desktop\python_practice_codes>python proj2.py

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

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

To-Do List:
1. add the numbers

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



```
C:\Users\Administrator\Desktop\python_practice_codes>type proj5.py
import random

number = random.randint(1, 100)

while True:
    try:
        guess = int(input("Guess the number (1-100): "))

        if guess < 1 or guess > 100:
            print("Out of range! Please enter a number between 1 and 100.")
            continue

        if guess < number:
            print("Too low! Try again.")
        elif guess > number:
            print("Too high! Try again.")
        else:
            print("≡ƒÄë Congratulations! You guessed it right.")
            break

    except ValueError:
        print("Invalid input! Please enter a number between 1 and 100.")

C:\Users\Administrator\Desktop\python_practice_codes>python proj5.py
Guess the number (1-100): 20
Too low! Try again.
Guess the number (1-100): 50
Too low! Try again.
Guess the number (1-100): 80
Too high! Try again.
Guess the number (1-100): 70
Too high! Try again.
Guess the number (1-100): 60
Too high! Try again.
Guess the number (1-100): 55
Too high! Try again.
Guess the number (1-100): 51
⍰  Congratulations! You guessed it right.
```



```
C:\Users\Administrator\Desktop\python_practice_codes>type proj4.py
import requests

API_KEY = "your_api_key"  # Replace with your OpenWeatherMap API key
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"City: {response['name']}")
    print(f"Temperature: {response['main']['temp']}┬░C")
    print(f"Weather: {response['weather'][0]['description']}")
else:
    print("City not found!")

C:\Users\Administrator\Desktop\python_practice_codes>python proj4.py
Enter city name: London
City not found!
```



```
:\Users\Administrator\Desktop\python_practice_codes>type proj6.py
mport qrcode
lata = input("Enter text or URL: ")
r = qrcode.make(data)
r.save("qrcode.png")
rint("QR Code generated and saved as 'qrcode.png'!")
:\Users\Administrator\Desktop\python_practice_codes>python proj6.py
nter text or URL: hruthingd
R Code generated and saved as 'qrcode.png'!
```