PRACTICAL NO. 3:
WRITE SIMPLE PYTHON PROGRAM USING OPERATORS: ARITHMETIC OPERATORS, LOGICAL OPERATORS, BITWISE OPERATORS

ARITHEMTIC OPERATORS

These operators are used to perform mathematical operations and are often known as arithmetical operators. They include addition, subtraction, division, multiplication, floor division and exponential calculation. They are binary operators except for increment/decrement operators that are otherwise unary

EXAMPLE

```
'''PRACTICAL NO. 3: WRITE SIMPLE PYTHON PROGRAM USING OPERATORS: ARITHMETIC
OPERATORS, LOGICAL OPERATORS, BITWISE OPERATORS '''

# ARITHMETIC OPERATORS
a = 10
b = 5
print("Addition is : ", (a + b))
print("Subtraction is : ", (a - b))
print("Multiplication is : ", (a * b))
print("Division is : ", (a / b))
print("Exponential  is : ", (a ** b))
print("Floor Division is : ", (a // b))
```

```
main
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmP
Addition is :  15
Subtraction is :  5
Multiplication is :  50
Division is :  2.0
Exponential  is :  9765625
Floor Division is :  2
```

LOGICAL OPERATORS

These operators are used to combine relational operators. According to the evaluation of their expression, their result is in true or false form

Logical operators include AND, OR and NOT

EXAMPLE

```
# LOGICAL OPERATORS

a = True
b = False
c = True
print("Logical AND : True and False  ", (a and b))
print("Logical AND : True and True  ", (a and c))
print("Logical OR : True or False  ", (a or b))
print("Logical OR : True or True  ", (a or c))
print("Logical NOT :  not True ", (not a))
print("Logical NOT: not False  ", (not b))
```

```
main
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProjects
Logical AND : True and False   False
Logical AND : True and True   True
Logical OR : True or False   True
Logical OR : True or True   True
Logical NOT :  not True  False
Logical NOT: not False   True

Process finished with exit code 0
```

BITWISE OPERATORS

They are used to perform bit by bit operation. They can be both unary and binary. First the number is converted into bits and then operation is directly performed upon them .They include Bitwise AND, Bitwise OR and Bitwise NOT, Bitwise left shift and Right shift

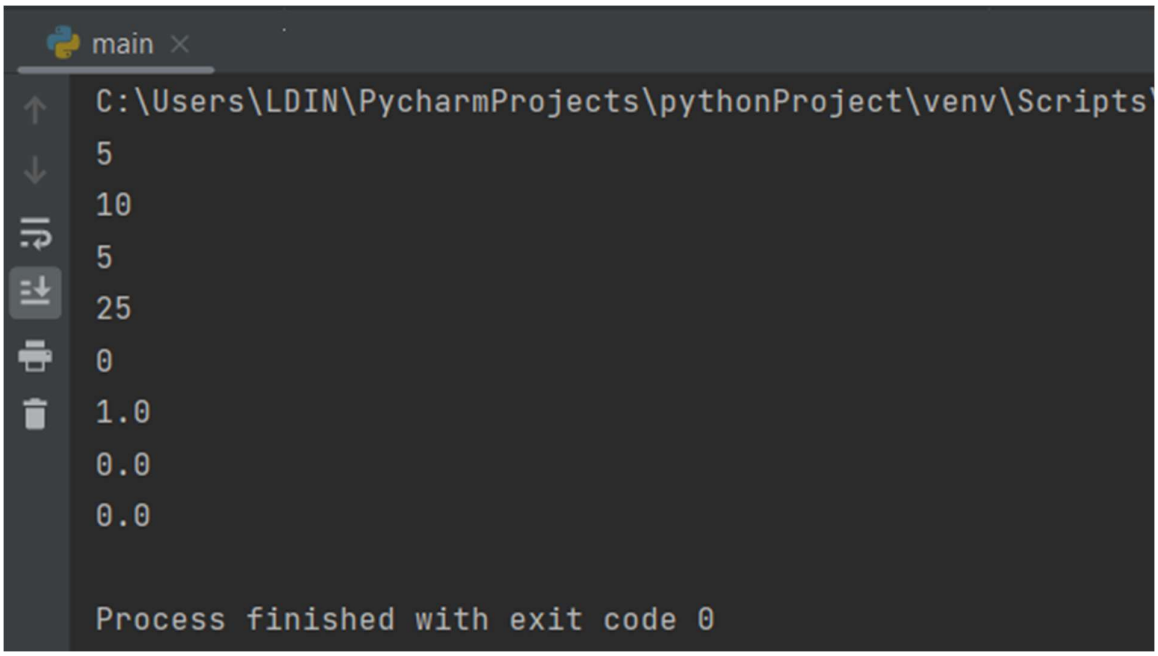
```
# BITWISE OPERATORS

x = 10
y = 4
print("Bitwise AND : ", (x & y))
print("Bitwise OR : ", (x | y))
print("Bitwise NOT: ", (~x))
print("Bitwise XOR : ", (x ^ y))
print("Bitwise RIGHT SHIFT : ", (x >> y))
print("Bitwise LEFT SHIFT : ", (x << y))
```

```
main
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProjects/pythonProject/main
Bitwise AND :  0
Bitwise OR :  14
Bitwise NOT:  -11
Bitwise XOR :  14
Bitwise RIGHT SHIFT :  0
Bitwise LEFT SHIFT :  160

Process finished with exit code 0
```

ASSIGNMENT OPERATORS

These operators are used to assign values to variables. The value to be assigned is at the right side whereas the variable to which value should be assigned is at the left side

```
#ASSIGNMENT OPERATORS

x=5
print(x)
x += 5
print(x)
x -= 5
print(x)
x *= 5
print(x)
x %= 5
print(x)
x /= 5
print(x)
x //= 10
print(x)
x **= 5
print(x)
```



IDENTITY OPERATORS

These operators are used to compare memory address of two objects.

They are used to check whether two values are located at the same part of memory or not

```
# SPECIAL OPERATORS
# IDENTITY OPERATORS
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1, 2, 3]
y3 = [1, 2, 3]
print(x1 is not y1)
print(x2 is y2)
print(x3 is y3)
```

```
main
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProjects/p
False
True
False

Process finished with exit code 0
```

MEMBERSHIP OPERATORS

These operators are used to check existence of a particular element in a sequence

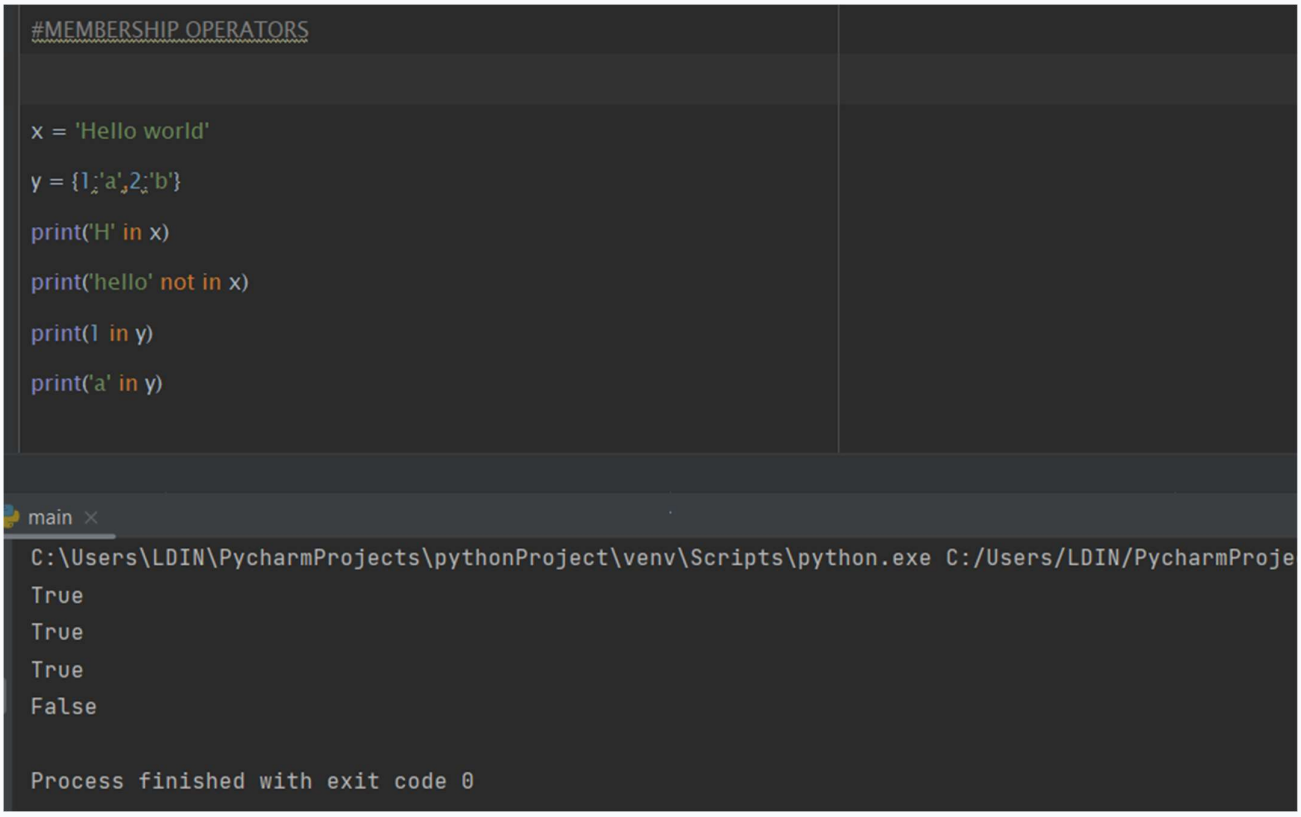
The result is in a Boolean form and the return values like true or false

We can use them in sequences like string, tuple, string, dictionary

It reduces effort of searching an element in the list

They are ‘in’ and ‘not in’

Operator	*Meaning*	*Example*
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x



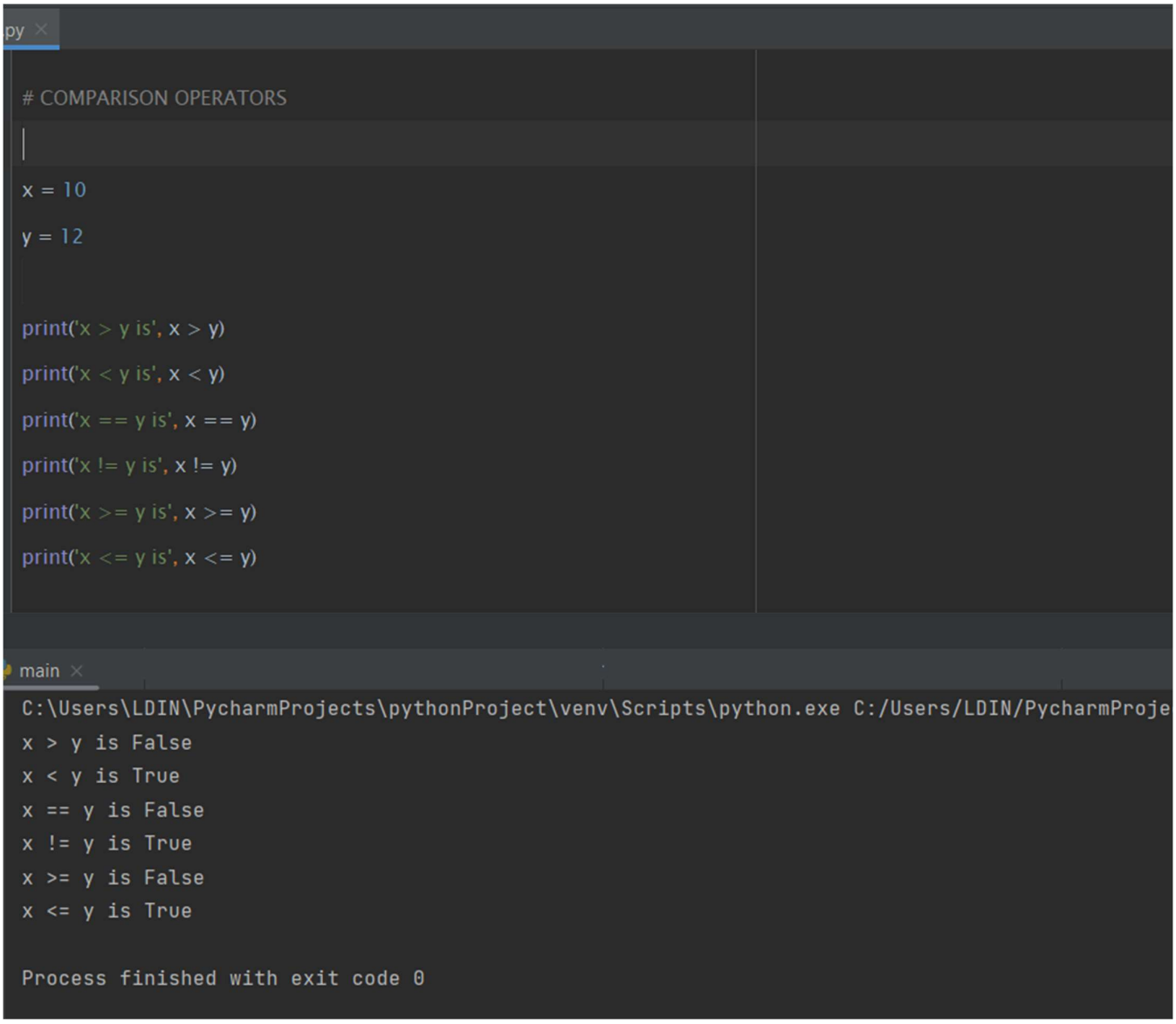
COMPARISON OPERATORS

Also known as Relational Operators, they are used to compare values

According to the evaluation of their expression, the results are returned in a Boolean form as either True or False

They are binary operators meaning the perform operation on binary operands

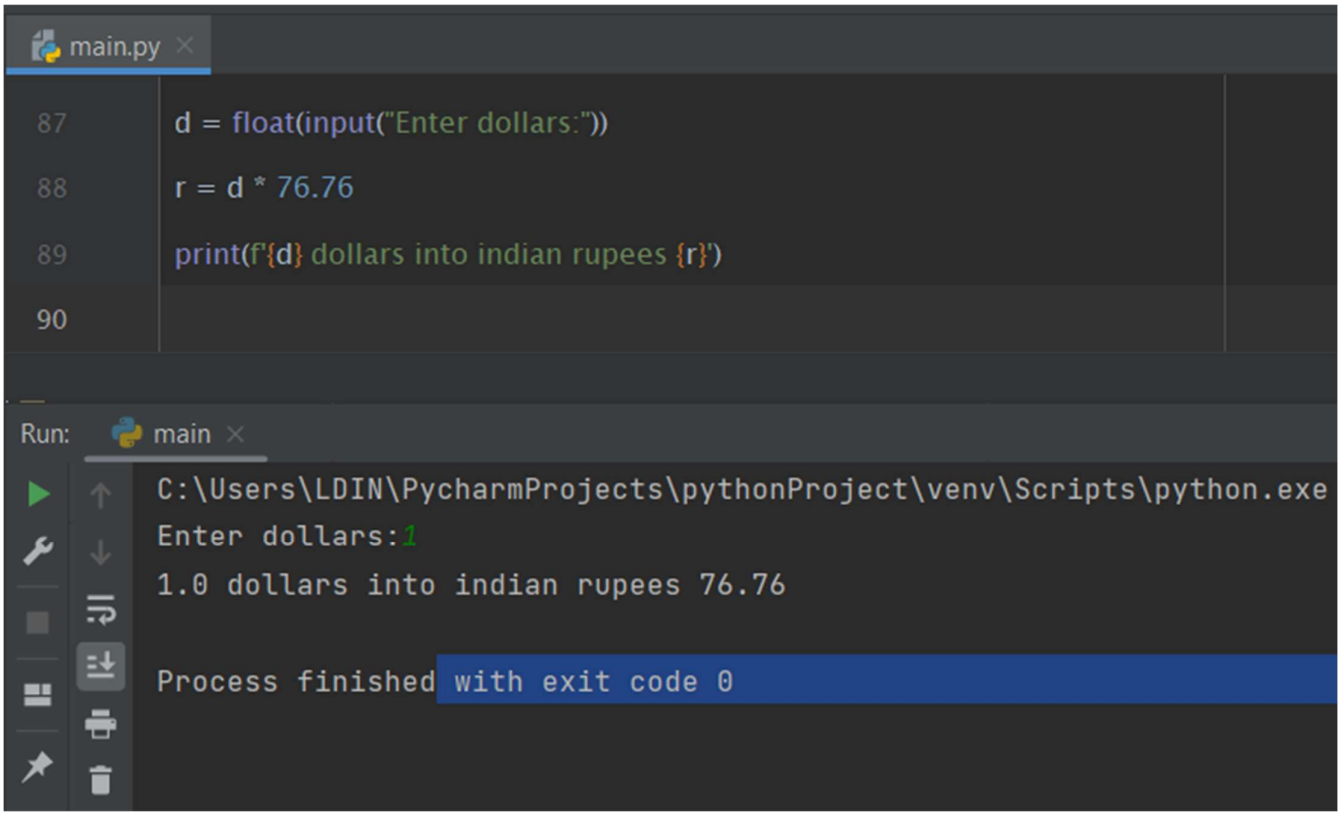
They include greater than, greater than or equal to, less than, less than or equal to, equal to and not Equal to



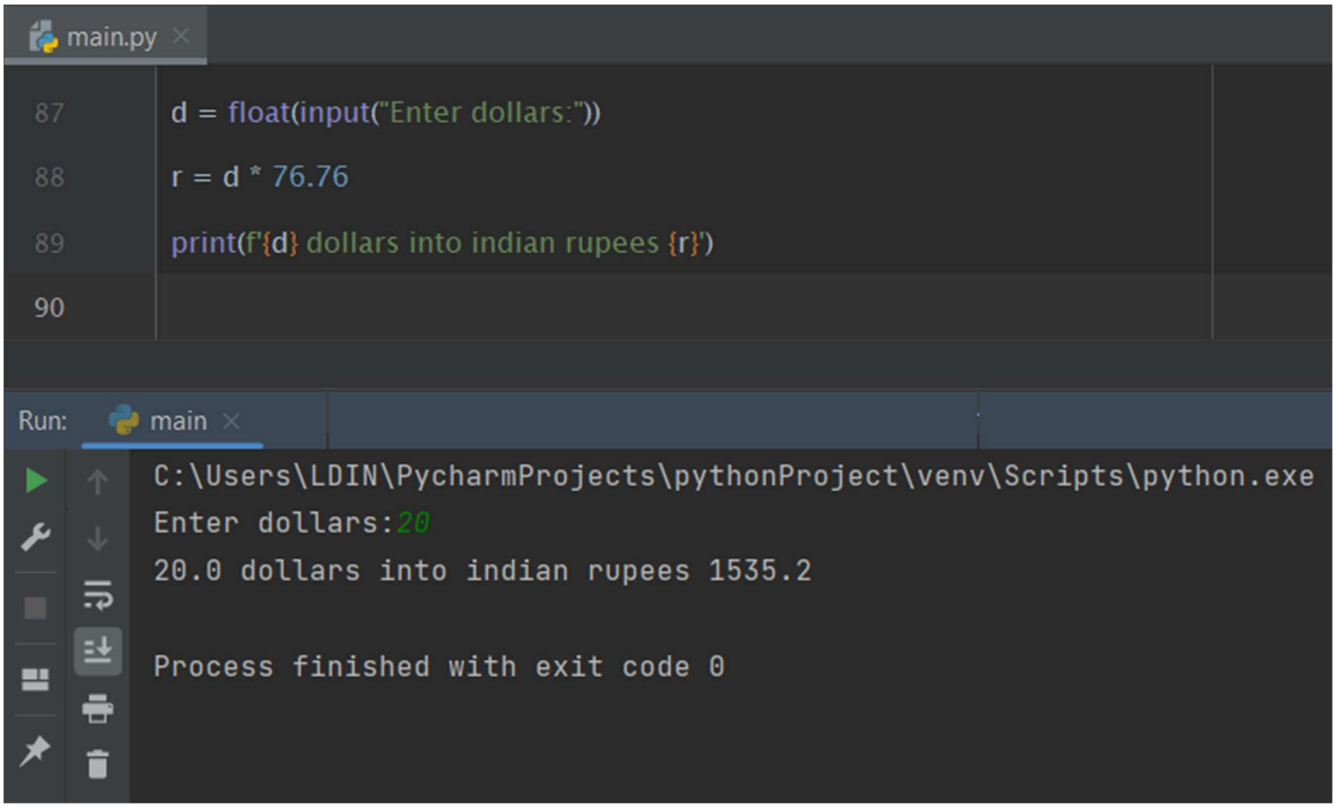
XI EXERCISE

1.WRITE A PROGRAM TO CONVERT U.S. DOLLARS TO INDIAN RUPEES.

OUTPUT 1

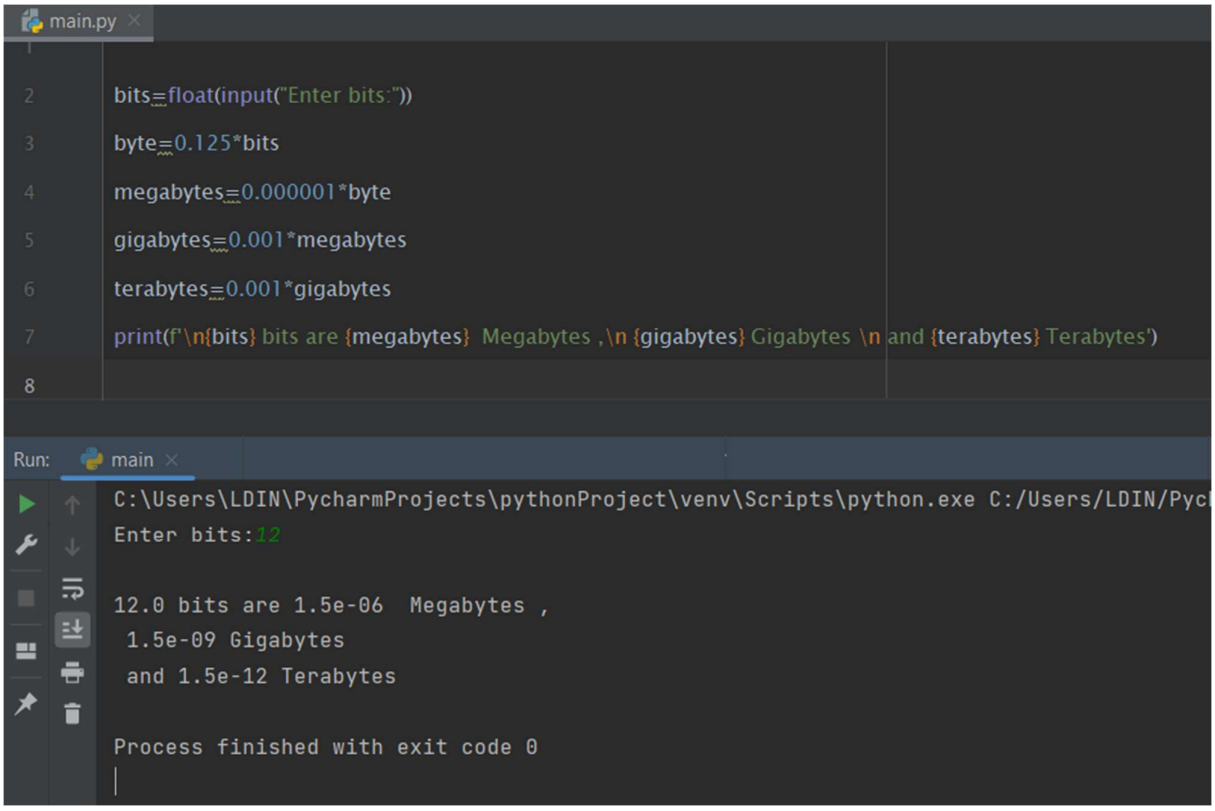


OUTPUT 2

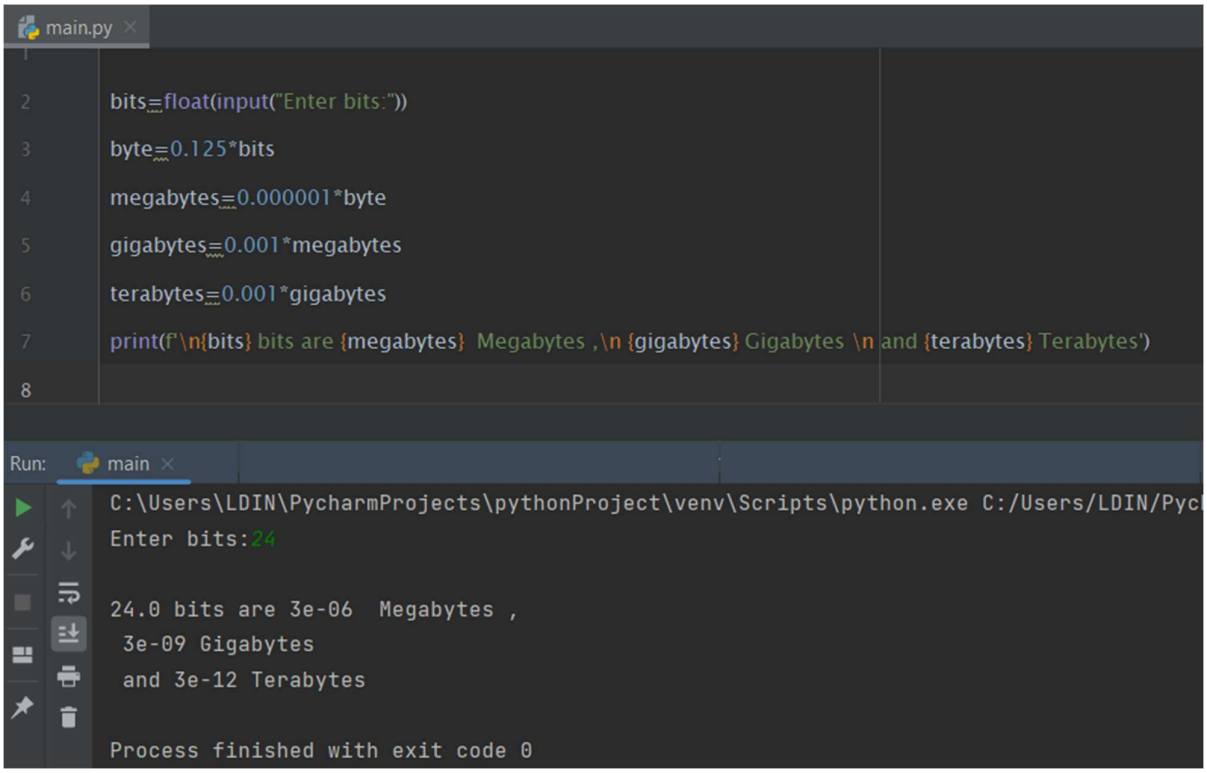


2. WRITE A PROGRAM TO CONVERT BITS TO MEGABYTES, GIGABYTES AND TERABYTES

OUTPUT 1

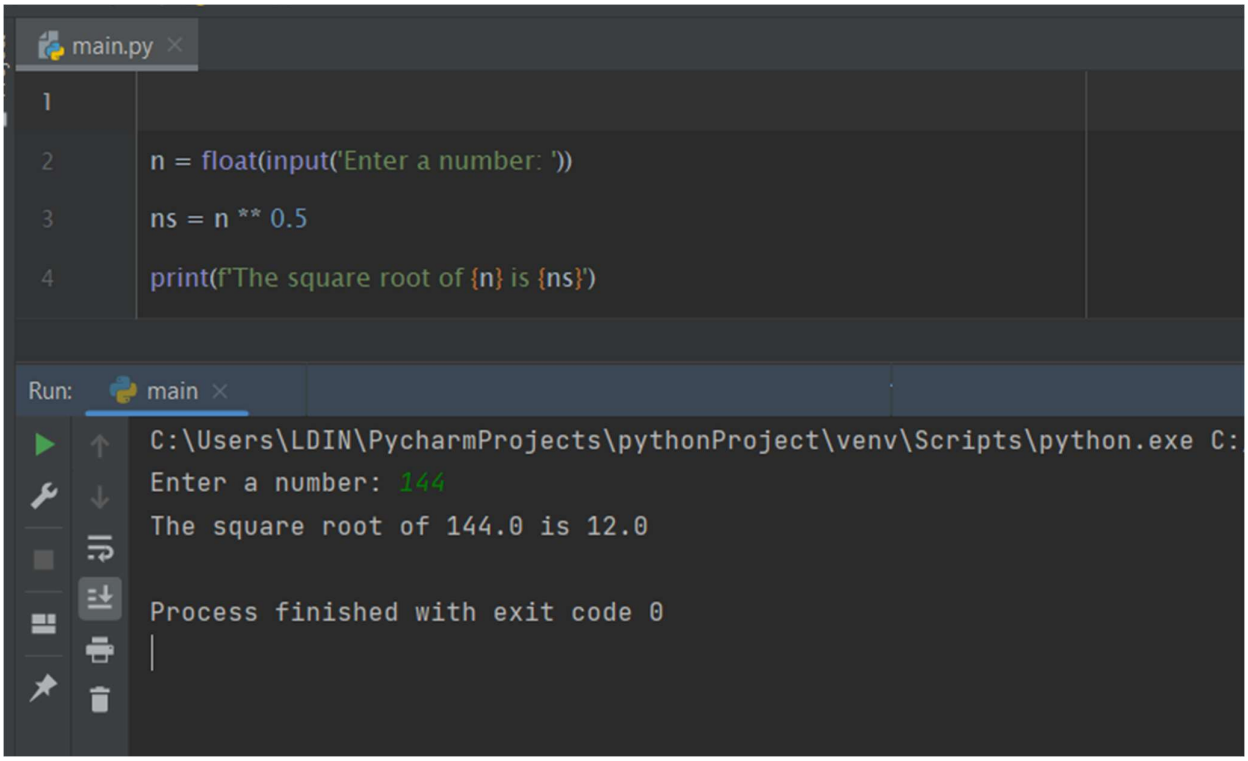


OUTPUT 2

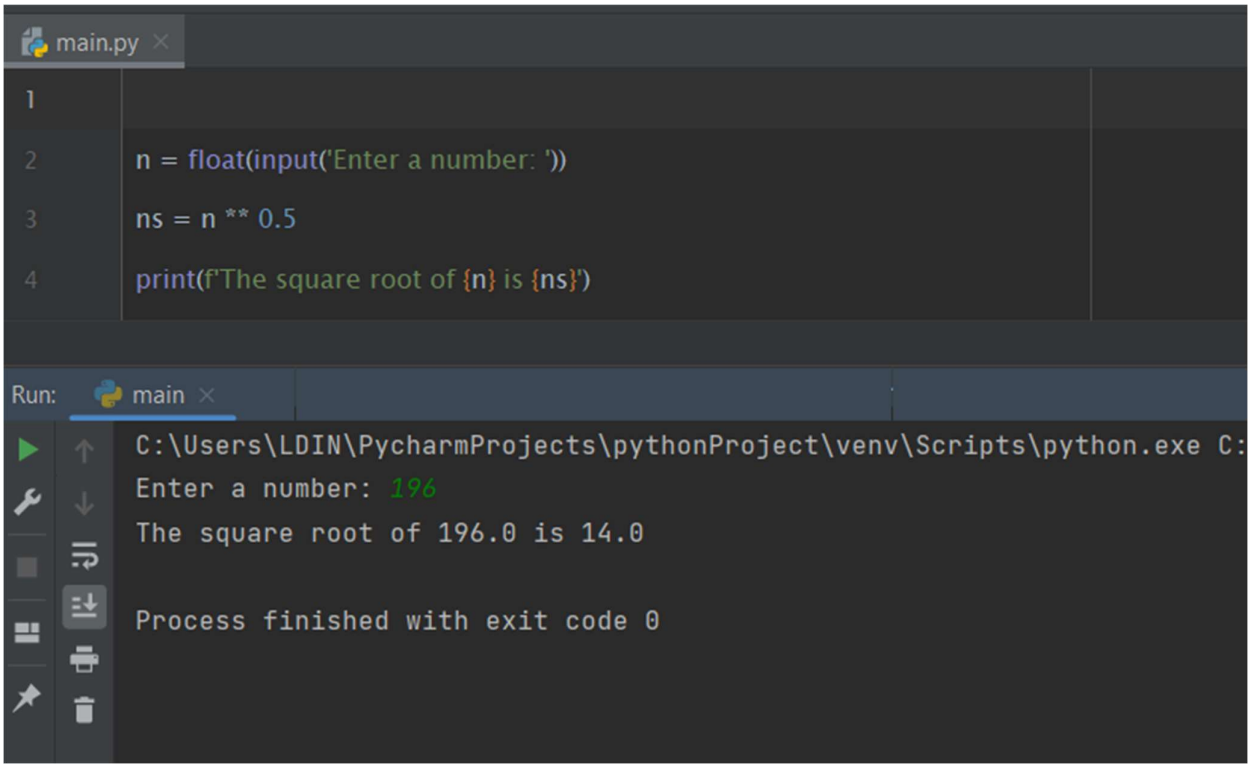


3. WRITE A PROGRAM TO FIND THE SQUARE ROOT OF A NUMBER

OUTPUT 1

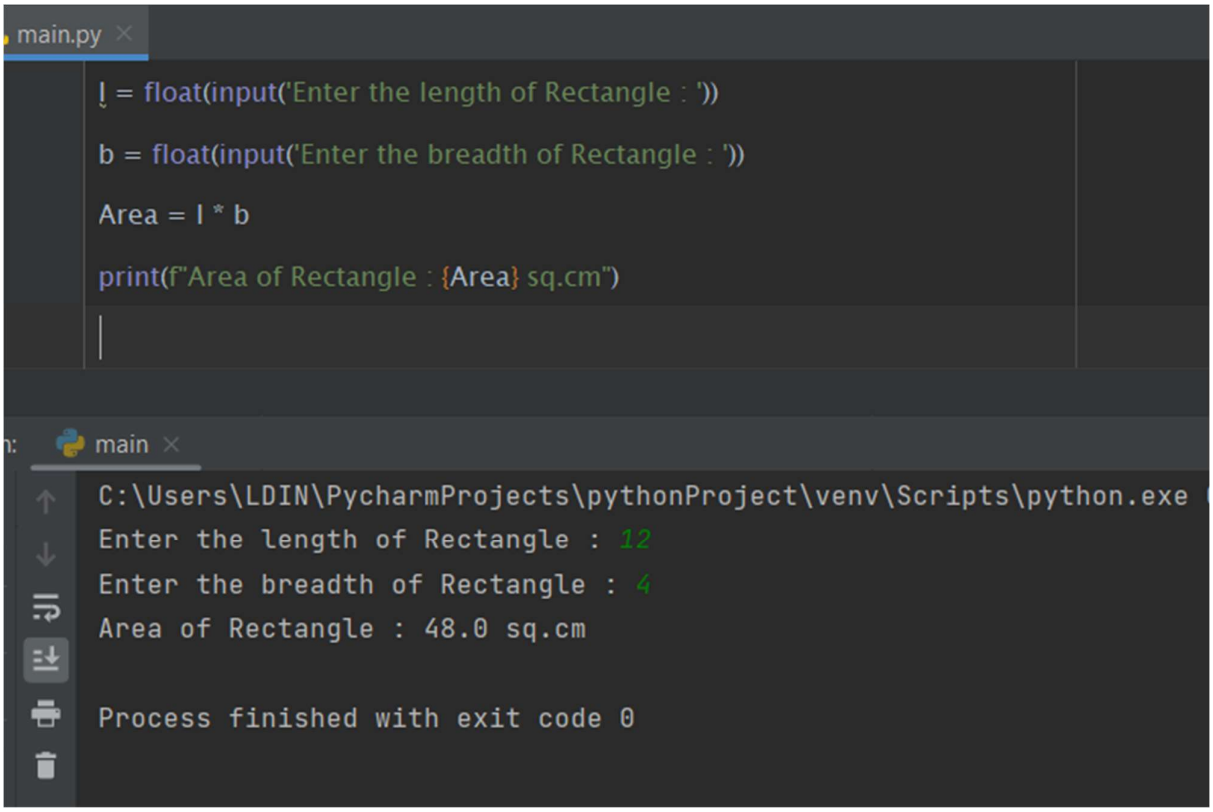


OUTPUT 2

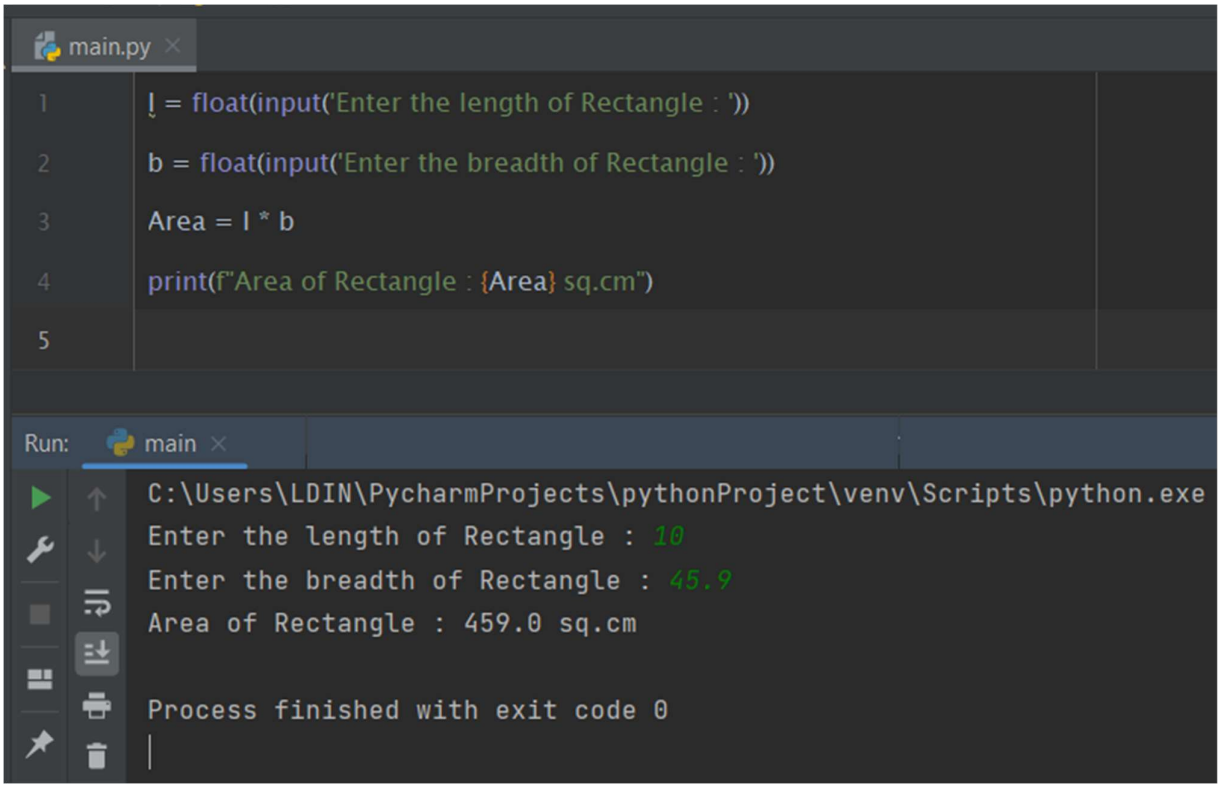


4. WRITE A PROGRAM TO FIND THE AREA OF RECTANGLE

OUTPUT 1

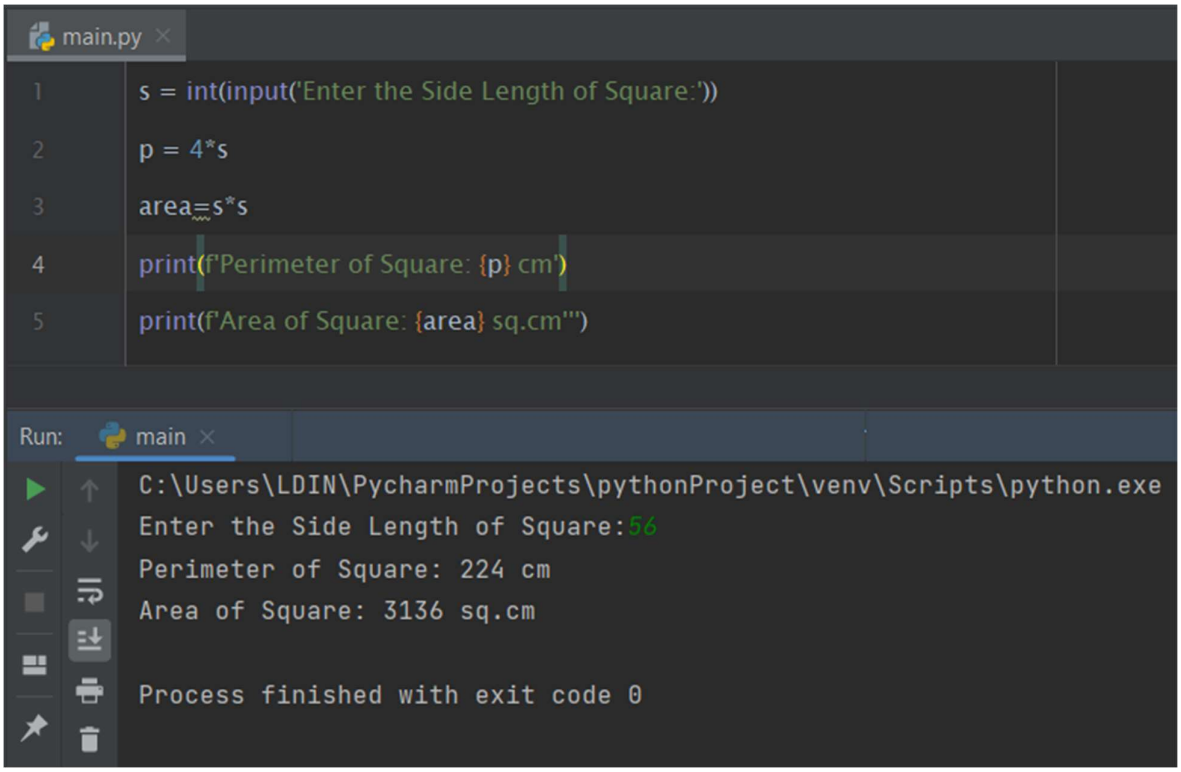


OUTPUT 2

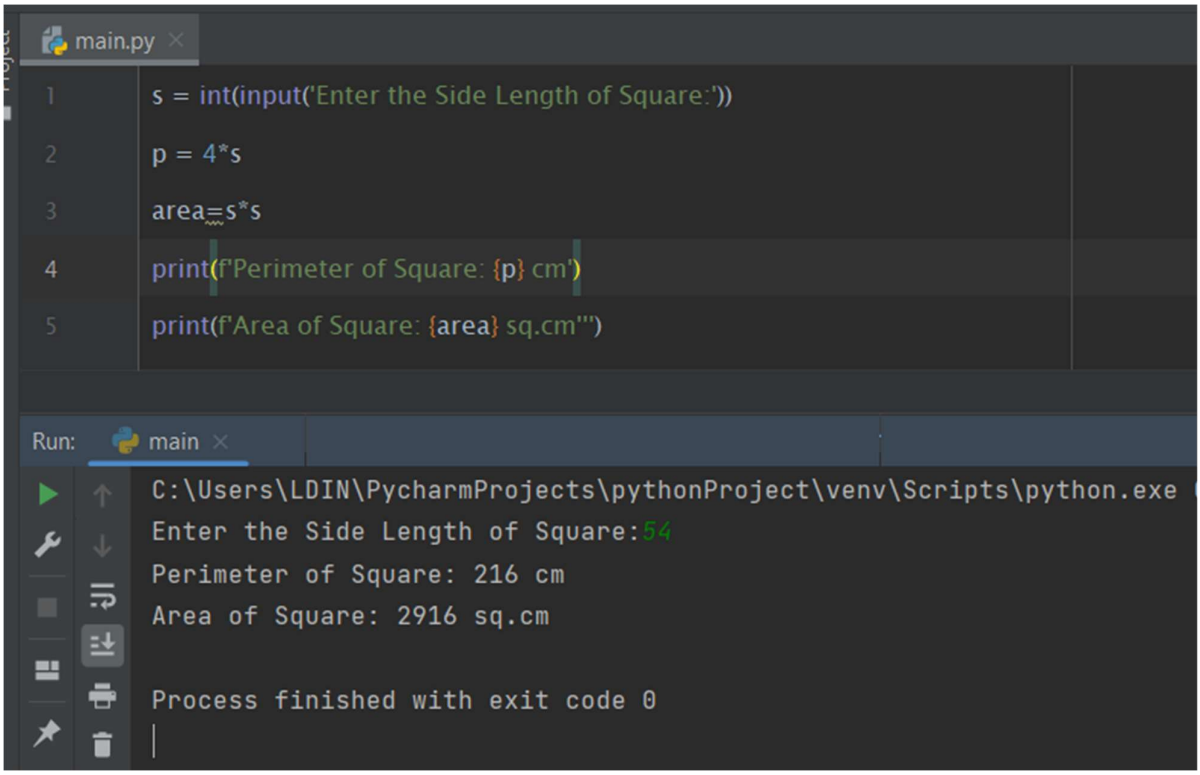


5. WRITE A PROGRAM TO CALCULATE AREA AND PERIMETER OF THE SQUARE

OUTPUT 1

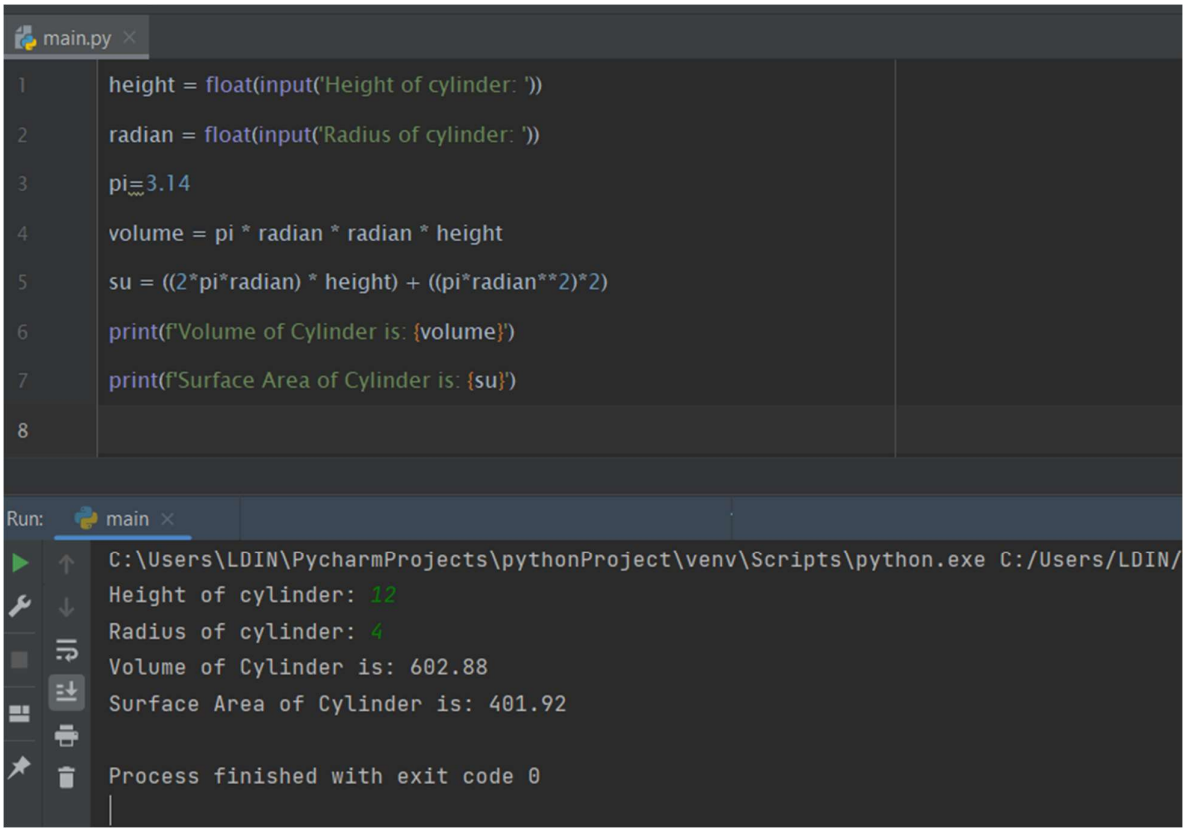


OUTPUT 2

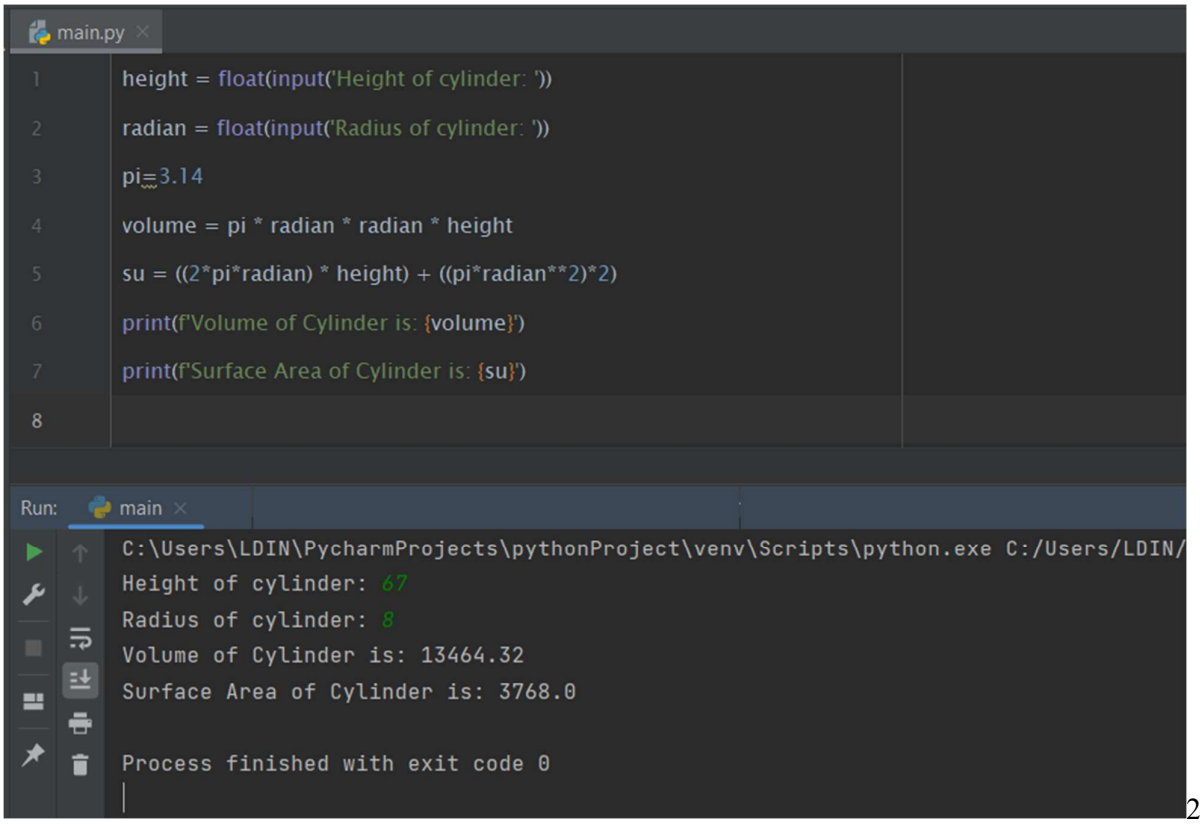


6.WRITE A PROGRAM TO CALCULATE SURFACE VOLUME AND AREA OF A CYLINDER.

OUTPUT 1

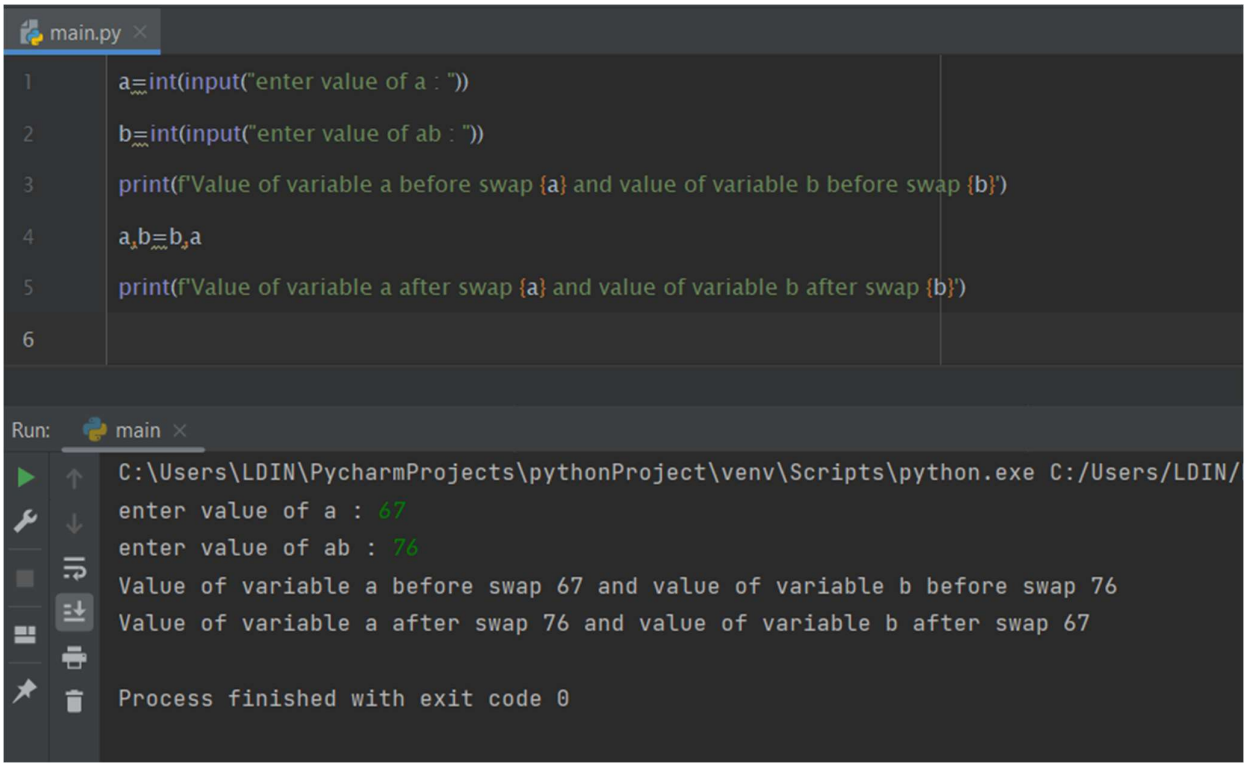


OUTPUT 2

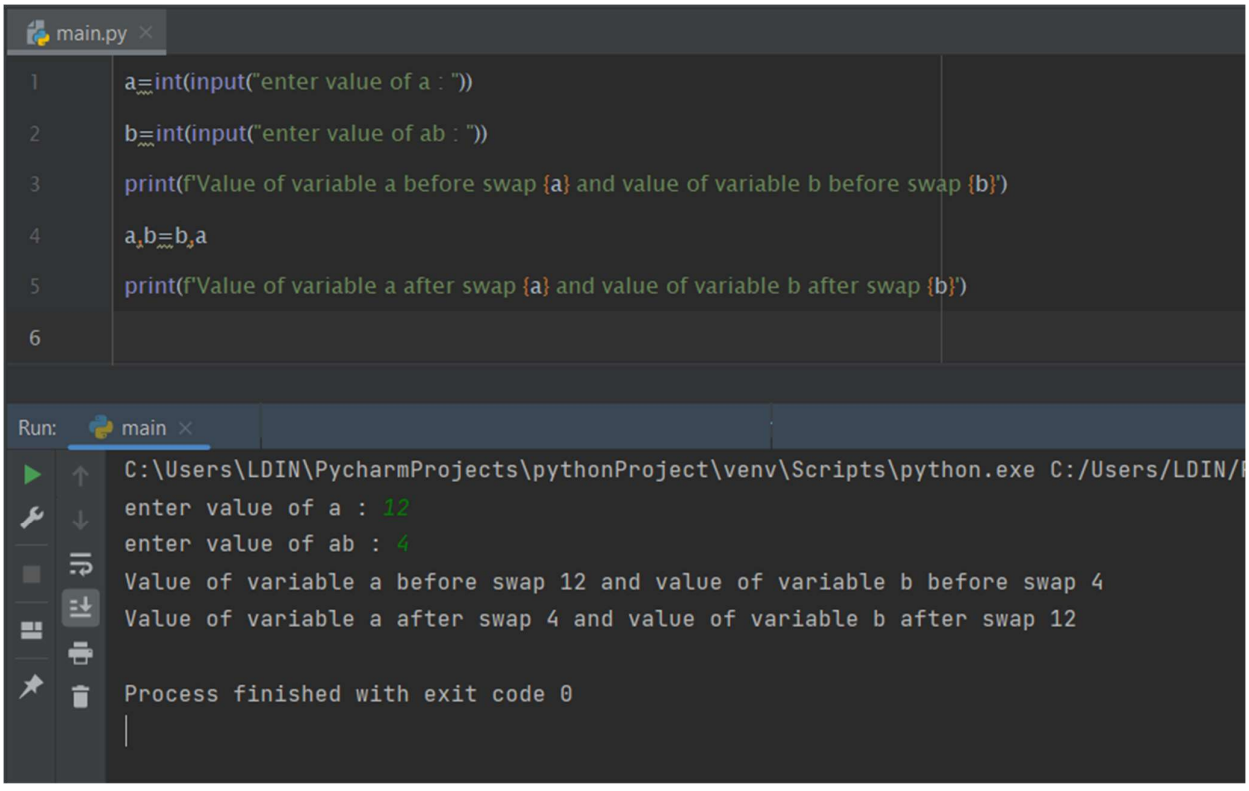


7. WRITE A PROGRAM TO SWAP THE VALUE OF TWO VARIABLES

OUTPUT 1



OUTPUT 2



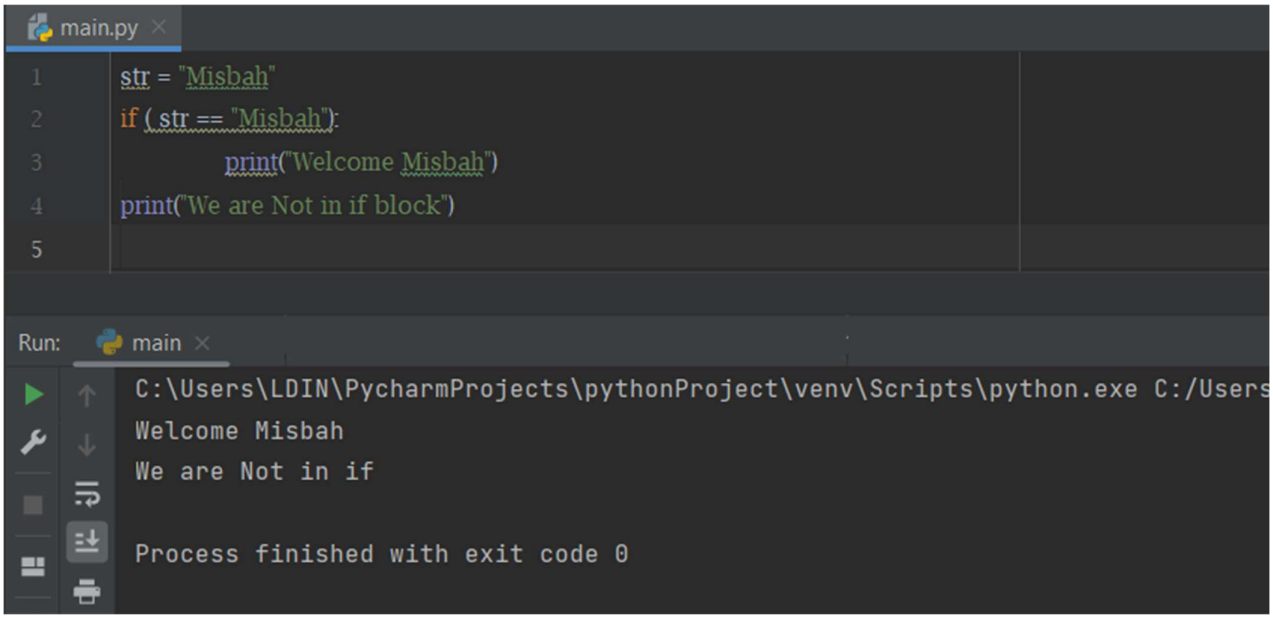
PRACTICAL NO. 4:
WRITE SIMPLE PYTHON PROGRAM TO DEMONSTRATE USE OF CONDITIONAL STATEMENTS: IF' STATEMENT, 'IF ... ELSE' STATEMENT, NESTED 'IF' STATEMENT

IF STATEMENT

The if statement is the simplest decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statement is executed otherwise not.

SYNTAX:

If condition:
    Statement(s)

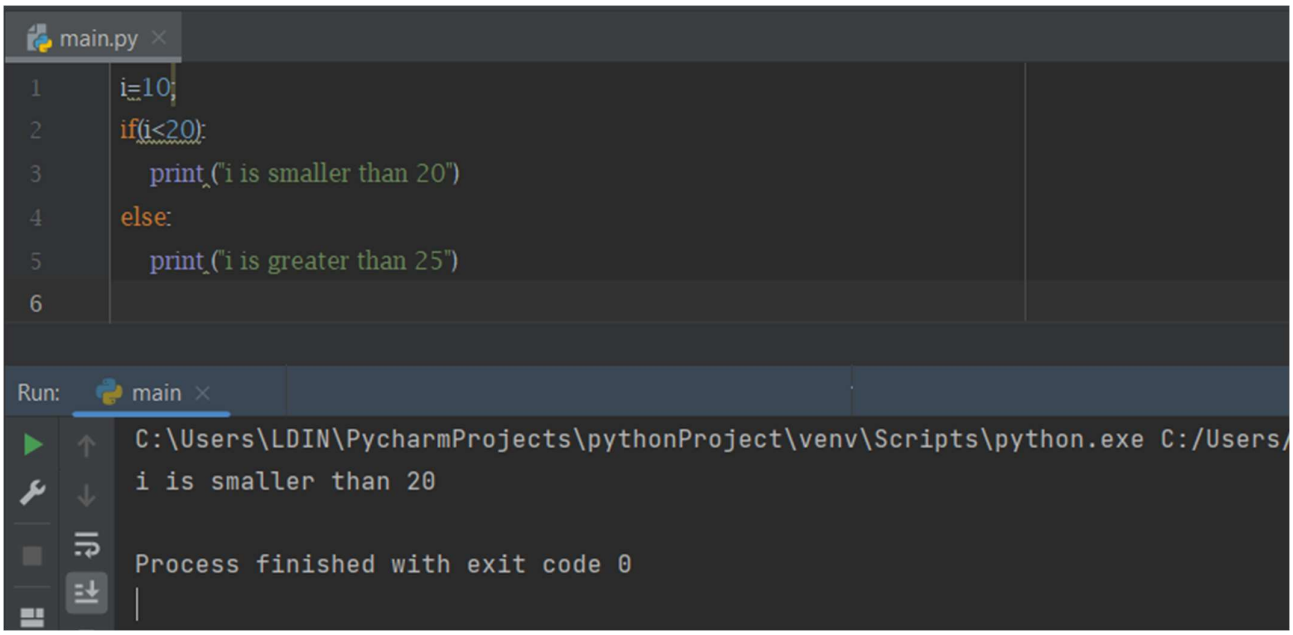


IF-ELSE STATEMENT:

The if statement alone tells us that if a condition is true, it will execute a block of statements and if the condition is false, it won't. But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with if statement to execute a block of code when the condition is false.

SYNTAX:

```
if (condition):
        # if Condition is true, Executes this block
else:
        # if condition is false, Executes this block
```



NESTED-IF STATEMENT:

A nested if is an if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement. Yes, Python allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement.

SYNTAX:

```
if (condition1):
        # Executes when condition1 is true
        if (condition2):
                # Executes when condition2 is true
        # if Block is end here
# if Block is end here
```

main.py

```
i =50
if(i ==50):
    # First if statement
    if(i < 80):
        print ("i is smaller than 20")
        # Nested - if statement will only be executed if statement above is true
if (i < 95):
    print ("i is smaller than 95 too")
else:
    print ("i is greater than 95")

```

Run: main

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/
i is smaller than 20
i is smaller than 95 too

Process finished with exit code 0
```

XI. EXERCISE

1. WRITE A PROGRAM TO CHECK WHETHER A NUMBER IS EVEN OR ODD

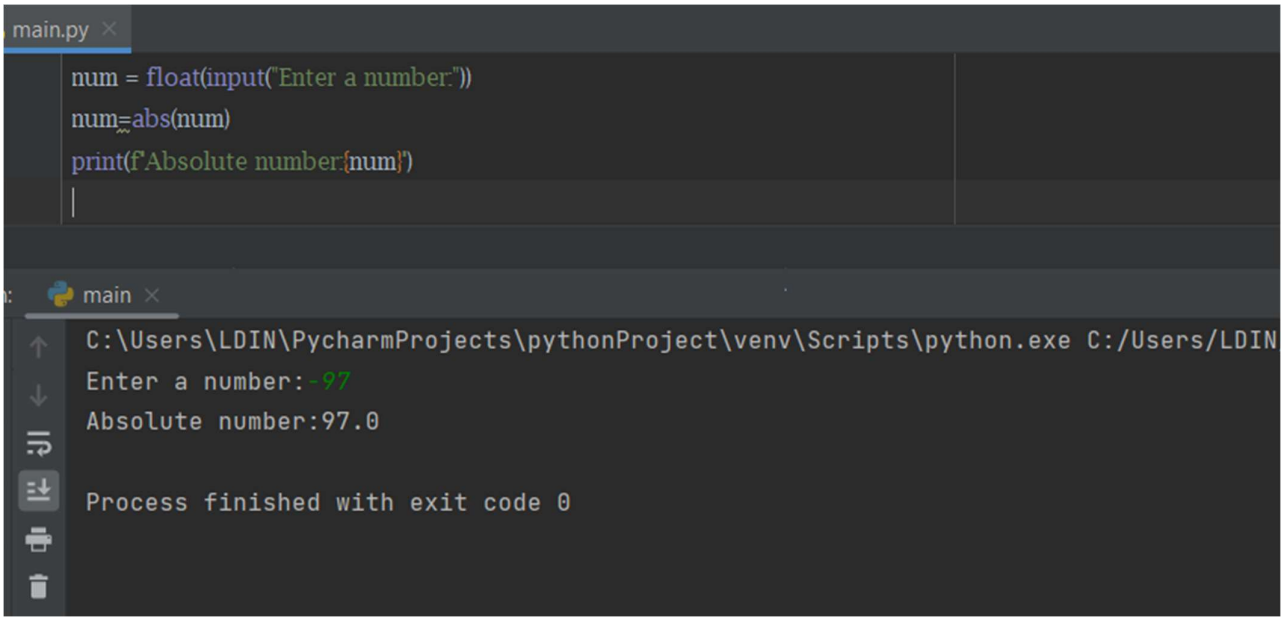
main.py

```
num = int(input('Enter a number: '))
if (num % 2) == 0:
  print(f'{num} is Even')
else:
  print(f'{num} is Odd')

```

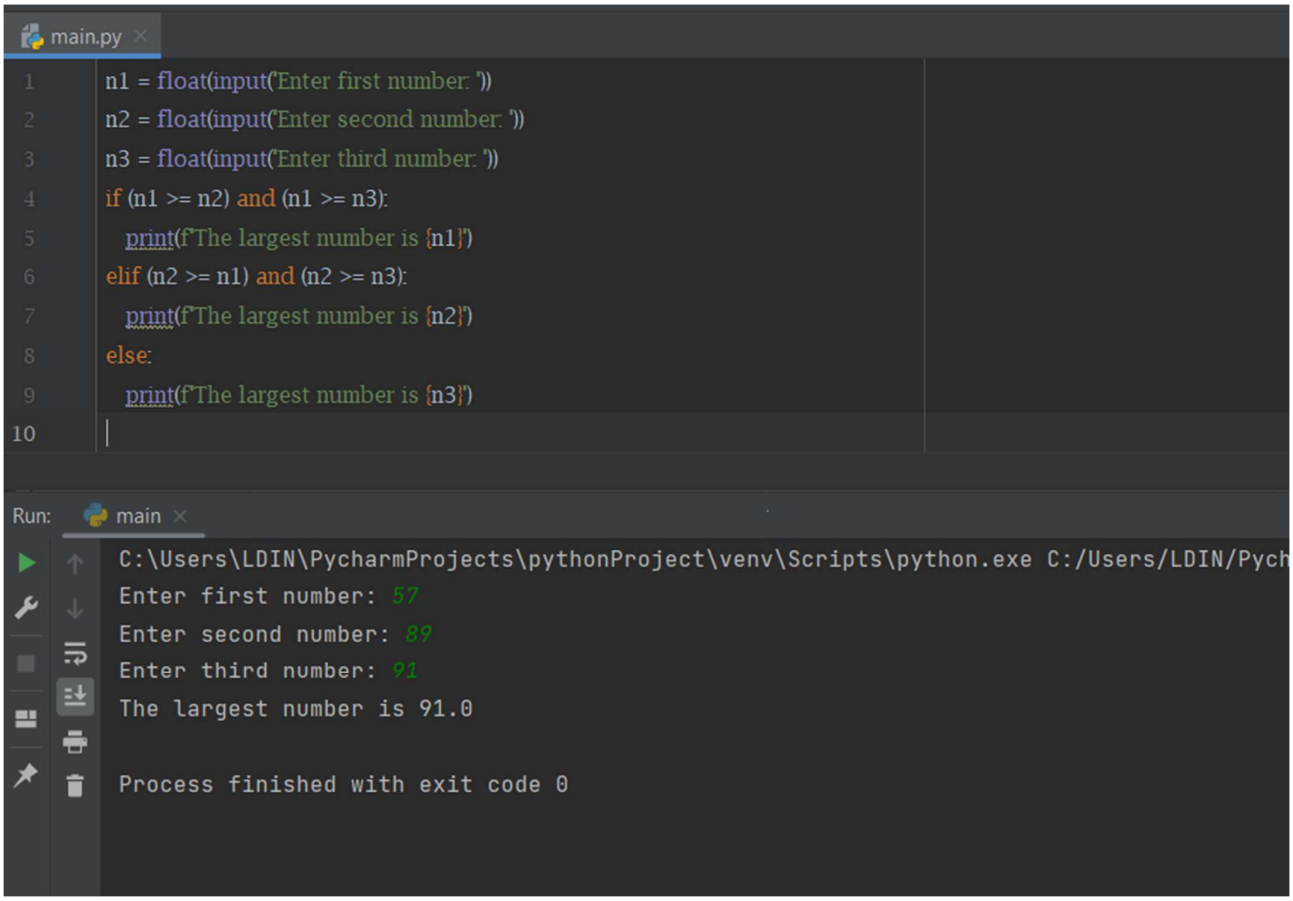
Run: main

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/
Enter a number: 44
44 is Even

Process finished with exit code 0
```

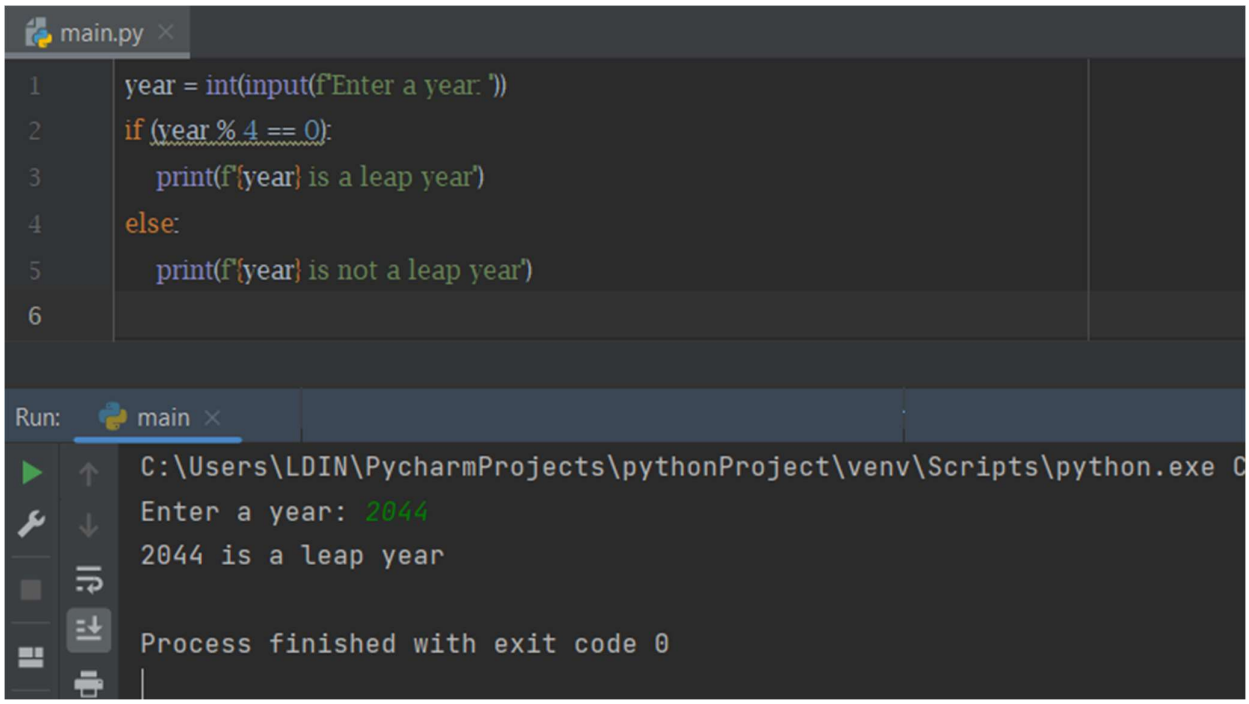
2. WRITE A PROGRAM TO FIND OUT ABSOLUTE VALUE OF AN INPUT NUMBER



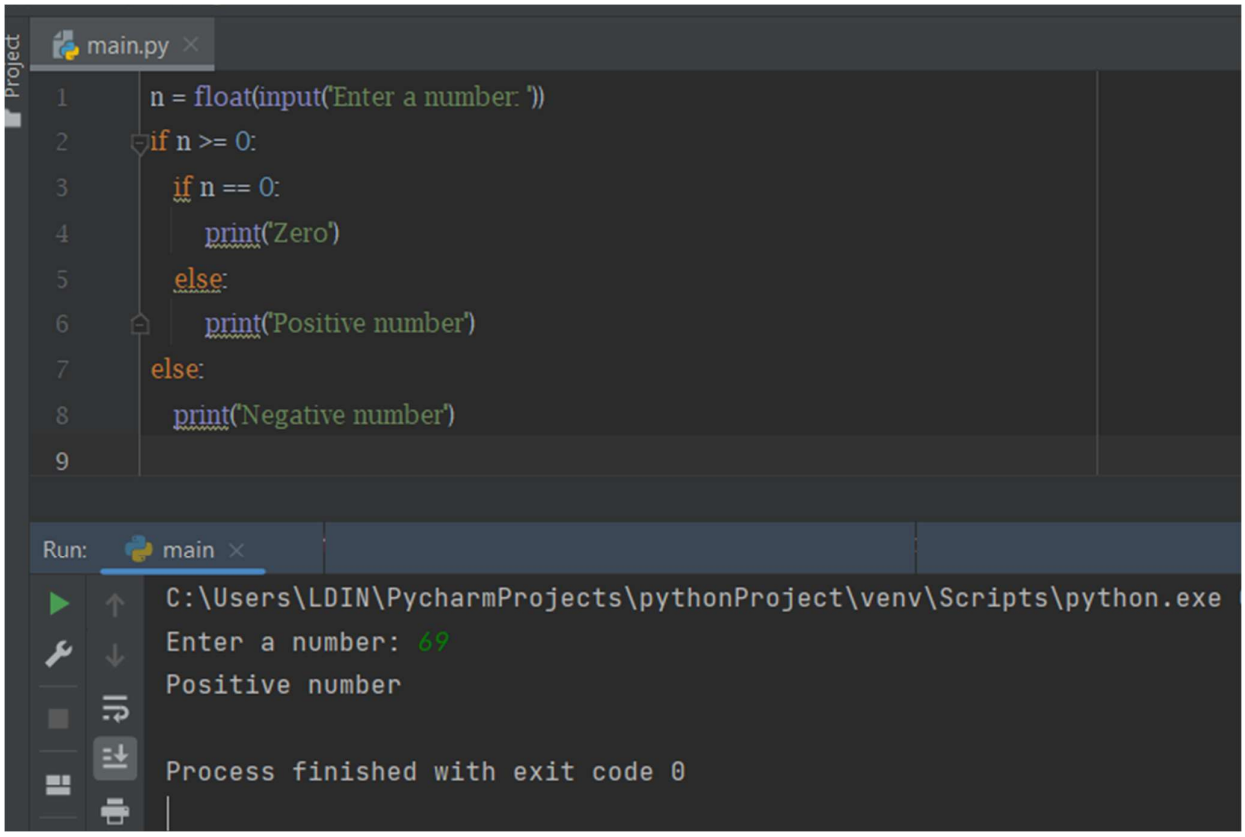
3. WRITE A PROGRAM TO CHECK THE LARGEST NUMBER AMONG THE THREE NUMBERS



4. WRITE A PROGRAM TO CHECK IF THE INPUT YEAR IS A LEAP YEAR OF NOT



5. WRITE A PROGRAM TO CHECK IF A NUMBER IS POSITIVE, NEGATIVE OR ZERO



```python
n = float(input('Enter a number: '))
if n >= 0:
    if n == 0:
        print('Zero')
    else:
        print('Positive number')
else:
    print('Negative number')
```

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProje
Enter a number: -42
Negative number

Process finished with exit code 0
```

6. WRITE A PROGRAM THAT TAKES THE MARKS OF 5 SUBJECTS AND DISPLAYS THE GRADE.

```python
s1=int(input('Enter marks of the first subject: '))
s2=int(input('Enter marks of the second subject: '))
s3=int(input('Enter marks of the third subject: '))
s4=int(input('Enter marks of the fourth subject: '))
s5=int(input('Enter marks of the fifth subject: '))
avg=(s1+s2+s3+s4+s5)/5
if(avg>=90):
    print('Grade: A')
elif(avg>=80 and avg<90):
    print('Grade: B')
elif(avg>=70 and avg<80):
    print('Grade: C')
elif(avg>=60 and avg<70):
    print('Grade: D')
else:
    print('Grade: F')
```

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProjects
Enter marks of the first subject: 85
Enter marks of the second subject: 70
Enter marks of the third subject: 92
Enter marks of the fourth subject: 87
Enter marks of the fifth subject: 89
Grade: B

Process finished with exit code 0
```

PRACTICAL NO. 5:
WRITE PYTHON PROGRAM TO DEMONSTRATE USE OF LOOPING STATEMENTS:
'WHILE' LOOP, 'FOR' LOOP AND NESTED LOOP

WHILE LOOP:

A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

SYNTAX :

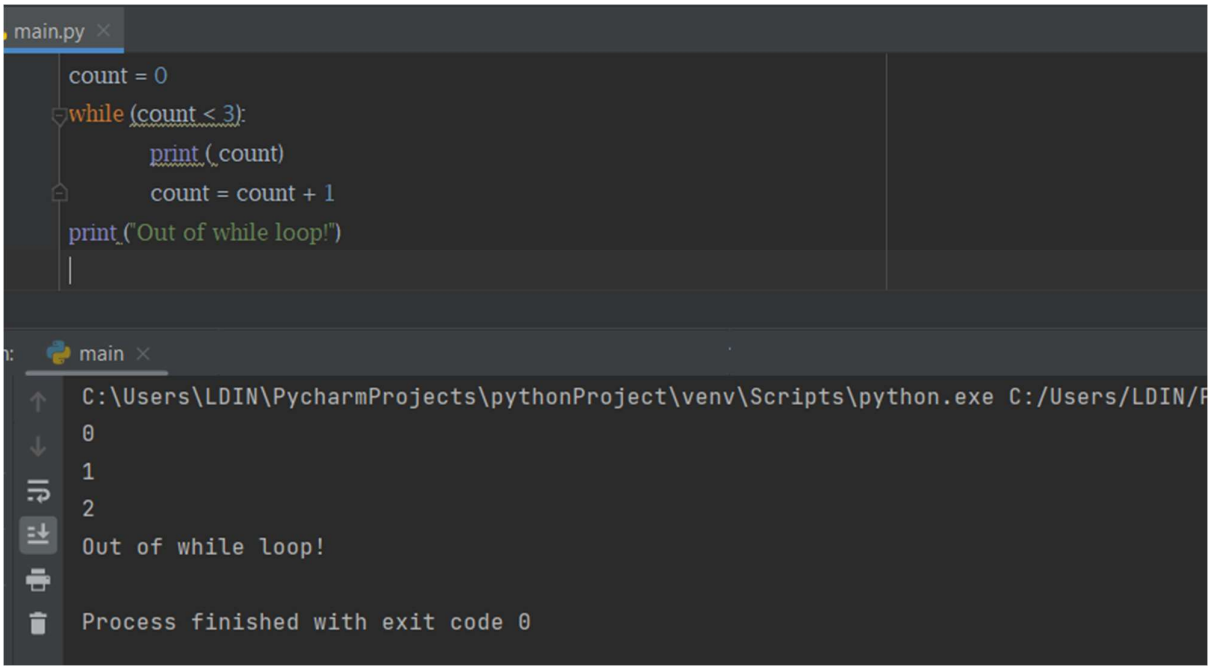
iteraror_var=initialized_value

while expression:

    statement(s)

    iterator inc/dec

The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop



FOR LOOP:

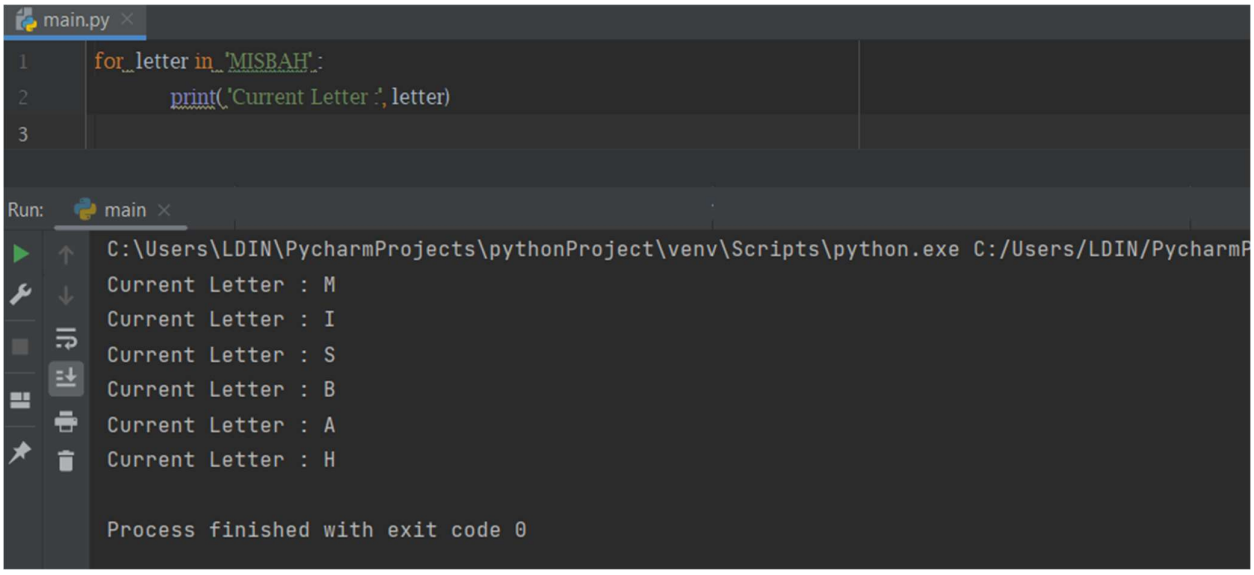
It has the ability to iterate over the items of any sequence, such as a list or a string.

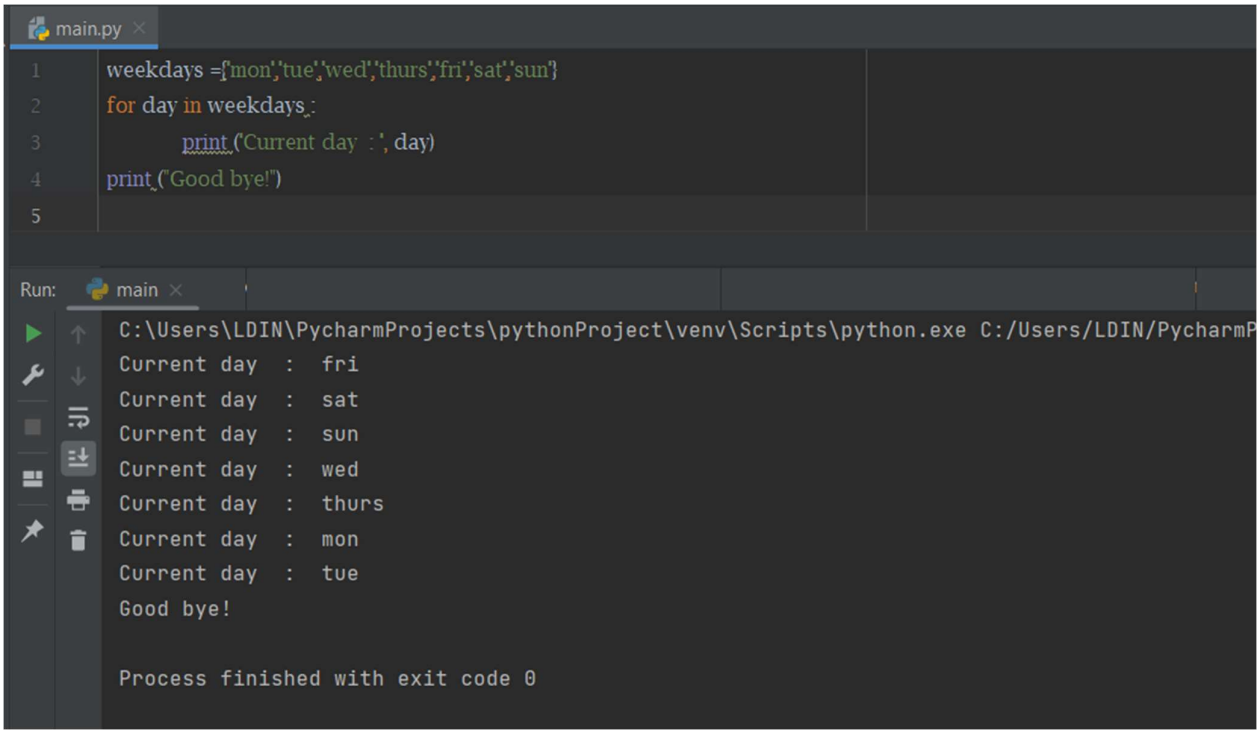
SYNTAX

for iterating_var in sequence:

statements(s)

EXAMPLE:





C) NESTED LOOPS:

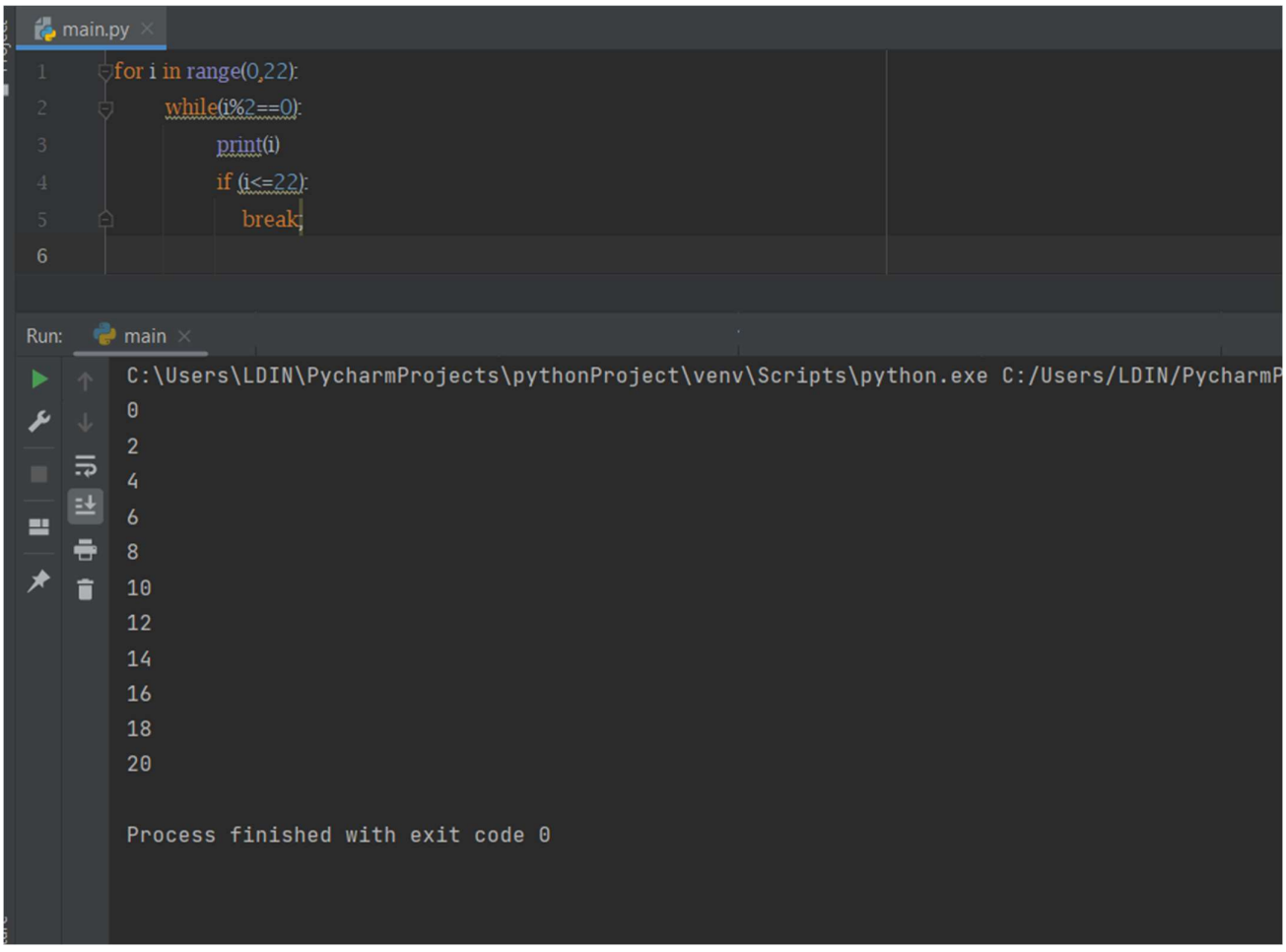
Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

SYNTAX

```
for iterating_var in sequence:
        for iterating_var in sequence:
            statements(s)
        statements(s)
```

EXAMPLE:

The code below uses two nested loops, an if statement and break statement to print all the even numbers from 0 to 20



LOOP MANIPULATION STATEMENTS

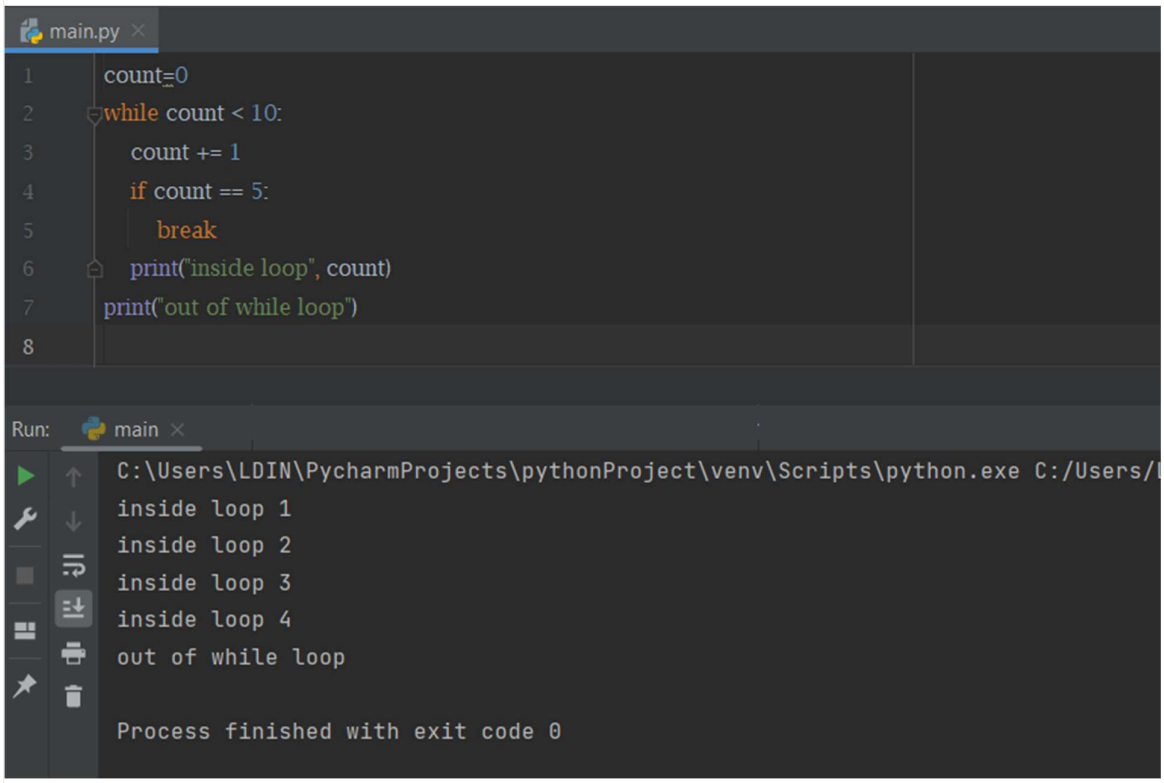
BREAK STATEMENT

We use break statement to terminate the current loop. When break statement is encountered inside a loop, control terminates execution of the current loop and control is transferred to the next executable statement

SYNTAX: -

break

EXAMPLE



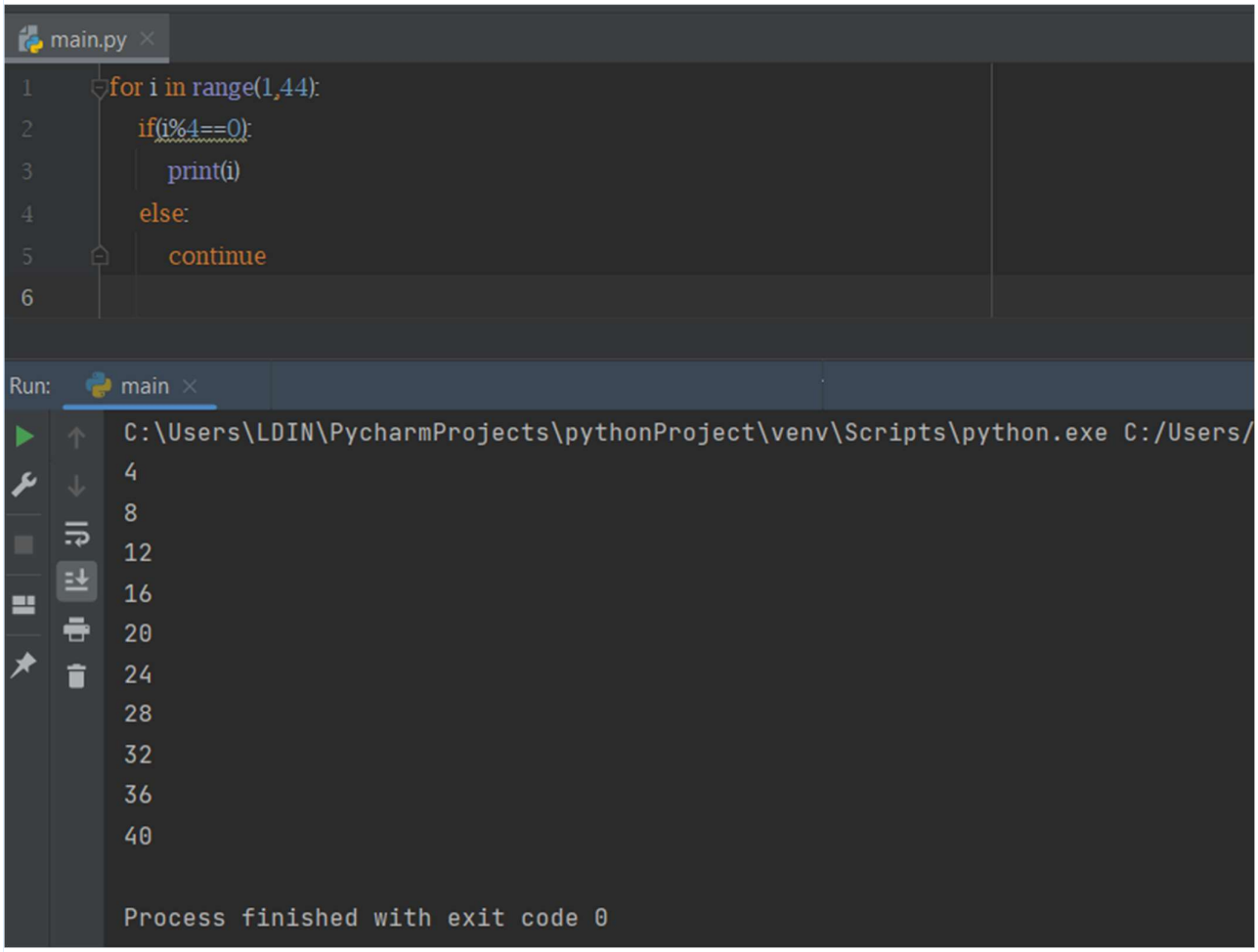
CONTINUE STATEMENT

The continue statement gives you way to skip over the current iteration of any loop. When a continue statement is encountered in the loop, the python interpreter ignores rest of statements in the loop body for current iteration and returns the program execution to the very first statement in the loop body. It does not terminate the loop rather continues with the next iteration.

SYNTAX: -

continue

EXAMPLE



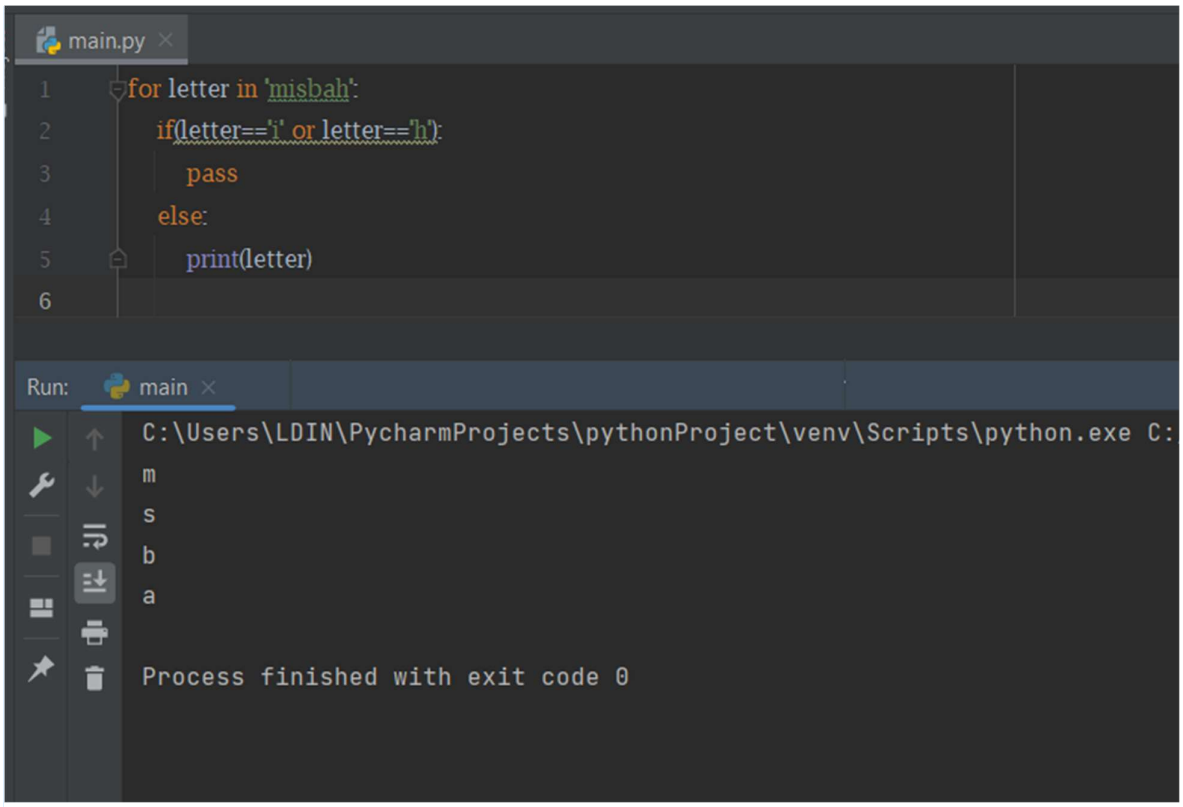
PASS STATEMENT

Pass statement is used to execute noting.it does nothing at all, also called as a null operation

SYNTAX: -

pass

EXAMPLE



XI. EXERCISE

1. PRINT THE FOLLOWING PATTERNS USING LOOP:

a. *

  **

  ***

  ****

b. *

  ***

 *****

  ***

  *

c. 1010101

  10101

  101

  1

main.py

```
for i in range(1,5):
    print("*"*i)
print('-'*15)
k = 0
for i in range(1, 3+1):
    for space in range(1, (3-i)+1):
        print(end="  ")
    while k!=(2*i-1):
        print("* ", end="")
        k += 1
    k = 0
    print()
for i in range(3, 1, -1):
    for space in range(0, 4-i):
        print("  ", end="")
    for j in range(i, 2*i-1):
        print("* ", end="")
    for j in range(1, i-1):
        print("* ", end="")
    print()

```

main

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN
*
**
***
****
---------------
    *
  * * *
* * * * *
  * * *
    *
```

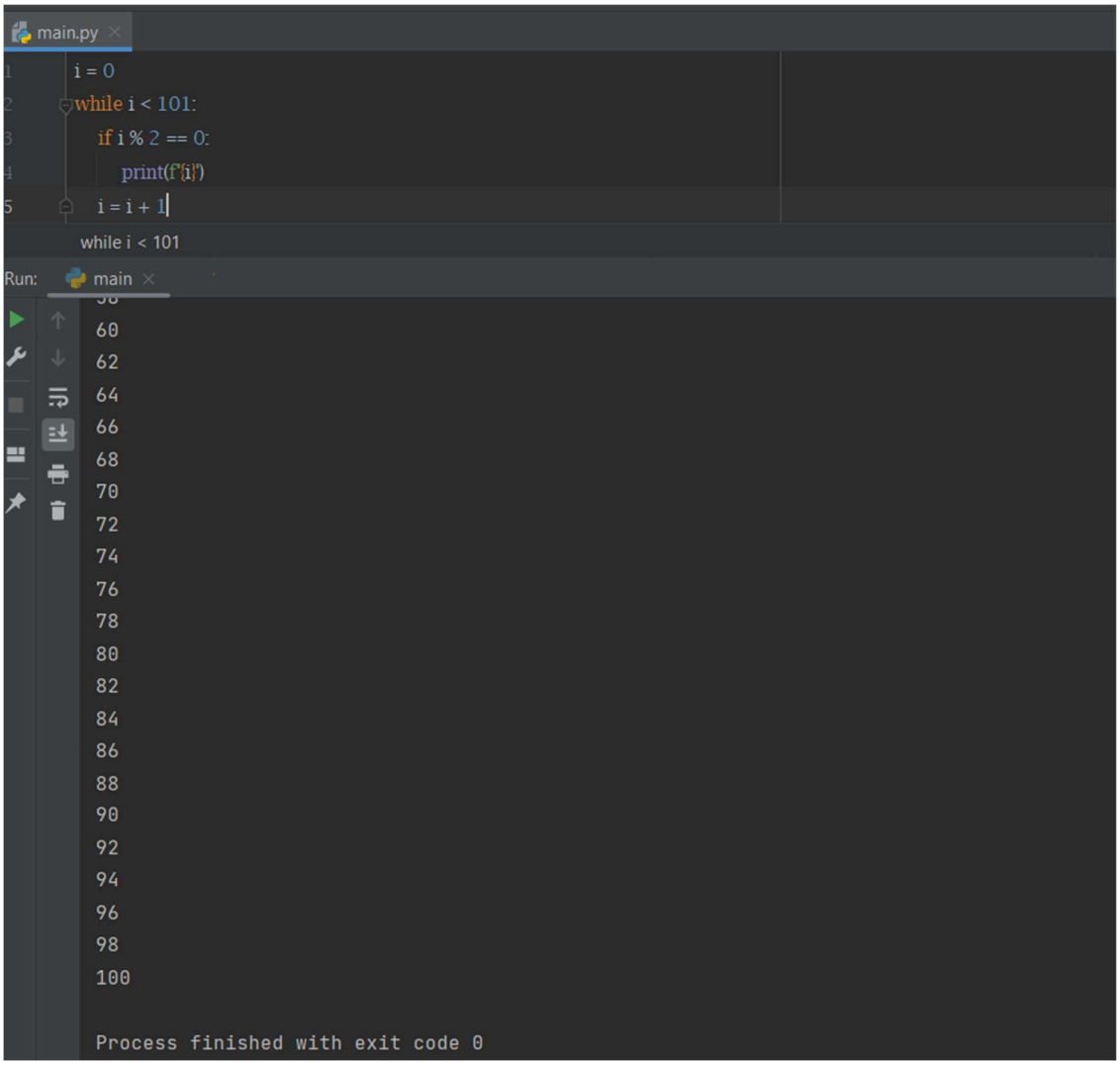
main.py

```
i = 7
s = 0
k=3
while (i>=0 and k>=0):
        a=(" "*s+"10"*k+"1"+" "*s)
        print(a)
        k=k-1
        i=i-2
        s=s+1
```

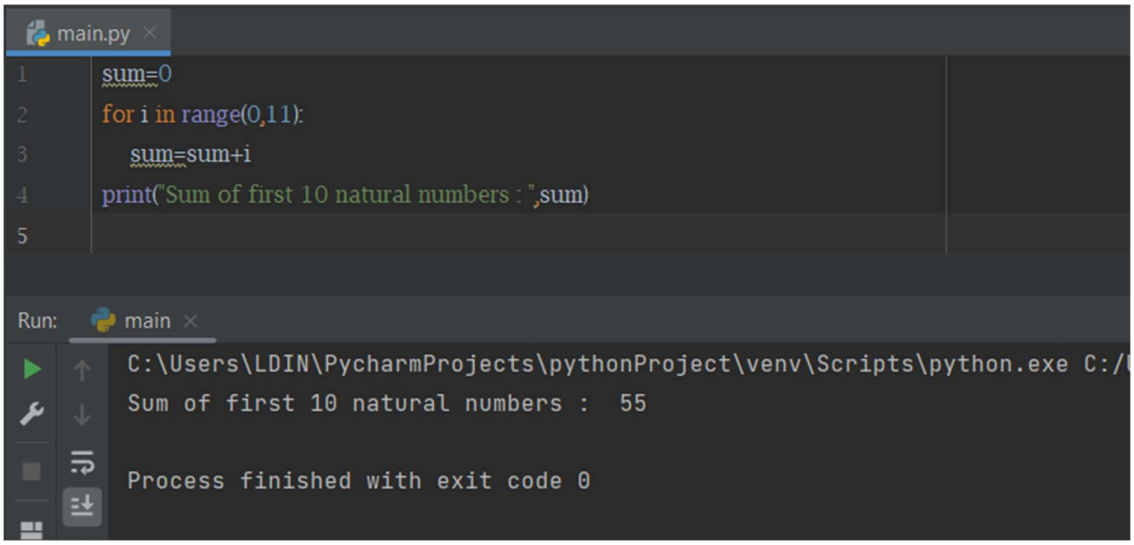
Run: main

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDI
1010101
 10101
  101
   1
```

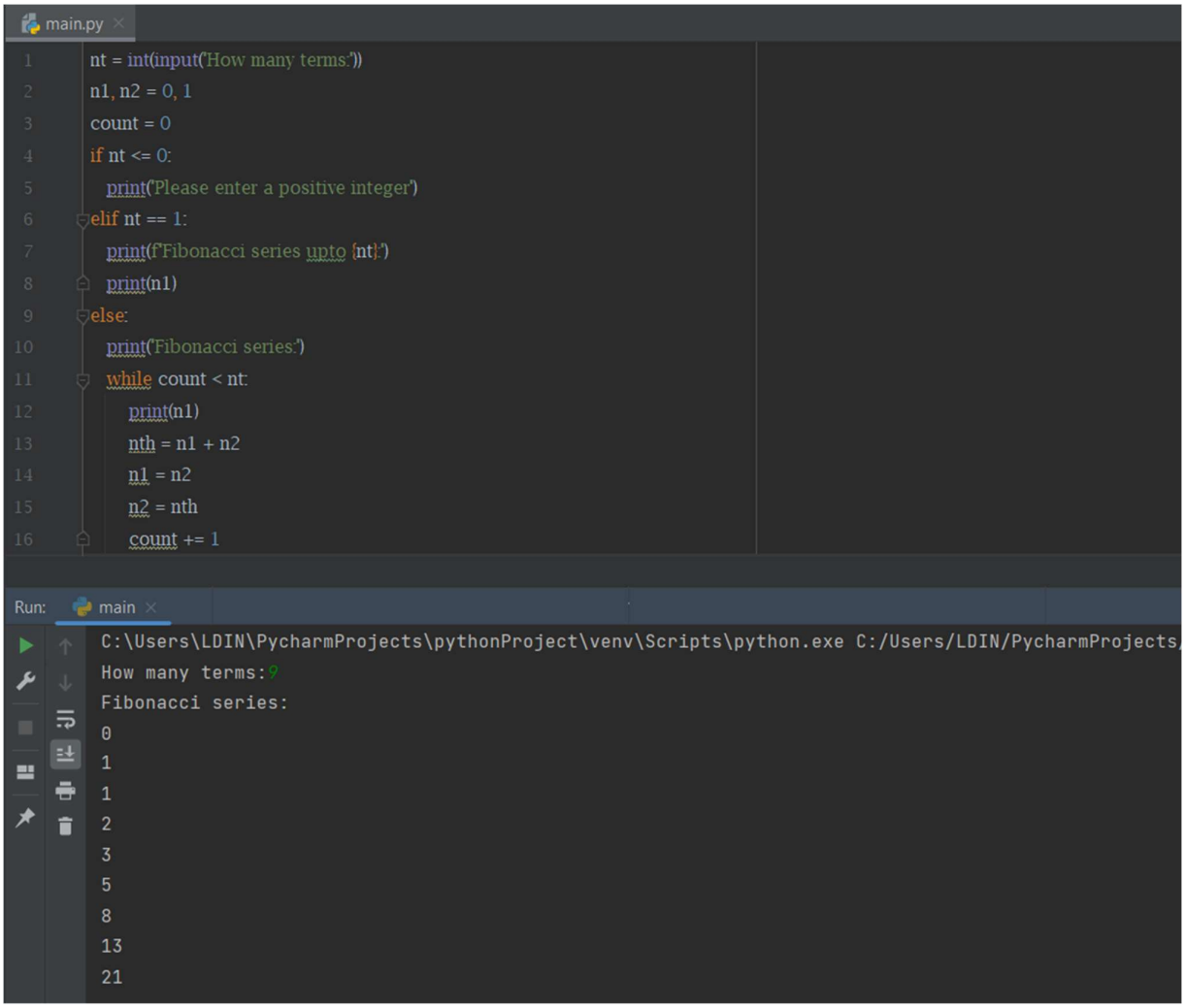
2. WRITE A PYTHON PROGRAM TO PRINT ALL EVEN NUMBERS BETWEEN 1 TO 100 USING WHILE LOOP.



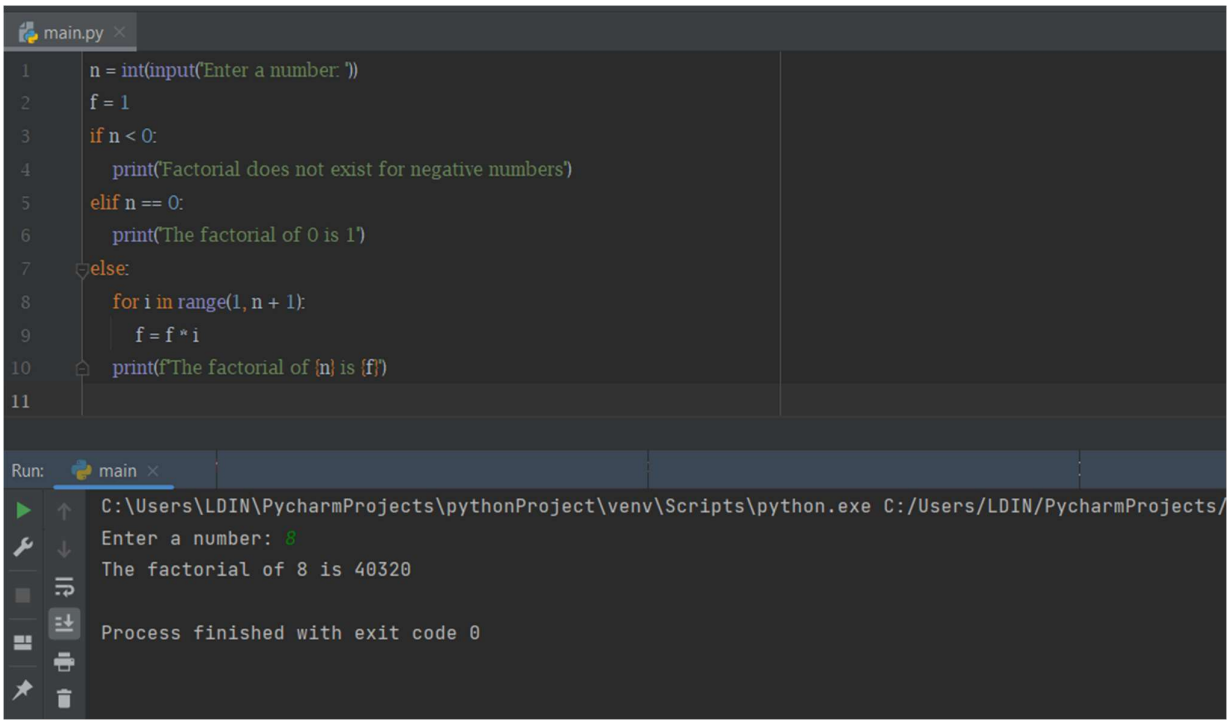
3. WRITE A PYTHON PROGRAM TO FIND THE SUM OF FIRST 10 NATURAL NUMBERS USING FOR LOOP



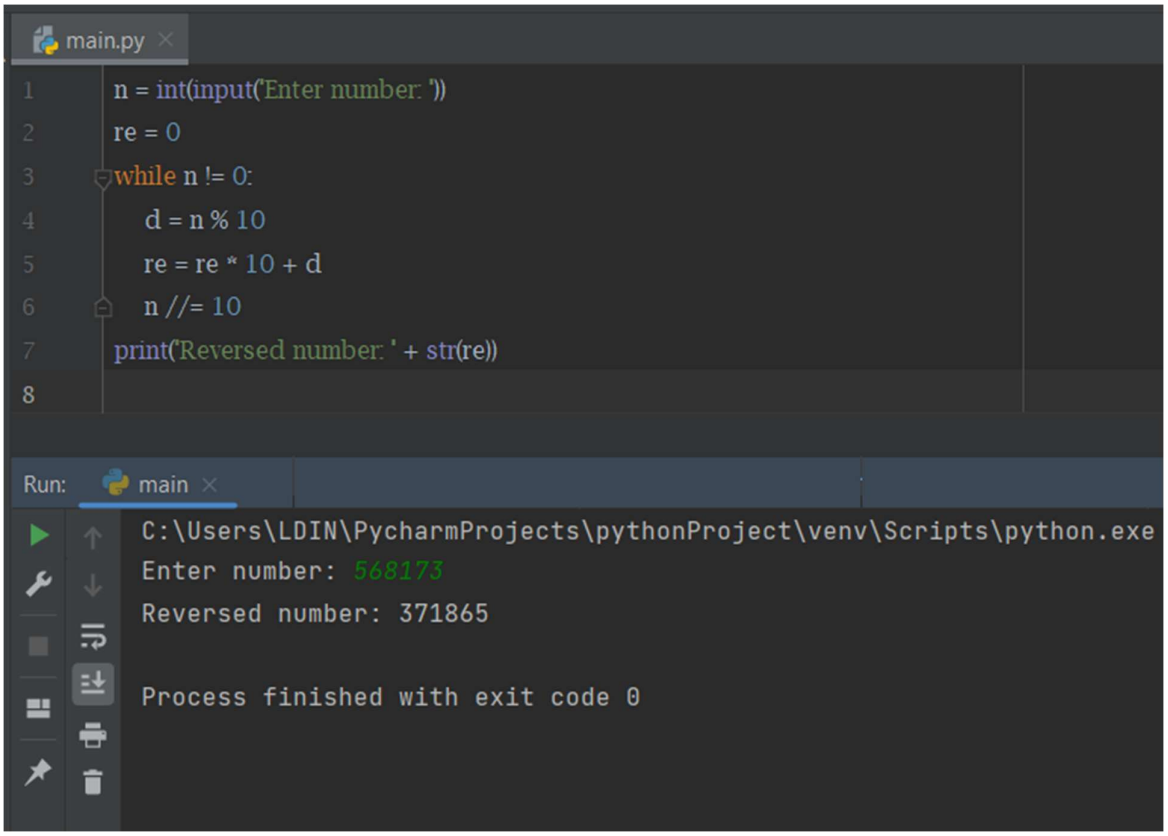
4. WRITE A PYTHON PROGRAM TO PRINT FIBONACCI SERIES.



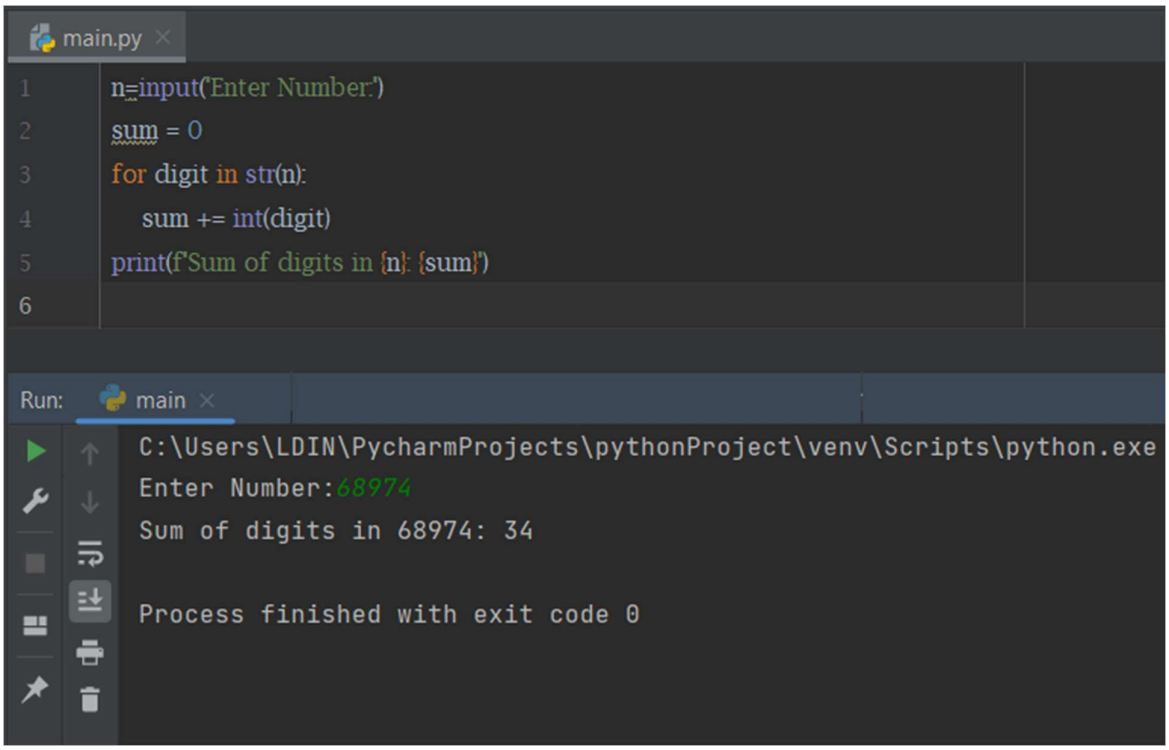
5. WRITE A PYTHON PROGRAM TO CALCULATE FACTORIAL OF A NUMBER



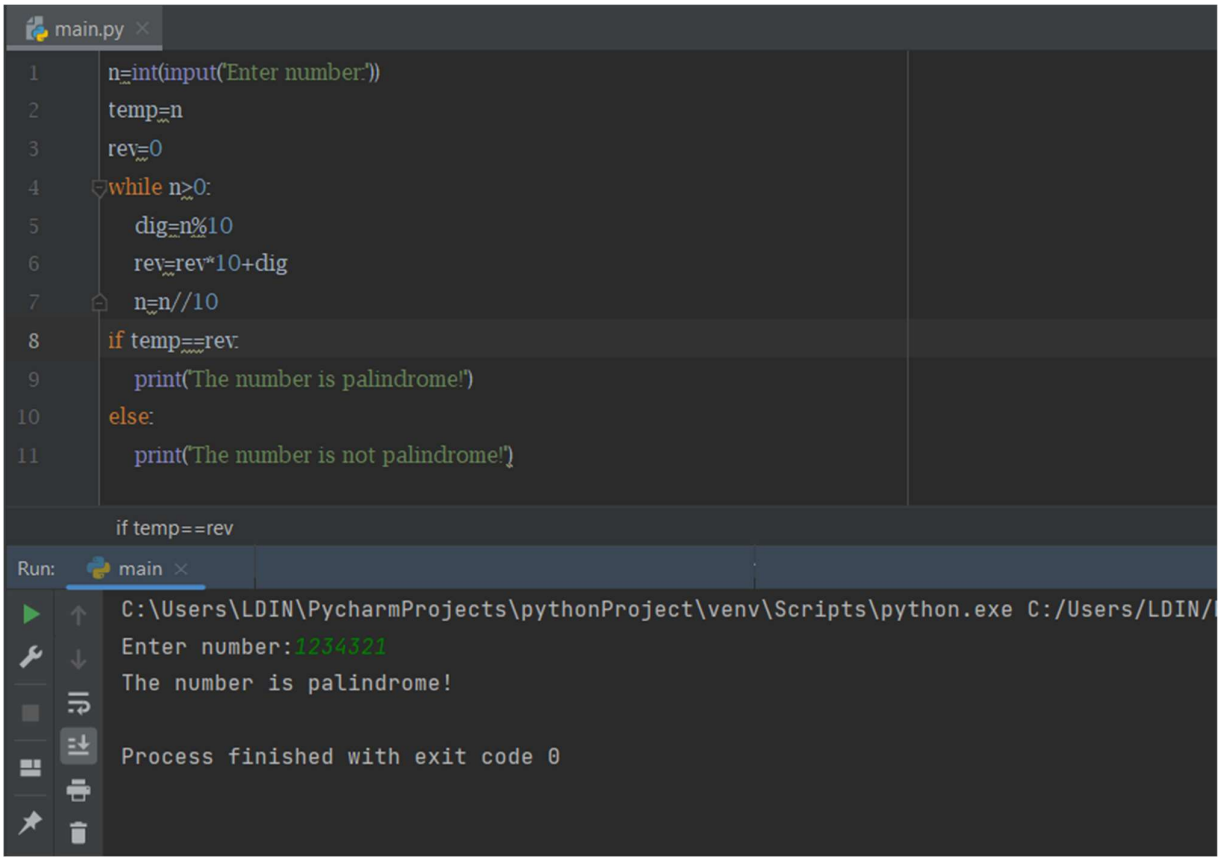
6. WRITE A PYTHON PROGRAM TO REVERSE A GIVEN NUMBER

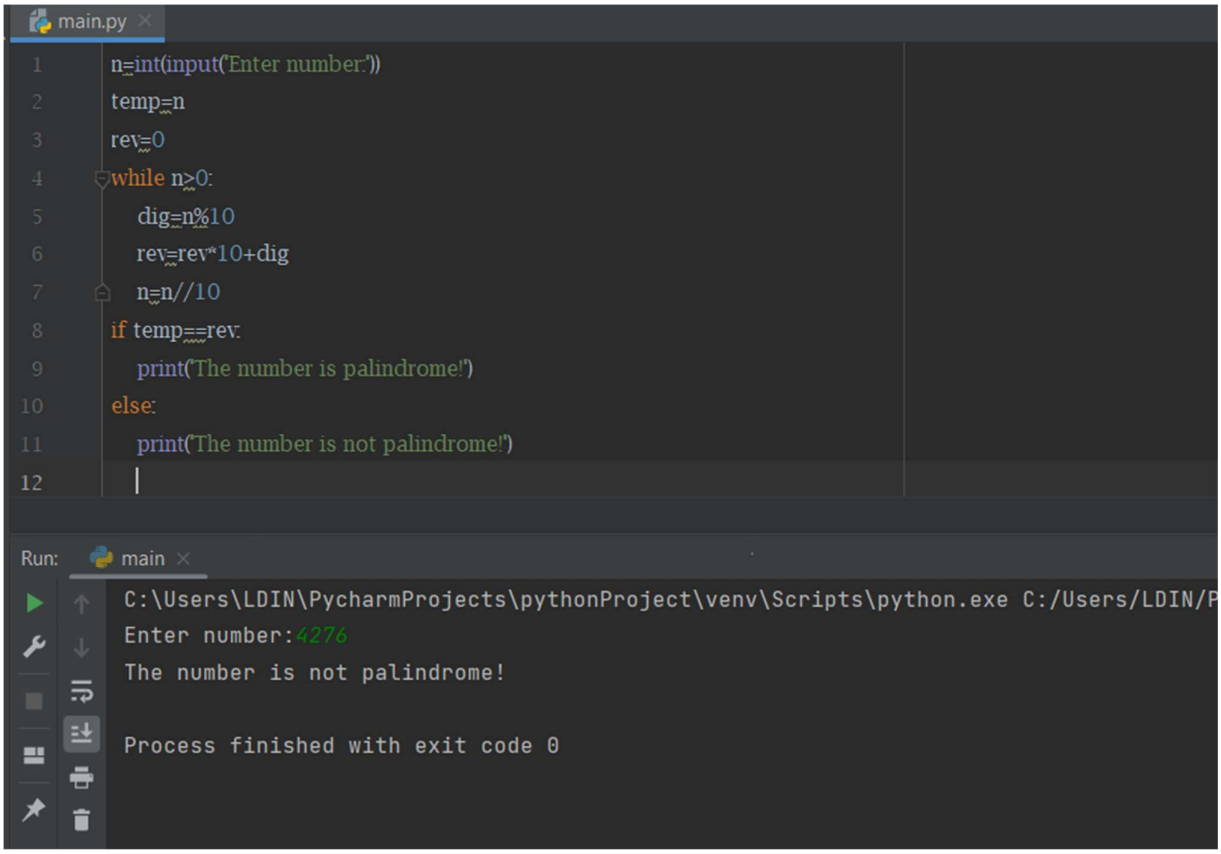


7. WRITE A PYTHON PROGRAM TAKES IN A NUMBER AND FINDS THE SUM OF DIGITS IN A NUMBER.



8. WRITE A PYTHON PROGRAM THAT TAKES A NUMBER AND CHECKS WHETHER IT IS A PALINDROME OR NOT.





PRACTICAL NO. 8:
WRITE PYTHON PROGRAM TO PERFORM FOLLOWING OPERATIONS ON SET: CREATE SET, ACCESS SET ELEMENTS, UPDATE SET, DELETE SET

SETS

1. Sets are immutable data structures in Python
2. Values are enclosed in {} curly braces and separated by commas
3. It is an unordered collection of unique items i.e., it doesn't contain duplicate elements
4. When we print a set, it isn't usually printed in the order in which it was declared
5. Sets are unindexed, values cannot be accessed using index position

CREATING A SET

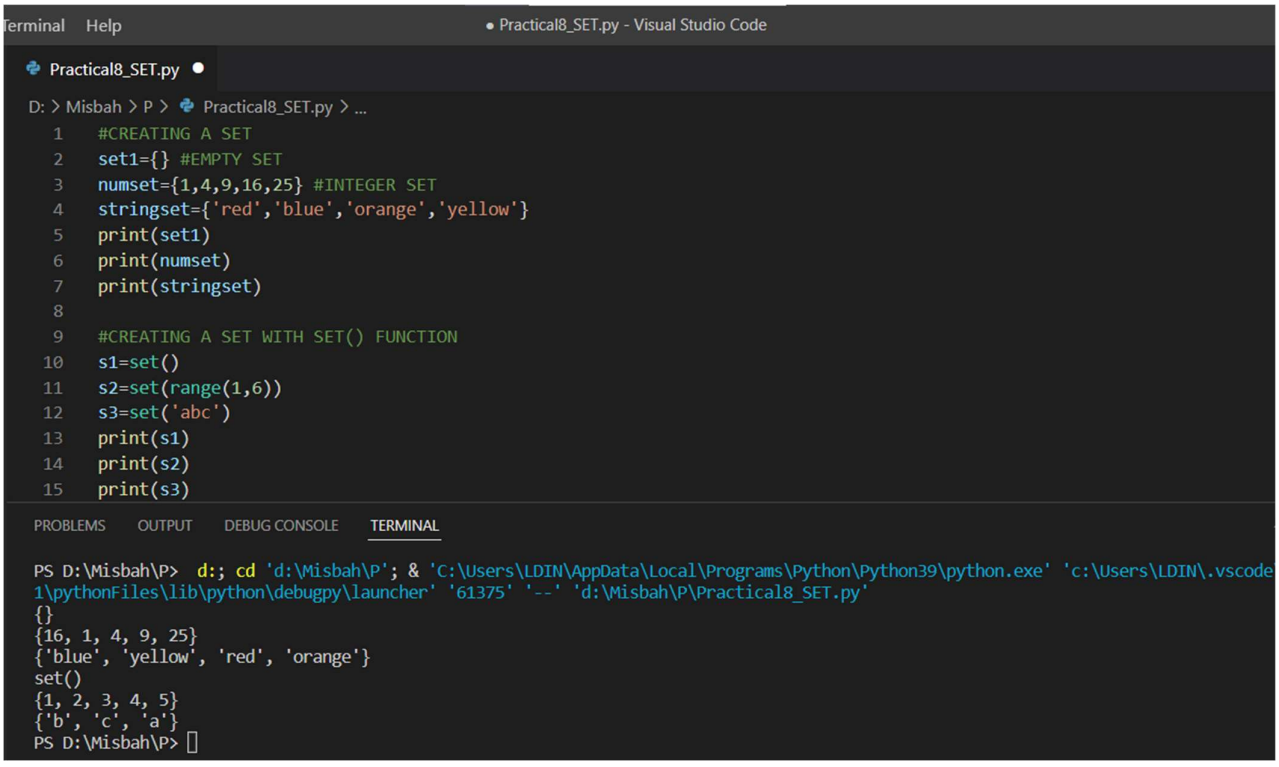
We can create set in following two ways

1.By using following Syntax

setName= {val1, val2, val2…. valn}

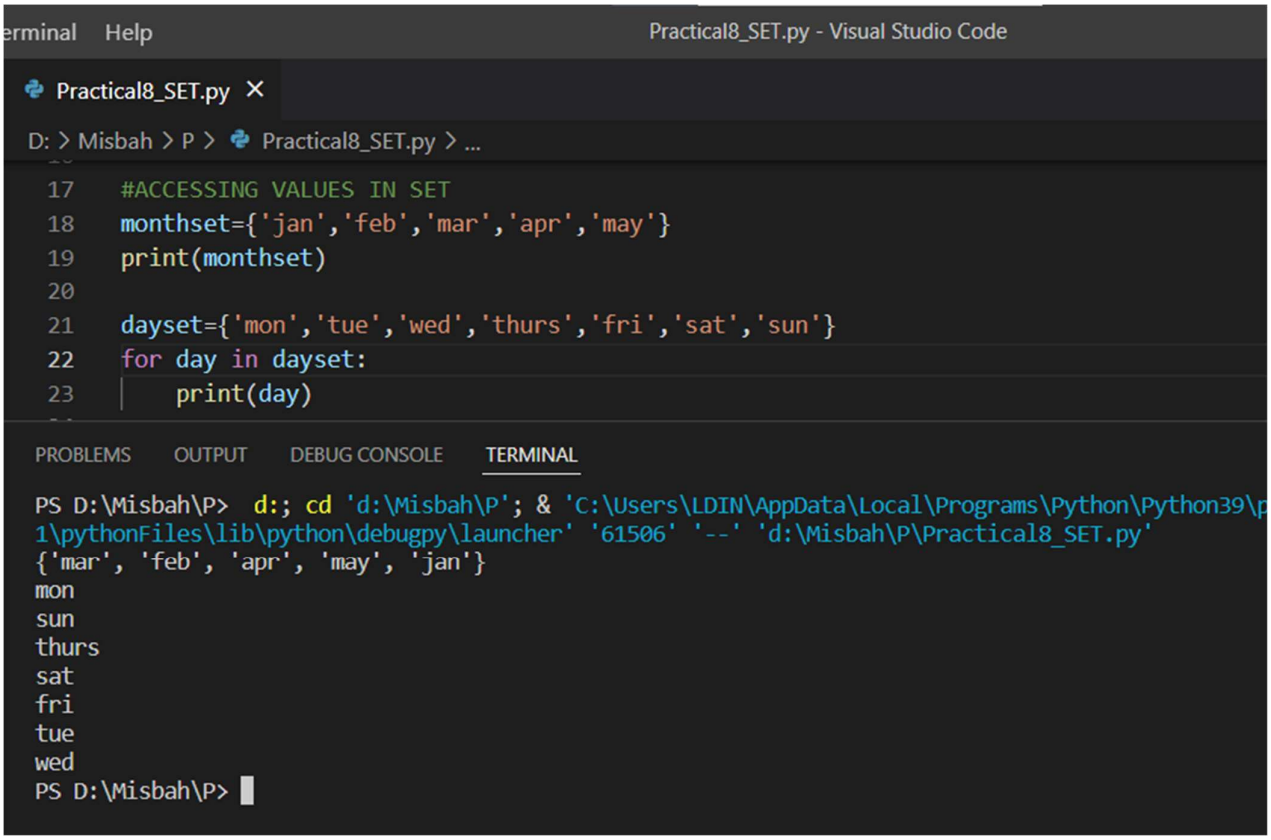
2.By using set() function

setName=set(value(s))



ACCESSING VALUES IN A SET

We can access values in a set by simply printing the set name or by using a for loop



UPDATING VALUES IN A SET

We can update values in Set by using add() function and update() function

1. **add()**

It adds elements to the set and can take only one argument

SYNTAX:

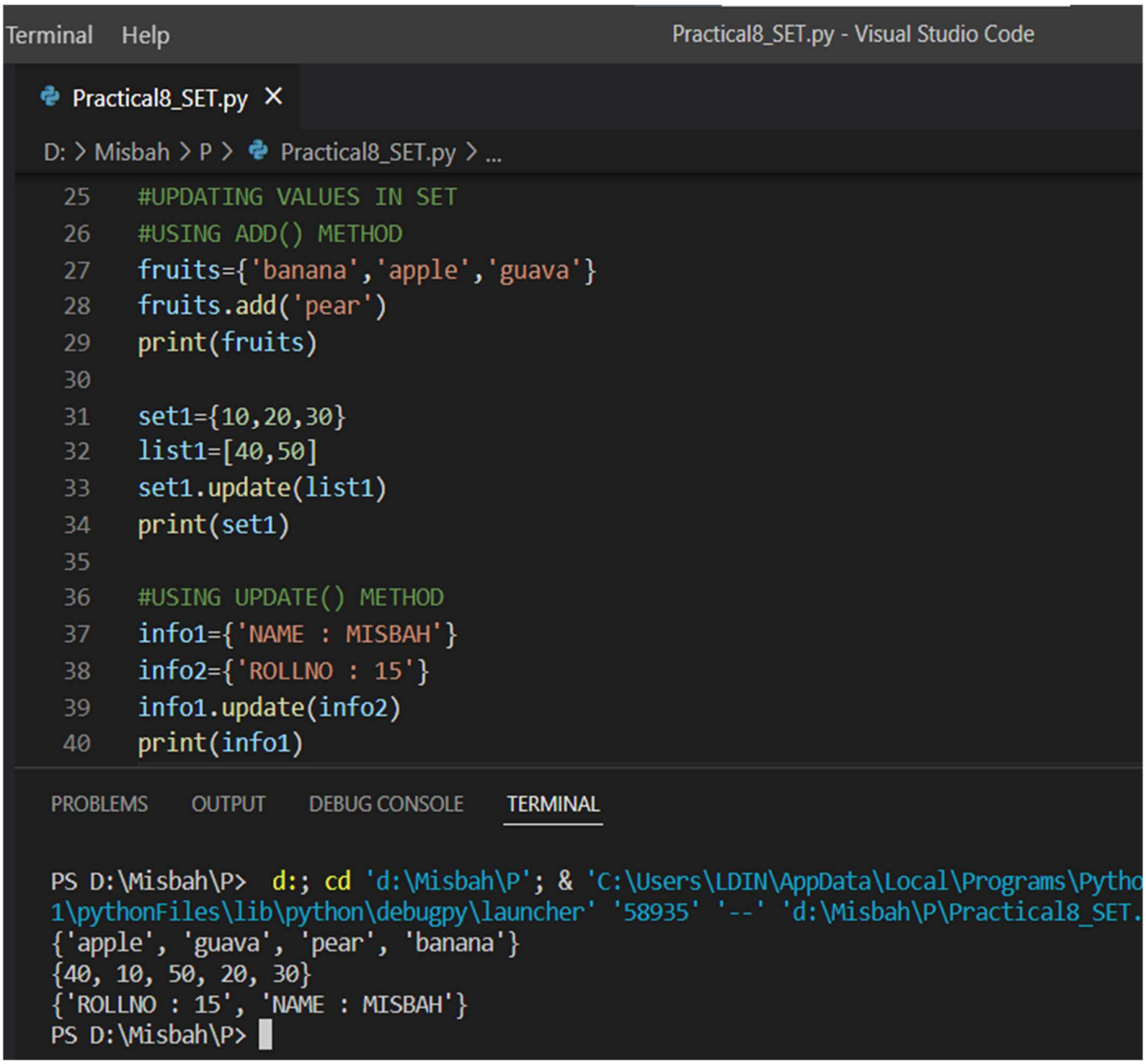
setName.add(element)

2. **update()**

With the help of this method, we can add set to a set, list to a set, string to set and tuples to a set, in all cases duplicate elements are not included

SYNTAX:

setName.update(setName | listName | tupleName | String)



DELETING VALUES IN A SET

Deletion of values in set is possible with the help of the following four methods

- pop() : removes random item or last item from the set
- discard() : deletes the item specified
- remove(): deletes the item specified ,except it raises a KeyError if element doesn't exist
- clear() : removes all elements from the set

```
Terminal  Help                                   Practical8_SET.py - Visual Studio Code
Practical8_SET.py
D: > Misbah > P > Practical8_SET.py > ...
42  #DELETING VALUES IN A SET
43
44  flowers={'rose','jasmine','papaya','lily','cherry'}
45  print(flowers)
46  flowers.discard('papaya')
47  print(flowers)
48  flowers.remove('jasmine')
49  print(flowers)
50  flowers.pop()
51  print(flowers)
52  flowers.clear()
53  print(flowers)
54
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\Misbah\P>  d:; cd 'd:\Misbah\P'; & 'C:\Users\LDIN\AppData\Local\Programs\Python\Python3
1\pythonFiles\lib\python\debugpy\launcher' '59019' '--' 'd:\Misbah\P\Practical8_SET.py'
{'papaya', 'cherry', 'lily', 'rose', 'jasmine'}
{'cherry', 'lily', 'rose', 'jasmine'}
{'cherry', 'lily', 'rose'}
{'lily', 'rose'}
set()
PS D:\Misbah\P>
```

BASIC SET OPERATIONS

Sets can be used to carry out mathematical set operations like *union, intersection, difference and symmetric difference*. We can do this with operators or methods. Consider following two sets

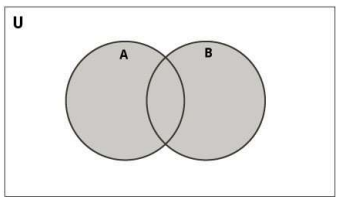
A = {1, 2, 3, 4, 5} B = {4, 5, 6, 7, 8}

1.UNION

Union of A and B is a set of all elements from both sets.

Union is performed using | operator.

Same can be accomplished using the union() method.

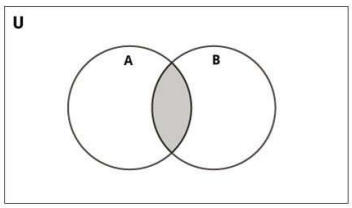


2.SET INTERSECTION

Intersection of A and B is a set of elements that are common in both the sets.

Intersection is performed using & operator.

Same can be accomplished using the intersection() method.

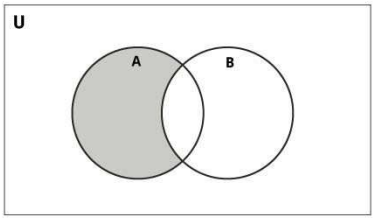


3.SET DIFFERENCE

Difference of the set B from set A(A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of elements in B but not in A.

Difference is performed using - operator.

Same can be accomplished using the difference() method.

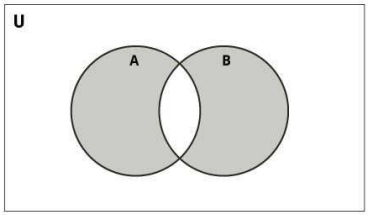


4.SET SYMMETRIC DIFFERENCE

Symmetric Difference of A and B is a set of elements in A and B but not in both (excluding the intersection).

Symmetric difference is performed using ^ operator.

Same can be accomplished using the method symmetric_difference()



SET OPERATIONS USING OPERATORS

```
D: > Misbah > P > Practical8_SET.py > ...
56  #SET OPERATIONS
57
58  A = {1, 2, 3, 4, 5}
59  B = {4, 5, 6, 7, 8}
60  print("Set union method")
61  print(A | B)
62  print("Set intersection method")
63  print(A & B)
64  print("Set difference method")
65  print(A - B)
66  print("Set symmetric differencce method")
67  print(A ^ B)
68
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Misbah\P>  d:; cd 'd:\Misbah\P'; & 'C:\Users\LDIN\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\LDIN
1\pythonFiles\lib\python\debugpy\launcher' '65362' '--' 'd:\Misbah\P\Practical8_SET.py'
Set union method
{1, 2, 3, 4, 5, 6, 7, 8}
Set intersection method
{4, 5}
Set difference method
{1, 2, 3}
Set symmetric differencce method
{1, 2, 3, 6, 7, 8}
PS D:\Misbah\P>
```

SET OPERATIONS USING INBUILT FUNCTIONS

```
D: > Misbah > P > Practical8_SET.py > ...
67  A = {1, 2, 3, 4, 5}
68  B = {4, 5, 6, 7, 8}
69  #WITH FUNCTIONS
70  print("Set union method")
71  print(A.union(B))
72  print(B.union(A))
73  print("Set intersection method")
74  print(A.intersection(B))
75  print(B.intersection(A))
76  print("Set difference method")
77  print(A.difference(B))
78  print(B.difference(A))
79  print("Set symmetric difference method")
80  print(A.symmetric_difference(B))
81  print(B.symmetric_difference(A))
82
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Misbah\P>  & 'C:\Users\LDIN\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\LD
on\debugpy\launcher' '65435' '--' 'd:\Misbah\P\Practical8_SET.py'
Set union method
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
Set intersection method
{4, 5}
{4, 5}
Set difference method
{1, 2, 3}
{8, 6, 7}
Set symmetric difference method
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
PS D:\Misbah\P>
```

XI. EXERCISE

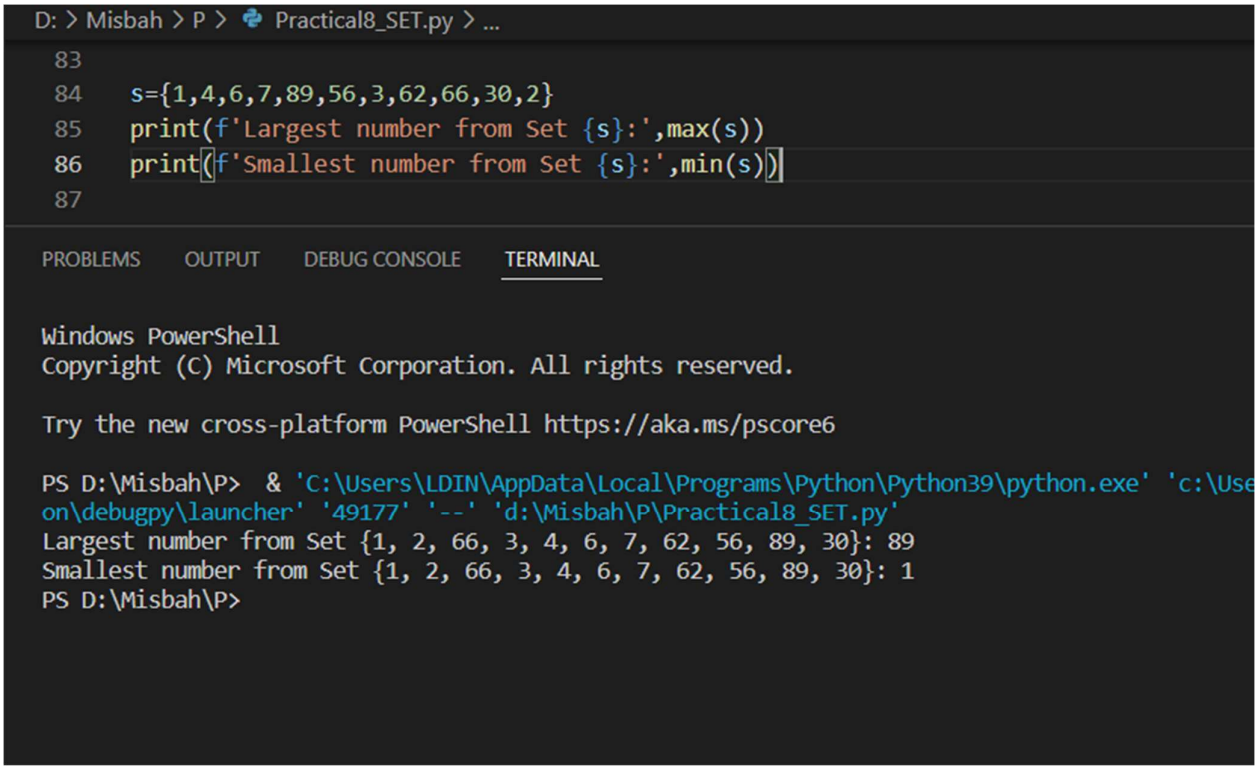
1. WRITE A PYTHON PROGRAM TO CREATE A SET, ADD MEMBER(S) IN A SET AND REMOVE ONE ITEM FROM SET.

```
D: > Misbah > P > Practical8_SET.py > ...
 82
 83    s1={12,45,32,76,14}
 84    print(f'Elements in {s1} before updating')
 85    s1.add(67)
 86    s1.add(98)
 87    print(f'Elements in {s1} after updating')
 88    s1.pop()
 89    print(f'Elements in {s1} after using pop')
 90    s1.discard(76)
 91    print(f'Elements in {s1} after using discard')
 92    s1.remove(98)
 93    print(f'Elements in {s1} after using remove')
 94

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

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

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

PS D:\Misbah\P>  & 'C:\Users\LDIN\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\LDIN\.vsco
on\debugpy\launcher' '65458' '--' 'd:\Misbah\P\Practical8_SET.py'
Elements in {32, 12, 45, 14, 76} before updating
Elements in {32, 98, 67, 12, 45, 14, 76} after updating
Elements in {98, 67, 12, 45, 14, 76} after using pop
Elements in {98, 67, 12, 45, 14} after using discard
Elements in {67, 12, 45, 14} after using remove
PS D:\Misbah\P>
```

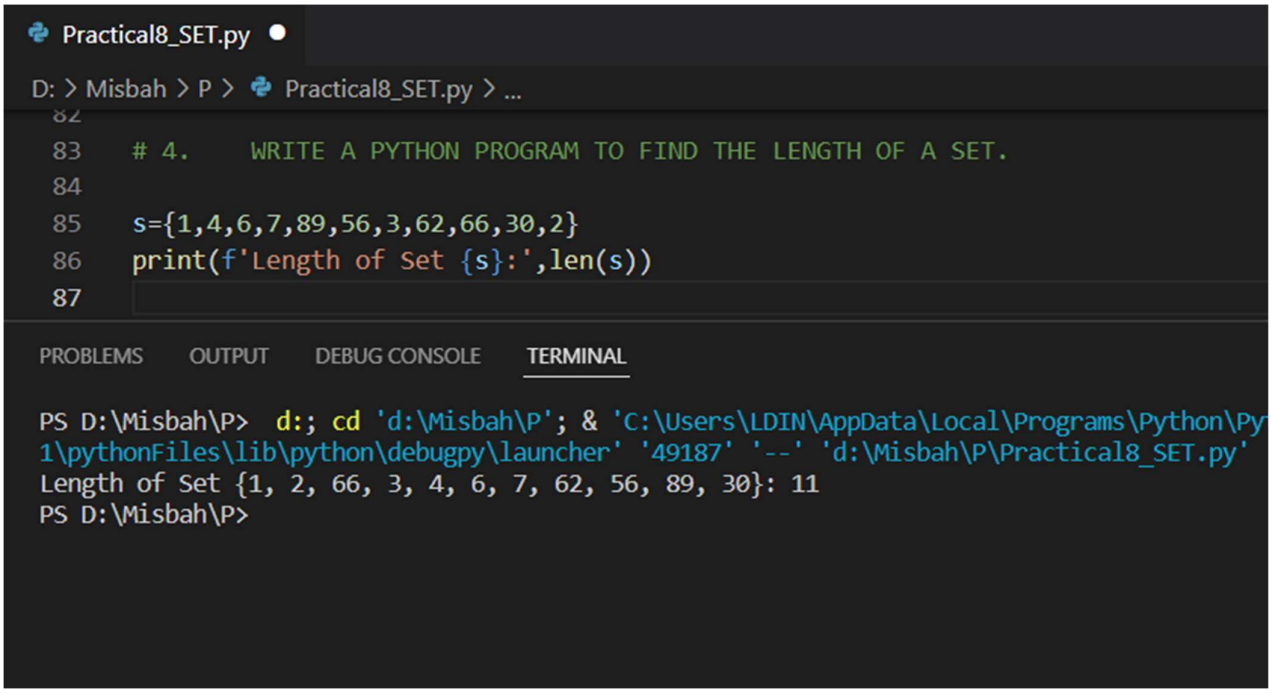
2. WRITE A PYTHON PROGRAM TO PERFORM FOLLOWING OPERATIONS ON SET: INTERSECTION OF SETS, UNION OF SETS, SET DIFFERENCE, SYMMETRIC DIFFERENCE, CLEAR A SET.

```
Practical8_SET.py
D: > Misbah > P > Practical8_SET.py > ...
 83    A = {0, 2, 4, 6, 8}
 84    B = {1, 2, 3, 4, 5}
 85    print("Union :", A | B)
 86    print("Intersection :", A & B)
 87    print("Difference :", A - B)
 88    print("Symmetric difference :", A ^ B)
 89

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Copyright (C) Microsoft Corporation. All rights reserved.

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

PS D:\Misbah\P>  & 'C:\Users\LDIN\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\LDIN\.vsc
on\debugpy\launcher' '65494' '--' 'd:\Misbah\P\Practical8_SET.py'
Union : {0, 1, 2, 3, 4, 5, 6, 8}
Intersection : {2, 4}
Difference : {0, 8, 6}
Symmetric difference : {0, 1, 3, 5, 6, 8}
PS D:\Misbah\P>
```

3. WRITE A PYTHON PROGRAM TO FIND MAXIMUM AND THE MINIMUM VALUE IN A SET.



4. WRITE A PYTHON PROGRAM TO FIND THE LENGTH OF A SET.



PRACTICAL NO. 9:
WRITE PYTHON PROGRAM TO PERFORM FOLLOWING OPERATIONS ON DICTIONARIES: CREATE DICTIONARY, ACCESS DICTIONARY ELEMENTS, UPDATE DICTIONARY, DELETE DICTIONARY, LOOPING THROUGH DICTIONARY

Dictionaries are associative data structures in Python

- It is not always possible to remember the index value of each element
- Hence, Dictionaries help us to identify an element with the help of its key
- Dictionaries contain key: values pair and each pair are separated by commas

CREATING A DICTIONARY

- We can create a dictionary by enclosing *key: value* pairs in curly braces

SYNTAX

dict1= {key1: val1, key2: val2, key3: val3,….., keyn: valn}

- We can directly use dict() function

SYNTAX

dict1= ({key1: val1, key2: val2, key3: val3,….., keyn: valn})

MISBAH PRACTICAL 9.py

D: > Misbah > P > MISBAH PRACTICAL 9.py > ...

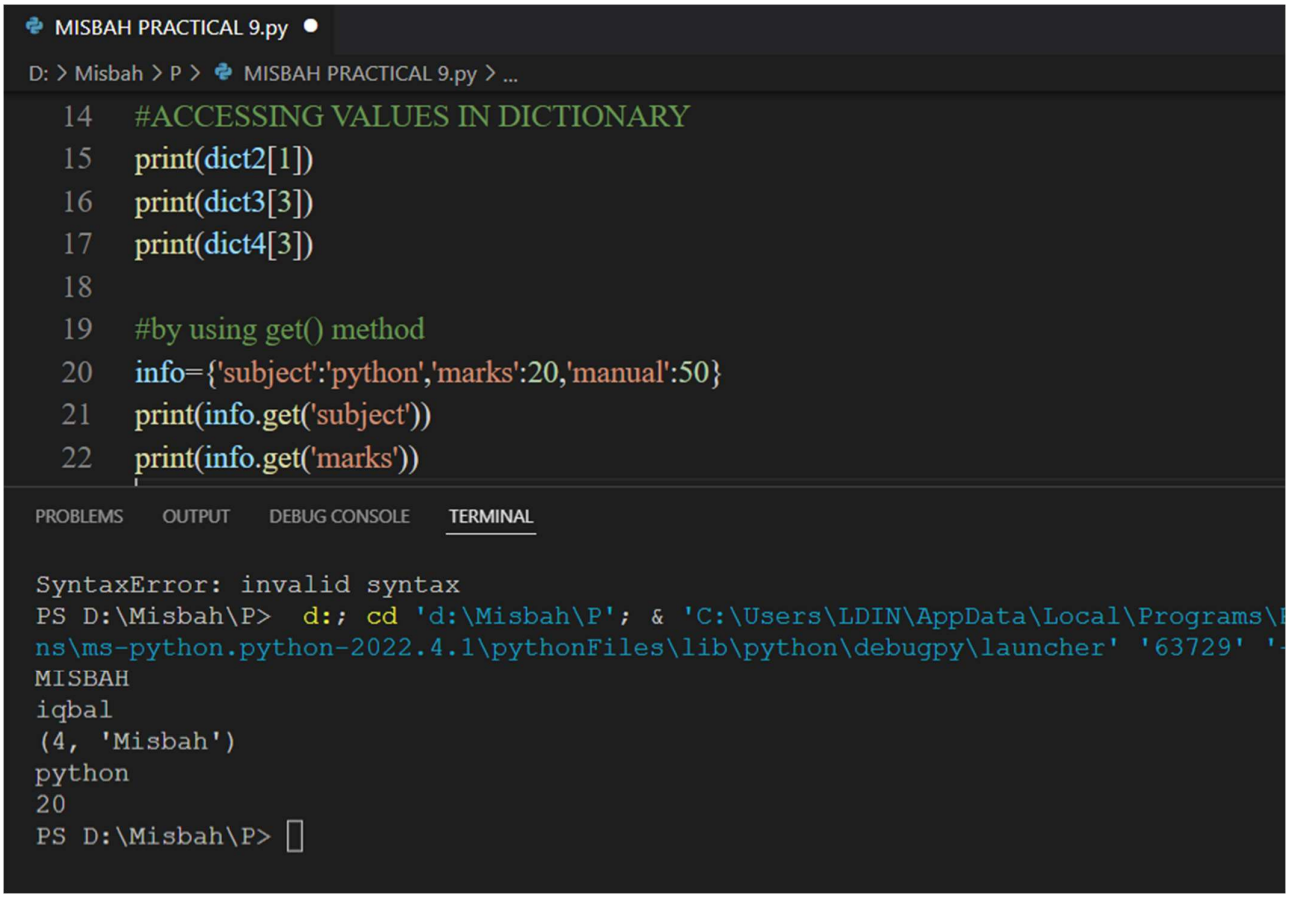
```
1  #CREATING A DICTIONARY
2  dict1 = {}  # empty dictionary
3  dict2 = {1: 'MISBAH', 2: 'BAGWAN'}  # integer dictionary
4  # by using dict() method
5  dict3 = dict({1:'bagwan', 2:'misbah',3:'iqbal'})
6  # from sequence having each item as a pair
7  dict4 =([(1,'My'),(2,'Name'),(3,'Is'),(4,'Misbah')])
8  print(dict1)
9  print(dict2)
10 print(dict3)
11 print(dict4)
12 print(type(dict3))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Misbah\P>  d:; cd 'd:\Misbah\P'; & 'C:\Users\LDIN\AppData\Local\Programs\Python\
.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '59444' '--' 'd:\Misbah\P\MI
{}
{1: 'MISBAH', 2: 'BAGWAN'}
{1: 'bagwan', 2: 'misbah', 3: 'iqbal'}
[(1, 'My'), (2, 'Name'), (3, 'Is'), (4, 'Misbah')]
<class 'dict'>
PS D:\Misbah\P>
```

ACCESSING VALUES IN DICTIONARY

- We can refer to its key name, inside square brackets[]
- We can also use get() method to access value of a key



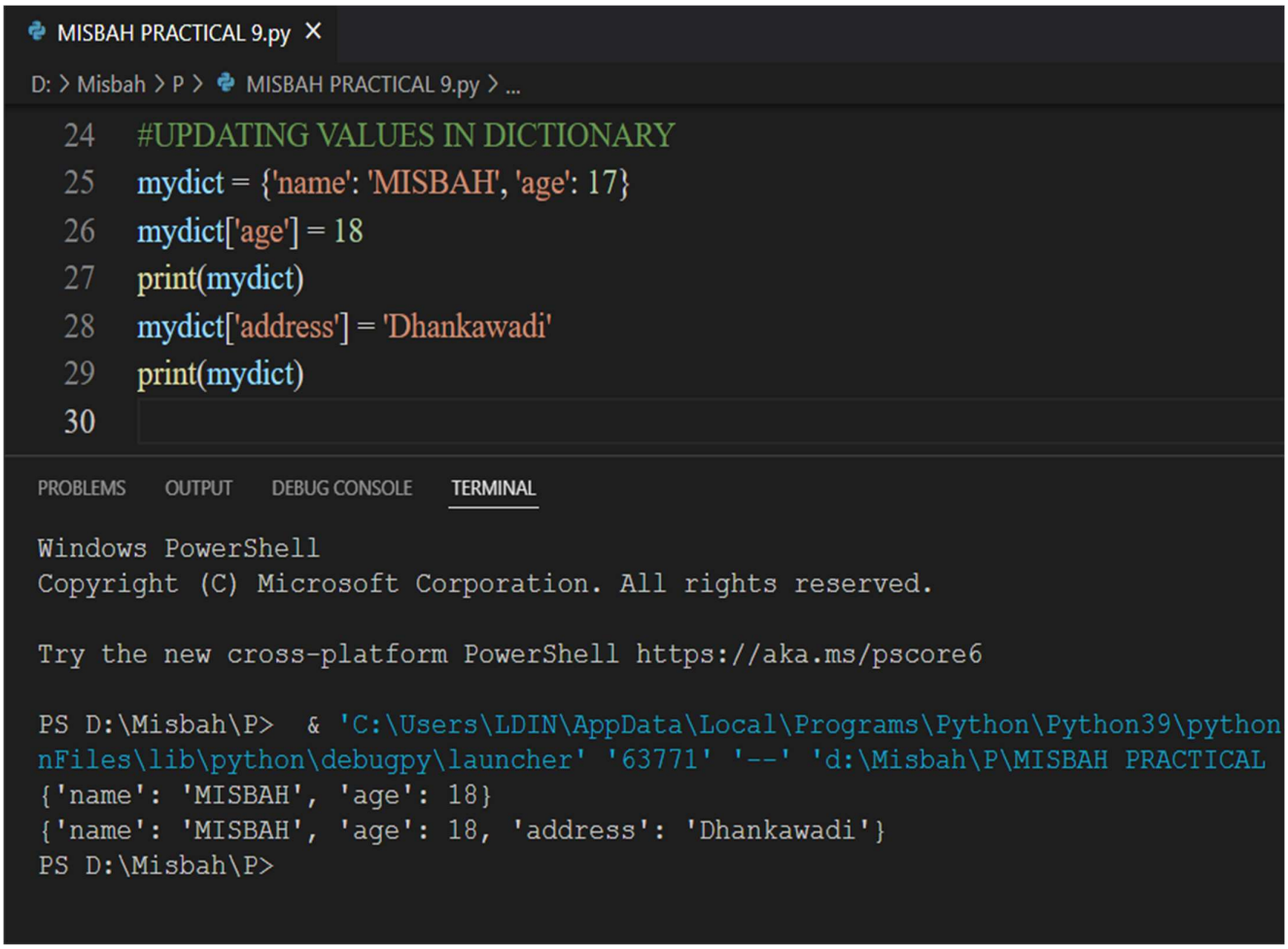
UPDATING DICTIONARY

- Dictionaries are mutable datatypes meaning they can be modified accordingly
- We can add new items or change the value of existing items using an **assignment operator**.
- If the key is already present, then the existing value gets updated. In case the key is not present, a new (**key: value**) pair is added to the dictionary.

SYNTAX

dict1[existing_key] = diff_value

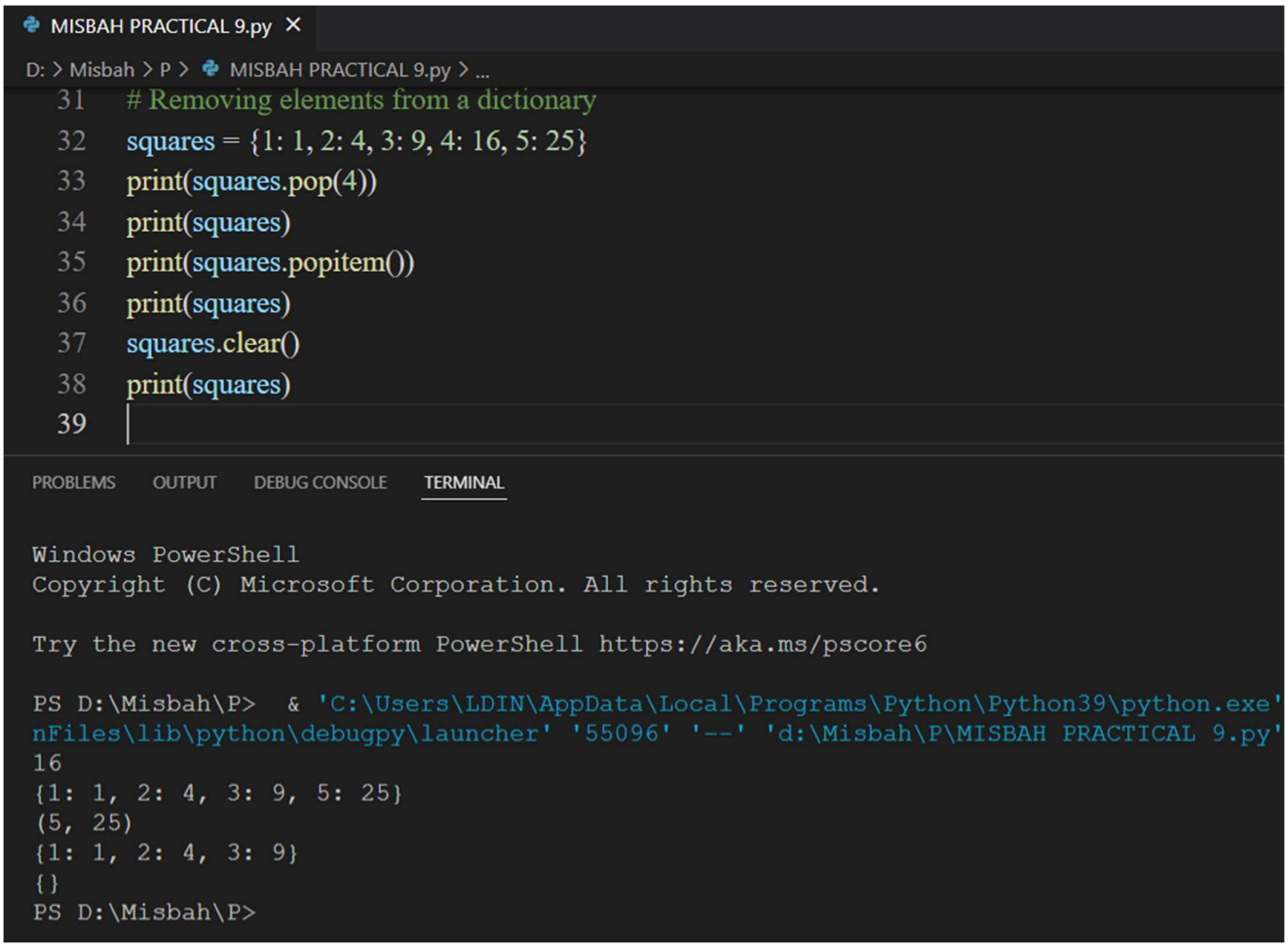
dict1[new_key]=new_key



REMOVING ELEMENTS FROM DICTIONARY

With the help of the following methods, it is easy to remove an item from the dictionary by specifying its key, deleting the last key: value pair, emptying the entire dictionary by removing all elements (clearing it) and deleting the entire dictionary

- pop(key): removes specified key: value pair from the dictionary
- popitem(): removes last element from the dictionary
- clear();removes all pairs from the dictionary
- del keyword: deletes the entire dictionary



BUILT IN FUNCTIONS AND METHODS IN DICTIONARY

Function	Description
all()	Return True if all keys of the dictionary are True (or if the dictionary is empty).
any()	Return True if any key of the dictionary is true. If the dictionary is empty, return False.
len()	Return the length (the number of items) in the dictionary.
Sorted()	Return a new sorted list of keys in the dictionary.

EXAMPLE

MISBAH PRACTICAL 9.py

D: > Misbah > P > MISBAH PRACTICAL 9.py > ...

```
30
31  # Dictionary Built-in Functions
32  squares = {0: 0, 1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
33  print(all(squares))
34  print(any(squares))
35  print(len(squares))
36  print(sorted(squares))
37
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Misbah\P>  d:; cd 'd:\Misbah\P'; & 'C:\Users\LDIN\AppData\Local\Programs\
.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '63421' '--' 'd:\Misb
False
True
6
[0, 1, 3, 5, 7, 9]
PS D:\Misbah\P>
```

LOOPING THROUGH DICTIONARY

A dictionary can be iterated using the for loop.

MISBAH PRACTICAL 9.py

D: > Misbah > P > MISBAH PRACTICAL 9.py > ...

```
38  month={1:'sun',2:'mon',3:'tue',4:'wed',5:'thurs',6:'fri',7:'sat'}
39  for key,value in month.items():
40      print(key,'-',value)
41
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\LDIN> & C:/Users/LDIN/AppData/Local/Programs/Python/Python39/python
1 - sun
2 - mon
3 - tue
4 - wed
5 - thurs
6 - fri
7 - sat
PS C:\Users\LDIN>
```

XI. EXERCISE

1. WRITE A PYTHON SCRIPT TO SORT (ASCENDING AND DESCENDING) A DICTIONARY BY VALUE.

EXP9A.py - C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9A.py (3.9.10)

File Edit Format Run Options Window Help

```
d={"One":1, "Three": 3, "Zero": 0, "Five": 5, "Four":4}
print(fDictionary before sorting:{d}')
li=list(d.items())
l=len(li)
for i in range(l-1):
    for j in range(i+1,l):
        if li[i][1]>li[j][1]:
            t=li[i]
            li[i]=li[j]
            li[j]=t
    sortd=dict(li)
print(fDictionary after sorting by value:{sortd}')
```

IDLE Shell 3.9.10

File Edit Shell Debug Options Window Help

```
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9A.py ====
Dictionary before sorting:{'One': 1, 'Three': 3, 'Zero': 0, 'Five': 5, 'Four': 4}
Dictionary after sorting by value:{'Zero': 0, 'One': 1, 'Three': 3, 'Four': 4, 'Five': 5}
>>>
```

2. WRITE A PYTHON SCRIPT TO CONCATENATE FOLLOWING DICTIONARIES TO CREATE A NEW ONE.

a. Sample Dictionary: b. dic1 = {1:10, 2:20} c. dic2 = {3:30, 4:40} d. dic3 = {5:50,6:60}

EXP9B.py - C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9B.py (3.9.10)

File Edit Format Run Options Window Help

```
d1={1:10, 2:20}
d2={3:30, 4:40}
d3={5:50,6:60}
d4 = {}
for i in (d1, d2, d3):
    d4.update(i)
print(d4)
```

IDLE Shell 3.9.10

File Edit Shell Debug Options Window Help

```
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9B.py ====
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
>>>
```

3. WRITE A PYTHON PROGRAM TO COMBINE TWO DICTIONARY ADDING VALUES FOR COMMON KEYS.
    a. D1 = {'A': 100, 'B': 200, 'C':300}
    b. D2 = {'A': 300, 'B': 200, 'D':400}

EXP9C.py - C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9C.py (3.9.10)

File Edit Format Run Options Window Help

```
d1 = {'a': 100, 'b': 200, 'c': 300}
d2 = {'a': 300, 'b': 200, 'd': 400}
for key in d2:
   if key in d1:
      d2[key] = d2[key] + d1[key]
   else:
      pass
res = d1 | d2
print(res)
```

IDLE Shell 3.9.10

File Edit Shell Debug Options Window Help

```
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/LDIN/AppData/Local/Programs/Python/
{'a': 400, 'b': 400, 'c': 300, 'd': 400}
>>>
```

4. WRITE A PYTHON PROGRAM TO PRINT ALL UNIQUE VALUES IN A DICTIONARY.

A. SAMPLE DATA: [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

EXP9D.py - C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9D.py (3.9.10)

File Edit Format Run Options Window Help

```
L = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]
print("Original List: ",L)
u_value = set( val for dic in L for val in dic.values())
print("Unique Values: ",u_value)
```

IDLE Shell 3.9.10

File Edit Shell Debug Options Window Help

```
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9D.py ====
Original List:  [{'V': 'S001'}, {'V': 'S002'}, {'VI': 'S001'}, {'VI': 'S005'}, {'VII': 'S005'}, {'V': 'S009'}, {'VIII': 'S007'}]
Unique Values:  {'S009', 'S007', 'S001', 'S002', 'S005'}
>>>
```

4. WRITE A PYTHON PROGRAM TO FIND THE HIGHEST 3 VALUES IN A DICTIONARY.

EXP9E.py - C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9E.py (3.9.10)

File Edit Format Run Options Window Help

```
d = {"A" : 105 , "B" : 274 , "C" : 13 , "D" : 41 ,"E" : 152 }
l = list( d.values() )
lr = l [ 0 ]
se = l[ 0 ]
tr=l[ 0 ]
for i in l :
    if lr <= i :
        lr = i
    elif se < i :
        if lr != i:
            se = i
    elif tr < i :
        if se != i :
            tr = i
print(fHighest three values in {d} are {lr},{se},{tr}')
```

IDLE Shell 3.9.10

File Edit Shell Debug Options Window Help

```
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/LDIN/AppData/Local/Programs/Python/Python39/EXP9E.py ====
Highest three values in {'A': 105, 'B': 274, 'C': 13, 'D': 41, 'E': 152} are 274,152,105
>>>
```

PRACTICAL NO. 10:

- WRITE PYTHON PROGRAM TO DEMONSTRATE MATH BUILT-IN FUNCTIONS (ANY 2 PROGRAMS)
- WRITE PYTHON PROGRAM TO DEMONSTRATE STRING BUILT-IN FUNCTIONS (ANY 2 PROGRAMS)

PYTHON MATH FUNCTIONS

TO USE THE MATH MODULE IN YOUR PYTHON APPLICATION, FIRST, YOU NEED TO IMPORT THE MODULE USING **IMPORT MATH** STATEMENTS IN YOUR PROGRAM.

main.py

```
import math
data = 21.6
print('The floor of 21.6 is:', math.floor(data))
print(math.ceil(21.6))
print('The value of PI:',math.pi)
print(math.pow(4, 2))
number = -21.19
print('The given number is :', number)
print('Floor value is :', math.floor(number))
print('Ceiling value is :', math.ceil(number))
print('Absolute value is :', math.fabs(number))
number = 1e-4
print('The given number (x) is :', number)
print('e^x (using exp() function) is :', math.exp(number)-1)
print('log(fabs(x), base) is :', math.log(math.fabs(number), 10))
angleInDegree = 90
angleInRadian = math.radians(angleInDegree)
print('The given angle is :', angleInRadian)
print('sin(x) is :', math.sin(angleInRadian))
print('cos(x) is :', math.cos(angleInRadian))
print('tan(x) is :', math.tan(angleInRadian))
```

main.py

Run: main

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProjects/
The floor of 21.6 is: 21
22
The value of PI: 3.141592653589793
16.0
The given number is : -21.19
Floor value is : -22
Ceiling value is : -21
Absolute value is : 21.19
The given number (x) is : 0.0001
e^x (using exp() function) is : 0.0001000050001667141
log(fabs(x), base) is : -3.999999999999999
The given angle is : 1.5707963267948966
sin(x) is : 1.0
cos(x) is : 6.123233995736766e-17
tan(x) is : 1.633123935319537e+16

Process finished with exit code 0
```

PYTHON STRING FUNCTIONS

Method	*Description*
center()	**Returns a centred string**
endswith()	**Returns true if the string ends with the specified value**
find()	**Searches the string for a specified value and returns the position of where it was found**
isalpha()	**Returns True if all characters in the string are in the alphabet**
islower()	**Returns True if all characters in the string are lower case**
isspace()	**Returns True if all characters in the string are whitespaces**
isupper()	**Returns True if all characters in the string are upper case**
ljust()	**Returns a left justified version of the string**
lower()	**Converts a string into lower case**
replace()	**Returns a string where a specified value is replaced with a specified value**
rfind()	**Searches the string for a specified value and returns the last position of where it was found**
rjust()	**Returns a right justified version of the string**
startswith()	**Returns true if the string starts with the specified value**
title()	**Converts the first character of each word to upper case**
upper()	**Converts a string into upper case**

main.py

```
text = 'miSBaH  iQBal  bAGWAn'
print("UPPER : ", text.upper())
print("LOWER : ", text.lower())
print("TITLE : ", text.title())
str = "I SAW A ROSE IN A GLASS VASE THROUGH THE ROSE TINTED GLASS WINDOW"; str2 = "ROSE"; str3 = "GLASS"
print("The first occurrence of ROSE and GLASS is at : ", end="")
print("\n", str.find(str2, 4))
print("\n", str.find(str3, 4))
print("The last occurrence of ROSE and GLASS is at : ", end="")
print("\n", str.rfind(str2, 4))
print("\n", str.rfind(str3, 4))
str3 = "Bagwan"
str4 = "My name is Misbah Bagwan"
if str4.startswith(str3):
    print("str1 begins with : " + str3)
else:
    print("str1 does not begin with : " + str3)
if str4.endswith(str3):
    print("str1 ends with : " + str)
else:
    print("str1 does not end with : " + str)
str5 = "MISBAH BAGWAN"
str6 = "misbah bagwan"
if str5.isupper():
```

main.py

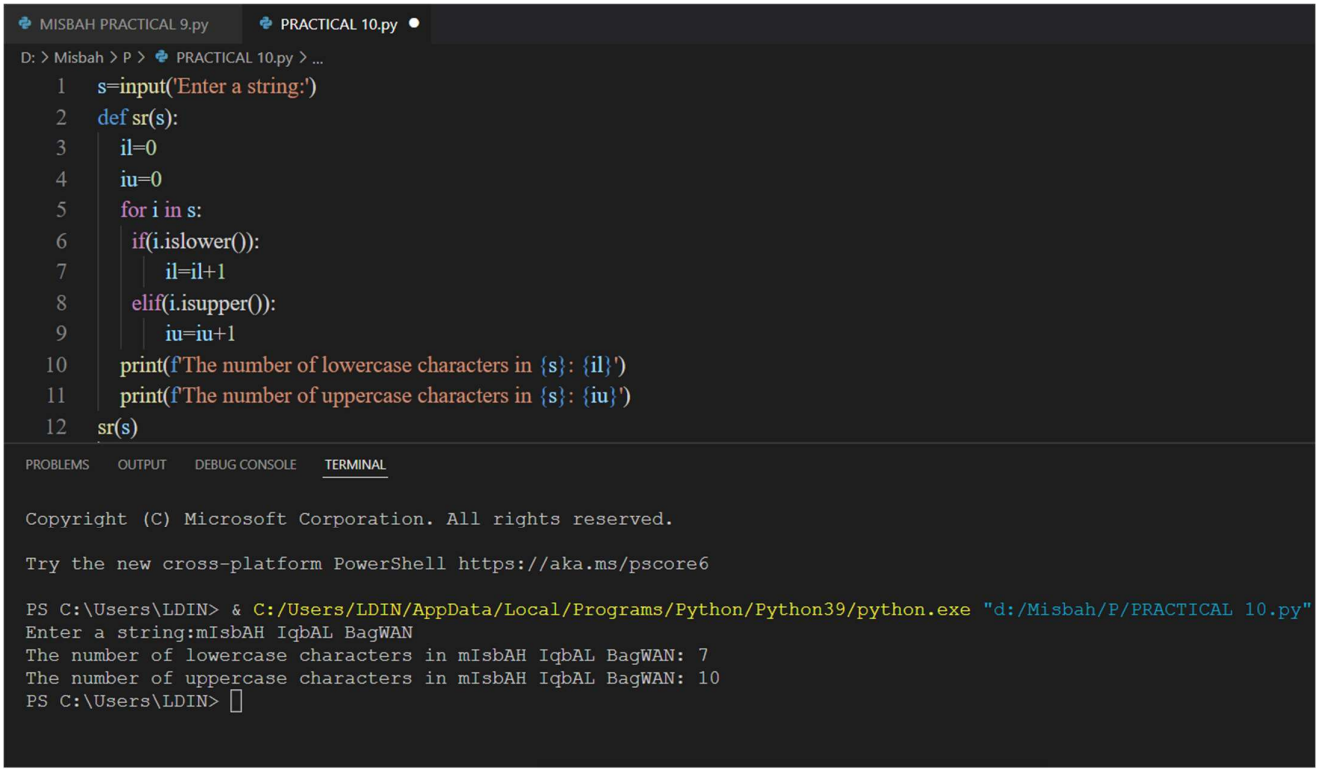
```
    print("All characters in str are upper cased")
else:
    print("All characters in str are not upper cased")
if str6.islower():
    print("All characters in str1 are lower cased")
else:
    print("All characters in str1 are not lower cased")
str7 = "My name is Misbah Bagwan"
print(" The length of string is : ", len(str7))
str8 = "Python is an easy programming language"
print("The string after centering with '-' is : ", end="")
print(str8.center(20, '-'))
print("The string after ljust is : ", end="")
print(str8.ljust(20, '-'))
print("The string after rjust is : ", end="")
print(str8.rjust(20, '-'))
str9 = "PYTHON MAD ETI "
str10 = " 22616"
print(str.isalpha(),str10.isalnum(),str10.isspace())
str11 = "i took the elevator and entered the flat"
str12 = "lift"
str13 = "elevator"
print("The string after replacing strings is : ", end="")
print (str11.replace( str12, str12, 2))
```

main

```
C:\Users\LDIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/LDIN/PycharmProjects/pythonProject/main.py
UPPER :  MISBAH  IQBAL  BAGWAN
LOWER :  misbah  iqbal  bagwan
TITLE :  Misbah  Iqbal  Bagwan
The first occurrence of ROSE and GLASS is at :
 8

 18
The last occurrence of ROSE and GLASS is at :
 41

 53
str1 does not begin with : Bagwan
str1 ends with : I SAW A ROSE IN A GLASS VASE THROUGH THE ROSE TINTED GLASS WINDOW
All characters in str are upper cased
All characters in str1 are lower cased
 The length of string is :  24
The string after centering with '-' is : Python is an easy programming language
The string after ljust is : Python is an easy programming language
The string after rjust is : Python is an easy programming language
False False False
The string after replacing strings is : i took the elevator and entered the flat

Process finished with exit code 0
```

XI. EXERCISE

1. WRITE A PYTHON FUNCTION THAT ACCEPTS A STRING AND CALCULATE THE NUMBER OF UPPER-CASE LETTERS AND LOWER-CASE LETTERS.



2. WRITE A PYTHON PROGRAM TO GENERATE A RANDOM FLOAT WHERE THE VALUE IS BETWEEN 5 AND 50 USING PYTHON MATH MODULE.