

FUNCTION

1. A function that returns the argument number's square when called.

The screenshot shows a terminal window with a dark background. At the top, there is a blue header bar with the text "main.py". The main area contains the following Python code:

```
1 def square( num ):
2     """
3         This function computes the square of the number.
4     """
5     return num**2
6 object_ = square(6)
7 print( "The square of the given number is: ", object_ )
```

Below the code, the terminal displays the output:

```
The square of the given number is: 36
```

At the bottom of the terminal window, there is a message indicating the program has finished and prompting the user to press Enter to exit:

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

2. Example Python Code for calling a function.

The screenshot shows a terminal window with a dark background. At the top, there is a blue header bar with the text "main.py". The main area contains the following Python code:

```
1 def a_function( string ):
2     "This prints the value of length of string"
3     return len(string)
4 print( "Length of the string Functions is: ", a_function( "Functions" ) )
5 print( "Length of the string Python is: ", a_function( "Python" ) )
```

Below the code, the terminal displays the output:

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

At the bottom of the terminal window, there is a message indicating the program has finished and prompting the user to press Enter to exit:

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

3. Python Code for Pass by Reference vs. Value

The screenshot shows a terminal window with two parts. The top part displays the Python code in a file named 'main.py'. The bottom part shows the terminal output.

```
main.py
1 def square( item_list ):
2     '''This function will find the square of items in the list'''
3     squares = [ ]
4     for l in item_list:
5         squares.append( l**2 )
6     return squares
7 my_list = [17, 52, 8];
8 my_result = square( my_list )
9 print( "Squares of the list are: ", my_result )
```

Terminal Output:

```
Squares of the list are: [289, 2704, 64]
...Program finished with exit code 0
Press ENTER to exit console
```

4. Python code to demonstrate the use of default arguments

The screenshot shows a terminal window with two parts. The top part displays the Python code in a file named 'main.py'. The bottom part shows the terminal output.

```
main.py
1 def function( n1, n2 ):
2     print("number 1 is: ", n1)
3     print("number 2 is: ", n2)
4 # Calling function and passing arguments without using keyword
5 print( "Without using keyword" )
6 function( 50, 30 )
7 print( "With using keyword" )
8 function( n2 = 50, n1 = 30 )
```

Terminal Output:

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

5. Python code to demonstrate the use of keyword arguments

The screenshot shows a terminal window with two parts. The top part displays the contents of a file named 'main.py' with the following code:

```
main.py
1- def function( n1, n2 = 20 ):
2-     print("number 1 is: ", n1)
3-     print("number 2 is: ", n2)
4- # Calling the function and passing only one argument
5- print( "Passing only one argument" )
6- function(30)
7- print( "Passing two arguments" )
8- function(50,30)
```

The bottom part shows the terminal output with the command 'input' at the prompt. The output is:

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

6. Python code to demonstrate the use of default arguments

The screenshot shows a terminal window with two parts. The top part displays the contents of a file named 'main.py' with the following code:

```
main.py
1- def function( n1, n2 ):
2-     print("number 1 is: ", n1)
3-     print("number 2 is: ", n2)
4-
5- print( "Passing out of order arguments" )
6- function( 30, 20 )
7-
8- # Calling function and passing only one argument
9- print( "Passing only one argument" )
10- try:
11-     function( 30 )
12- except:
13-     print( "Function needs two positional arguments" )
```

The bottom part shows the terminal output with the command 'input' at the prompt. The output is:

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

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

7. Python code to demonstrate the use of variable-length arguments

The screenshot shows a terminal window with the title bar "main.py". The code defines two functions: one for args and one for kwargs. It then creates an object and prints it. The output shows the object as a list of lists, where each inner list contains a key and its corresponding value. The terminal prompt is "input" and ends with "...Program finished with exit code 0 Press ENTER to exit console."

```
1 def function( *args_list ):
2     ans = []
3     for l in args_list:
4         ans.append( l.upper() )
5     return ans
6 # Passing args arguments
7 object = function('Python', 'Functions', 'tutorial')
8 print( object )
9 # defining a function
10 def function( **kwargs_list ):
11     ans = []
12     for key, value in kwargs_list.items():
13         ans.append([key, value])
14     return ans
15 # Paasing Kwargs arguments
16 object = function(first = "Python", Second = "Functions", Third = "Tutorial")
17 print(object)
```

```
['PYTHON', 'FUNCTIONS', 'TUTORIAL']
[['First', 'Python'], ['Second', 'Functions'], ['Third', 'Tutorial']]
...Program finished with exit code 0
Press ENTER to exit console.
```

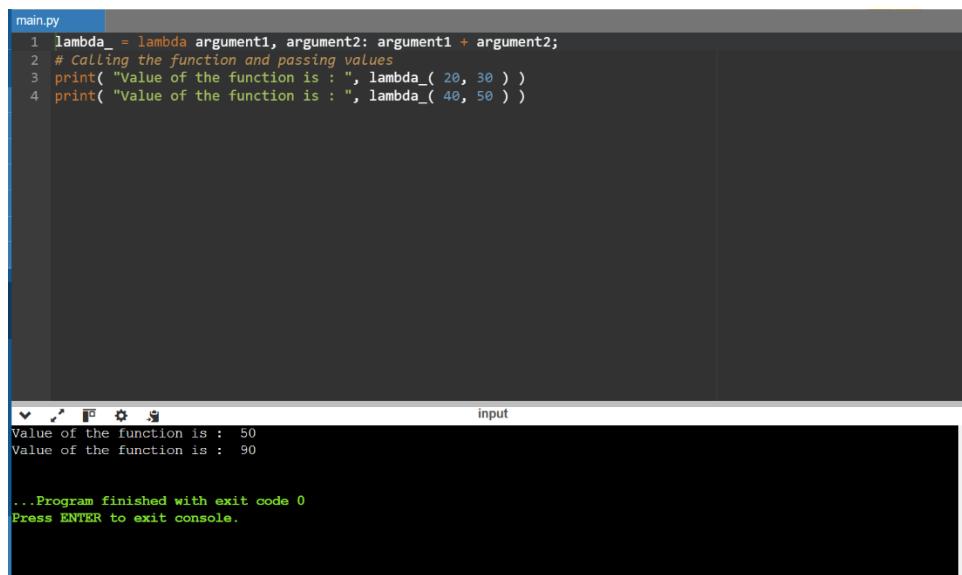
8. Python code to demonstrate the use of return statements

The screenshot shows a terminal window with the title bar "main.py". The code defines a square function with and without a return statement. It then prints the results of calling the function with 52 as an argument. The output shows the result of the function with a return statement (2704) and the result of the function without a return statement (None). The terminal prompt is "input" and ends with "...Program finished with exit code 0 Press ENTER to exit console."

```
1 def square( num ):
2     return num**2
3 # Calling function and passing arguments.
4 print( "With return statement" )
5 print( square( 52 ) )
6 # Defining a function without return statement
7 def square( num ):
8     num**2
9
10 # Calling function and passing arguments.
11 print( "Without return statement" )
12 print( square( 52 ) )
```

```
With return statement
2704
Without return statement
None
...Program finished with exit code 0
Press ENTER to exit console.
```

9. Python code to demonstrate anonymous functions

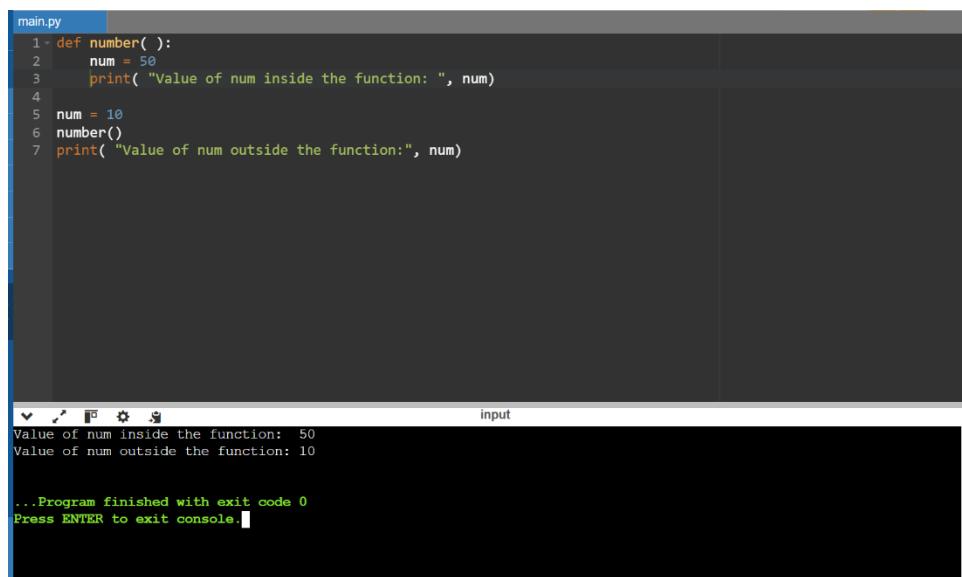


```
main.py
1 lambda_ = lambda argument1, argument2: argument1 + argument2;
2 # Calling the function and passing values
3 print( "Value of the function is : ", lambda_( 20, 30 ) )
4 print( "Value of the function is : ", lambda_( 40, 50 ) )

Value of the function is : 50
Value of the function is : 90

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

10. Python code to demonstrate scope and lifetime of variable



```
main.py
1 def number( ):
2     num = 50
3     print( "Value of num inside the function: ", num)
4
5 num = 10
6 number()
7 print( "Value of num outside the function:", num)

Value of num inside the function: 50
Value of num outside the function: 10

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

11. Python code to show how to access variables of a nested functions

The screenshot shows a terminal window with two sections. The top section is labeled 'main.py' and contains the following Python code:

```
1 - def word():
2     string = 'Python functions tutorial'
3     x = 5
4     def number():
5         print( string )
6         print( x )
7
8     number()
9 word()
```

The bottom section is labeled 'input' and shows the output of running the code. It displays the string 'Python functions tutorial' followed by the integer '5'. Below the output, the terminal prompt shows the message "...Program finished with exit code 0 Press ENTER to exit console.".

12. Python abs() Function Example

The screenshot shows a terminal window with two sections. The top section is labeled 'main.py' and contains the following Python code:

```
1 integer = -20
2 print('Absolute value of -40 is:', abs(integer))
3
4 # floating number
5 floating = -20.83
6 print('Absolute value of -40.83 is:', abs(floating))
```

The bottom section is labeled 'input' and shows the output of running the code. It displays the absolute value of -20 as 20 and the absolute value of -40.83 as 20.83. Below the output, the terminal prompt shows the message "...Program finished with exit code 0 Press ENTER to exit console.".

13. Python all() Function Example

```
main.py
1 k = [1, 3, 4, 6]
2 print(all(k))
3 # all values false
4 k = [0, False]
5 print(all(k))
6 # one false value
7 k = [1, 3, 7, 0]
8 print(all(k))
9 # one true value
10 k = [0, False, 5]
11 print(all(k))
12 # empty iterable
13 k = []
14 print(all(k))

input
True
False
False
False
True

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

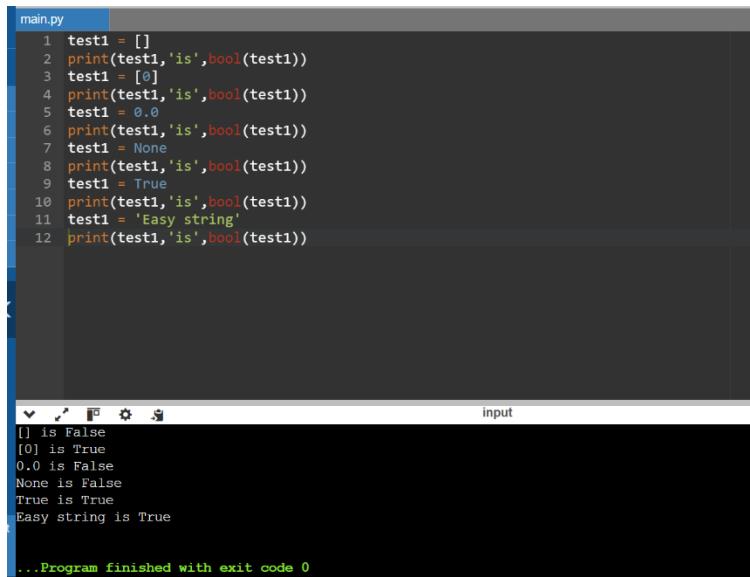
14. Python bin() Function Example

```
main.py
1 x=10
2 y=bin(x)
3 print(y)
4

input
0b1010

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

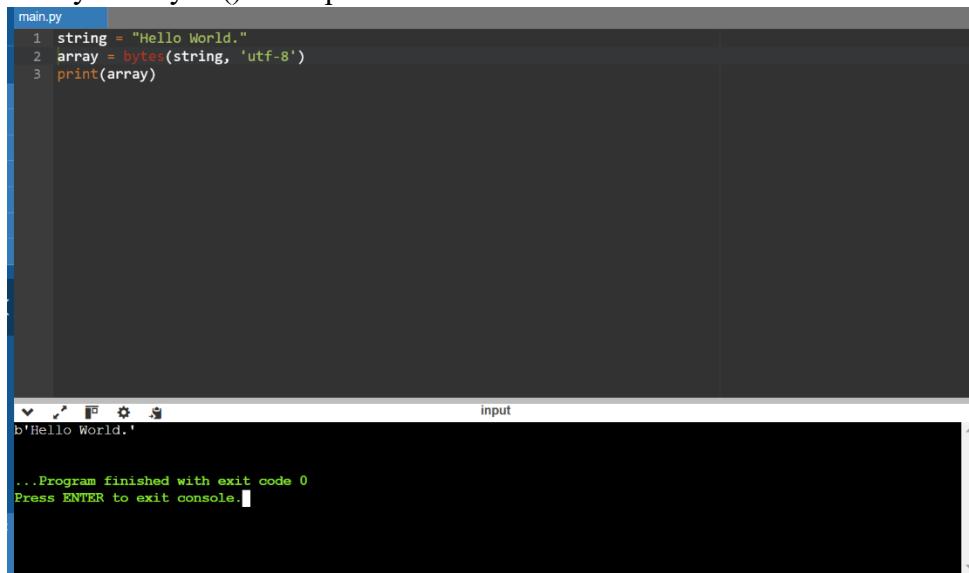
15. Python bool() Example



```
main.py
1 test1 = []
2 print(test1,'is',bool(test1))
3 test1 = [0]
4 print(test1,'is',bool(test1))
5 test1 = 0.0
6 print(test1,'is',bool(test1))
7 test1 = None
8 print(test1,'is',bool(test1))
9 test1 = True
10 print(test1,'is',bool(test1))
11 test1 = 'Easy string'
12 print(test1,'is',bool(test1))

...Program finished with exit code 0
```

16. Python bytes() Example



```
main.py
1 string = "Hello World."
2 array = bytes(string, 'utf-8')
3 print(array)

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

17. Python callable() Function Example

The screenshot shows a terminal window with the title bar "main.py". The code in the editor is:

```
1 x= 8
2 print(callable(x))
3
```

The terminal output is:

```
input
False

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

18. Python compile() Function Example

The screenshot shows a terminal window with the title bar "main.py". The code in the editor is:

```
1 code_str = 'x=5\ny=10\nprint("sum =",x+y)'
2 code = compile(code_str, 'sum.py', 'exec')
3 print(type(code))
4 exec(code)
5 exec(x)
```

The terminal output is:

```
input
<class 'code'>
sum = 15
Traceback (most recent call last):
  File "/home/main.py", line 5, in <module>
    exec(x)
    ^^^^^^
TypeError: exec() arg 1 must be a string, bytes or code object
```

19. Python exec() Function Example



```
main.py
1 x = 8
2 exec('print(x==8)')
3 exec('print(x+4)')

True
12

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

20. Python sum() Function Example

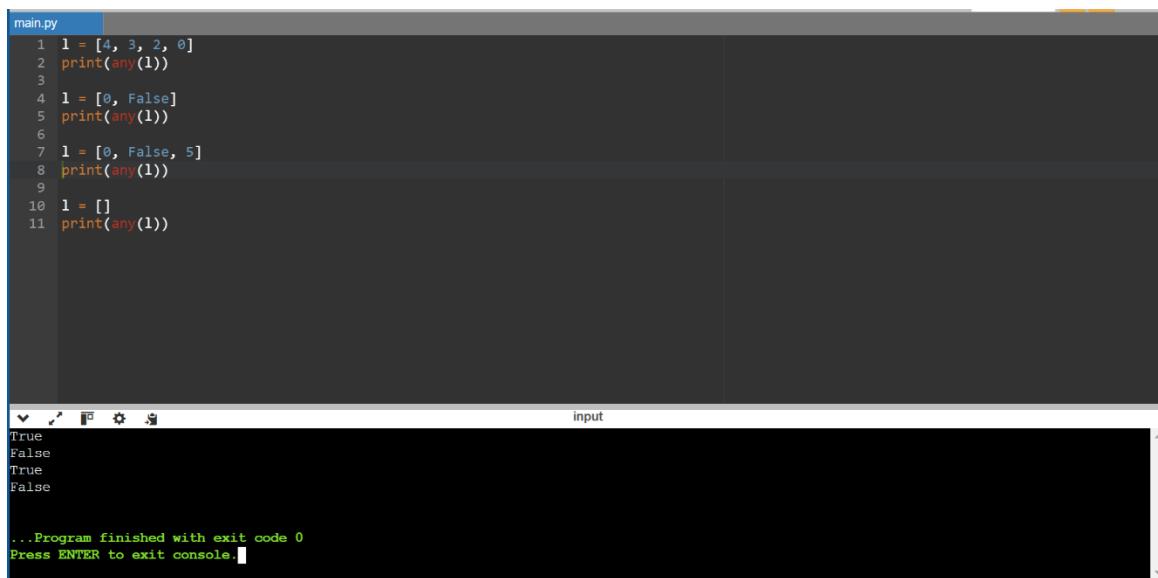


```
main.py
1 s = sum([1, 2, 4])
2 print(s)
3
4 s = sum([1, 2, 4], 10)
5 print(s)

7
17

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

21. Python any() Function Example

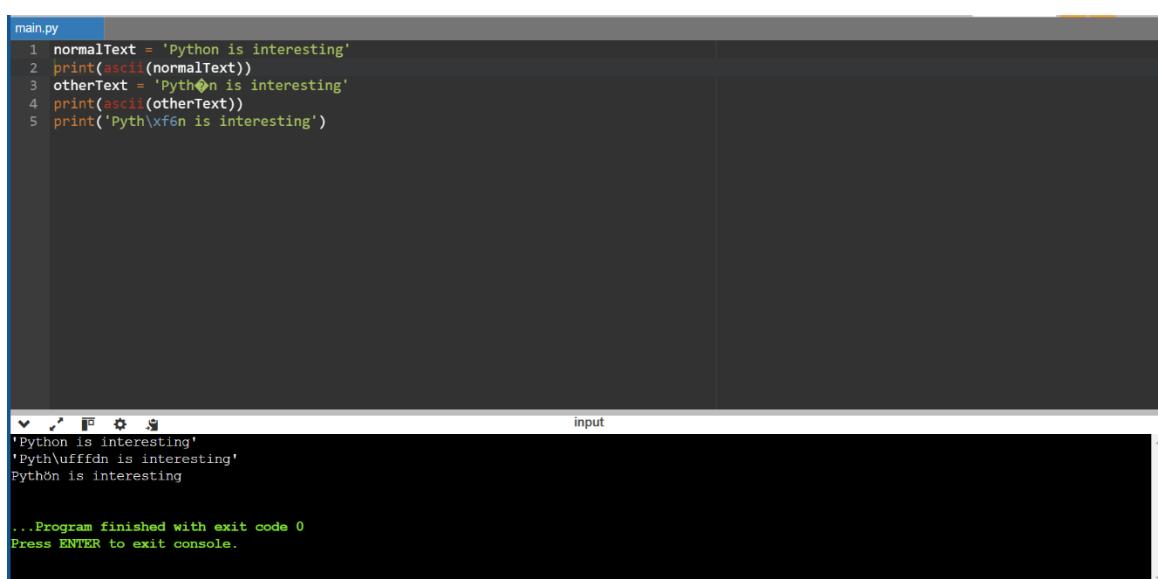


```
main.py
1 l = [4, 3, 2, 0]
2 print(any(l))
3
4 l = [0, False]
5 print(any(l))
6
7 l = [0, False, 5]
8 print(any(l))
9
10 l = []
11 print(any(l))

True
False
True
False

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

22. Python ascii() Function Example

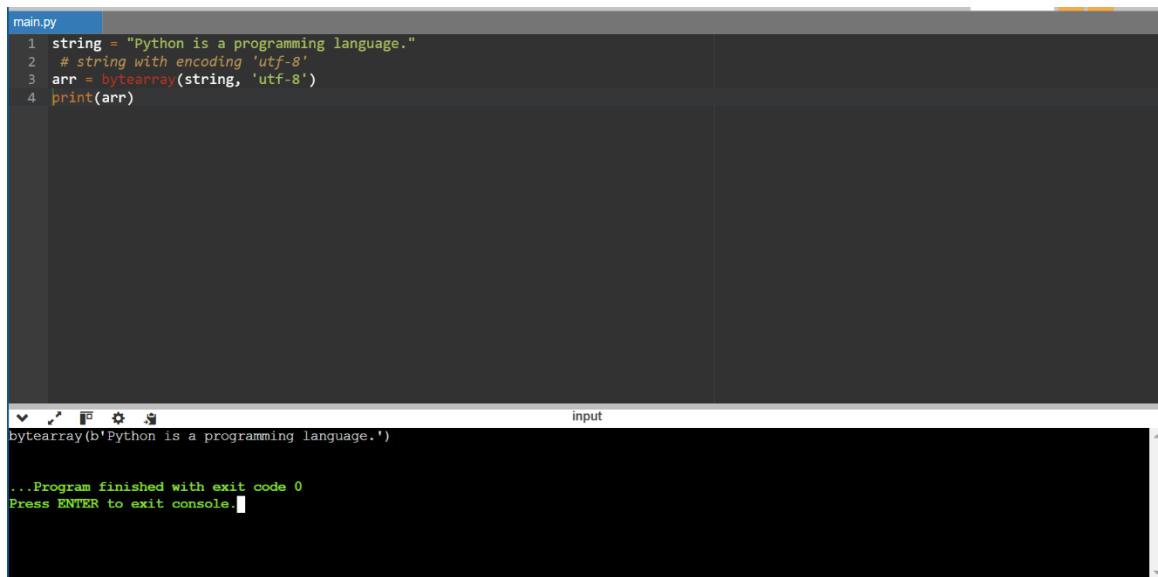


```
main.py
1 normalText = 'Python is interesting'
2 print(ascii(normalText))
3 otherText = 'Pyth n is interesting'
4 print(ascii(otherText))
5 print('Pyth\xf6n is interesting')

'Python is interesting'
'Pyth\xufffdn is interesting'
Pyth n is interesting

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

23. Python bytearray() Example



The screenshot shows a terminal window with two panes. The top pane displays the code in a file named 'main.py':

```
main.py
1 string = "Python is a programming language."
2 # string with encoding 'utf-8'
3 arr = bytearray(string, 'utf-8')
4 print(arr)
```

The bottom pane shows the terminal output:

```
bytearray(b'Python is a programming language.')
...Program finished with exit code 0
Press ENTER to exit console.
```

24. Python eval() Function Example



The screenshot shows a terminal window with two panes. The top pane displays the code in a file named 'main.py':

```
main.py
1 x = 8
2 print(eval('x+1'))
```

The bottom pane shows the terminal output:

```
9
...Program finished with exit code 0
Press ENTER to exit console.
```

25. Python float() Example

The screenshot shows a terminal window with the following content:

```
main.py
1 # for integers
2 print(float(9))
3
4 # for floats
5 print(float(8.19))
6
7 # for string floats
8 print(float("-24.27"))
9
10 # for string floats with whitespaces
11 print(float(" -17.19\n"))
12
13 # string float error
14 print(float("xyz"))

8.0
8.19
-24.27
-17.19
Traceback (most recent call last):
  File "/home/main.py", line 14, in <module>
    print(float("xyz"))
    ^^^^^^^^^^^^^^
ValueError: could not convert string to float: 'xyz'

...Program finished with exit code 1
Press ENTER to exit console.
```

26.

26. Python format() Function

The screenshot shows a terminal window with the following content:

```
main.py
1 print(format(123, "d"))
2
3 # float arguments
4 print(format(123.4567898, "f"))
5
6 # binary format
7 print(format(12, "b")) |
```

Output:

```
123
123.456790
1100

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

27. Python frozenset() Example

```
main.py
1 letters = ('m', 'r', 'o', 't', 's')
2 fset = frozenset(letters)
3 print('Frozen set is:', fset)
4 print('Empty frozen set is:', frozenset())

Frozen set is: frozenset({'o', 'r', 't', 's', 'm'})
Empty frozen set is: frozenset()

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

28. Python getattr() Function Example

```
main.py
1 - class Details:
2     age = 22
3     name = "Phill"
4
5 details = Details()
6 print('The age is:', getattr(details, "age"))
7 print('The age is:', details.age)

The age is: 22
The age is: 22

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

29. Python globals() Function Example

The screenshot shows a terminal window with two panes. The top pane displays the code in `main.py`:

```
main.py
1 age = 22
2
3 globals()['age'] = 22
4 print('The age is:', age)
```

The bottom pane shows the terminal output:

```
The age is: 22
input
...Program finished with exit code 0
Press ENTER to exit console.
```

30. Python hasattr() Function Example

The screenshot shows a terminal window with two panes. The top pane displays the code in `main.py`:

```
main.py
1 l = [4, 3, 2, 0]
2 print(any(l))
3
4 l = [0, False]
5 print(any(l))
6
7 l = [0, False, 5]
8 print(any(l))
9
10 l = []
11 print(any(l))
```

The bottom pane shows the terminal output:

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

31. Python iter() Function Example

```
main.py
1 # List of numbers
2 list = [1,2,3,4,5]
3
4 listIter = iter(list)
5
6 # prints '1'
7 print(next(listIter))
8
9 # prints '2'
10 print(next(listIter))
11
12 # prints '3'
13 print(next(listIter))
14
15 # prints '4'
16 print(next(listIter))
17
18 # prints '5'
19 print(next(listIter))

input
1
2
3
4
5
...Program finished with exit code 0
```

32. Python len() Function Example

```
main.py
1 strA = 'Python'
2 print(len(strA))

input
6
...Program finished with exit code 0
Press ENTER to exit console.
```

33. Python list() Example

```
main.py
1 print(list())
2
3 # string
4 String = 'abcde'
5 print(list(String))
6
7 # tuple
8 Tuple = (1,2,3,4,5)
9 print(list(Tuple))
10
11 # list
12 List = [1,2,3,4,5]
13 print(list(List))
```

```
[] input
['a', 'b', 'c', 'd', 'e']
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]

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

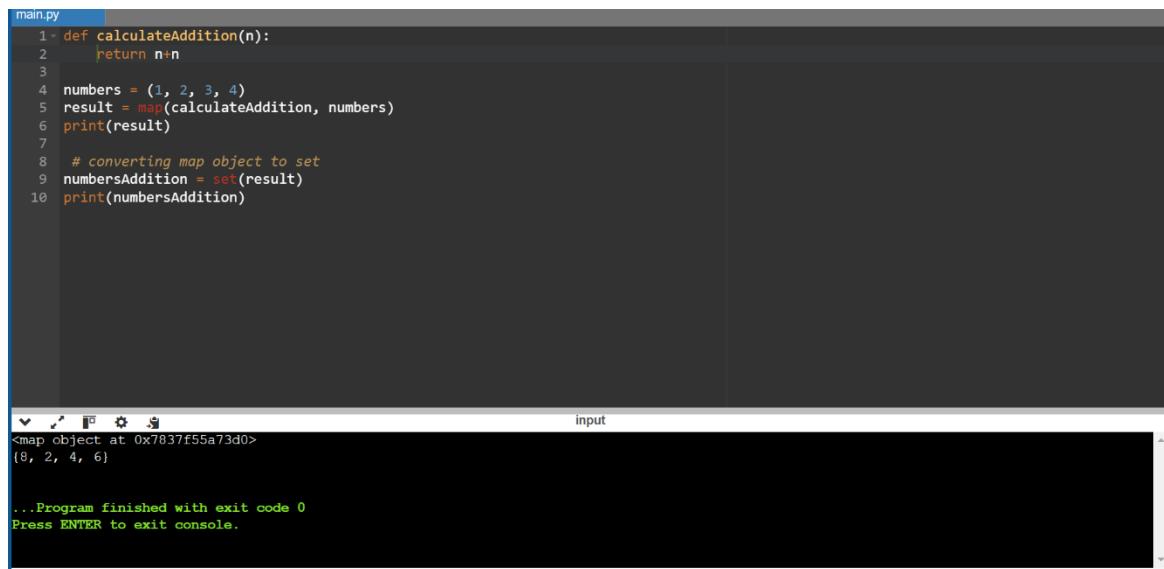
34. Python locals() Function Example

```
main.py
1 def localsAbsent():
2     return locals()
3
4 def localsPresent():
5     present = True
6     return locals()
7
8 print('localsNotPresent:', localsAbsent())
9 print('localsPresent:', localsPresent())
```

```
localsNotPresent: {}
localsPresent: {'present': True}

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

35. Python map() Function Example



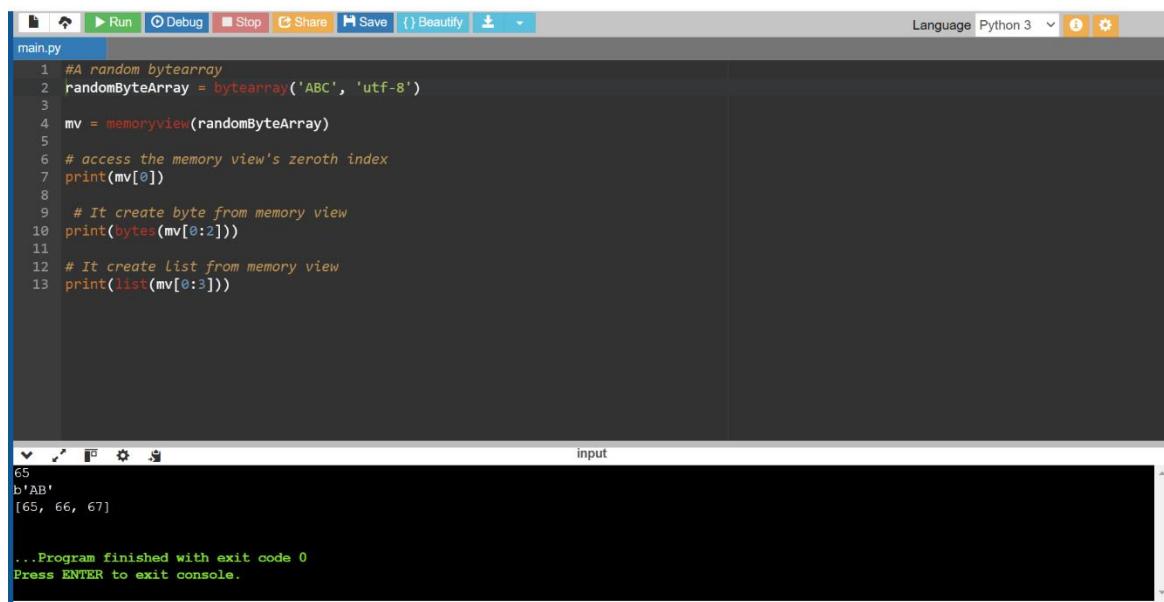
```
main.py
1 def calculateAddition(n):
2     return n*n
3
4 numbers = (1, 2, 3, 4)
5 result = map(calculateAddition, numbers)
6 print(result)
7
8 # converting map object to set
9 numbersAddition = set(result)
10 print(numbersAddition)
```

input

```
<map object at 0x7837f55a73d0>
{8, 2, 4, 6}

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

36. Python memoryview () Function Example



```
main.py
1 #A random bytearray
2 randomByteArray = bytearray('ABC', 'utf-8')
3
4 mv = memoryview(randomByteArray)
5
6 # access the memory view's zeroth index
7 print(mv[0])
8
9 # It create byte from memory view
10 print(bytes(mv[0:2]))
11
12 # It create list from memory view
13 print(list(mv[0:3]))
```

input

```
65
b'AB'
[65, 66, 67]

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

37. Python object() Example

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 python = object()
2
3 print(type(python))
4 print(dir(python))
```

Below the code editor is a terminal window titled "input" showing the program's output:

```
<class 'object'>
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']

...Program finished with exit code 0
www.onlinegdb.com/online_python_compiler#tab-stdin
```

38. Python chr() Function Example

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 # Calling function
2 result = chr(102) # It returns string representation of a char
3 result2 = chr(112)
4 # Displaying result
5 print(result)
6 print(result2)
7 # Verify, is it string type?
8 print("is it string type:", type(result) is str)
```

Below the code editor is a terminal window titled "input" showing the program's output:

```
f
p
is it string type: True

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

39. Python complex() Example

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 # Python complex() function example
2 # Calling function
3 a = complex(1) # Passing single parameter
4 b = complex(1,2) # Passing both parameters
5 # Displaying result
6 print(a)
7 print(b)
```

Below the code editor is a terminal window titled "input" showing the program's output:

```
(1+0j)
(1+2j)

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

40. Python delattr() Function Example

The screenshot shows a code editor window titled "main.py" containing the following Python code:

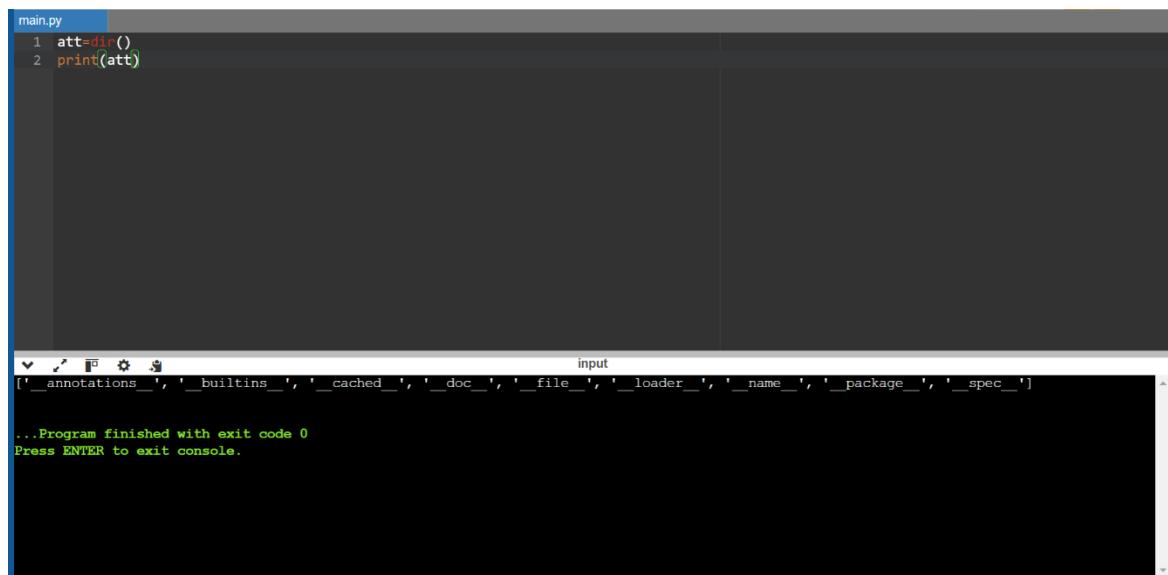
```
1 class Student:
2     id = 101
3     name = "Pranshu"
4     email = "pranshu@abc.com"
5     # Declaring function
6     def getinfo(self):
7         print(self.id, self.name, self.email)
8 s = Student()
9 s.getinfo()
10 delattr(Student, 'course') # Removing attribute which is not available
11 s.getinfo() # error: throws an error
```

Below the code editor is a terminal window titled "input" showing the program's output:

```
101 Pranshu pranshu@abc.com
Traceback (most recent call last):
  File "/home/main.py", line 10, in <module>
    delattr(Student,'course') # Removing attribute which is not available
                                ^
AttributeError: type object 'Student' has no attribute 'course'

...Program finished with exit code 1
Press ENTER to exit console.
```

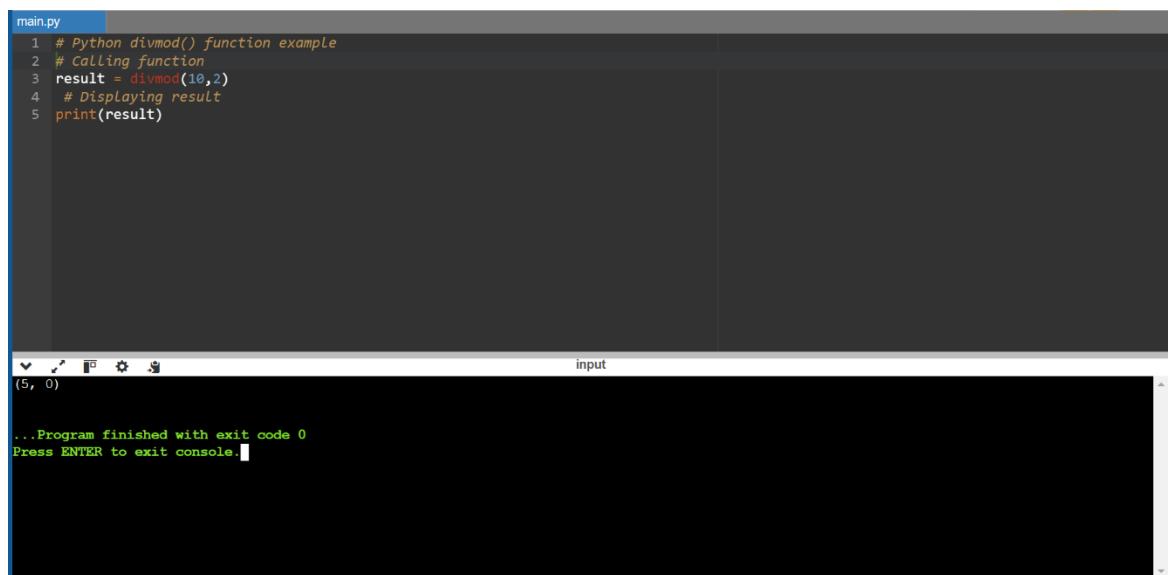
41. Python dir() Function Example



```
main.py
1 att=dir()
2 print(att)

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

42. Python divmod() Function Example



```
main.py
1 # Python divmod() function example
2 # Calling function
3 result = divmod(10,2)
4 # Displaying result
5 print(result)

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

43. Python enumerate() Function Example

The screenshot shows a terminal window with the title bar "main.py". The code in the editor is:

```
1 # Calling function
2 result = enumerate([1,2,3])
3 # Displaying result
4 print(result)
5 print(list(result))
```

The terminal output is:

```
input
<enumerate object at 0x7529737da390>
[(0, 1), (1, 2), (2, 3)]

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

44. Python dict() Example

The screenshot shows a terminal window with the title bar "main.py". The code in the editor is:

```
1 # Calling function
2 result = dict() # returns an empty dictionary
3 result2 = dict(a=1,b=2)
4 # Displaying result
5 print(result)
6 print(result2)
```

The terminal output is:

```
input
{}

{'a': 1, 'b': 2}

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

45. Python filter() Function Example

The screenshot shows a terminal window with a dark background. At the top, there's a blue header bar with the text "main.py" and some icons. Below the header, the terminal window has a title bar with "input" and a status bar with "[6]". The main area of the terminal contains the following text:

```
1 # Python filter() function example
2 def filteredata(x):
3     if x>5:
4         return x
5 # Calling function
6 result = filter(filteredata,(1,2,6))
7 # Displaying result
8 print(list(result))
```

At the bottom of the terminal window, there is a message from the program: "...Program finished with exit code 0 Press ENTER to exit console."

46. Python hash() Function Example

The screenshot shows a terminal window with a dark background. At the top, there's a blue header bar with the text "main.py" and some icons. Below the header, the terminal window has a title bar with "input" and a status bar with "[6]". The main area of the terminal contains the following text:

```
1 # Calling function
2 result = hash(21) # integer value
3 result2 = hash(22.2) # decimal value
4 # Displaying result
5 print(result)
6 print(result2)
```

At the bottom of the terminal window, there is a message from the program: "...Program finished with exit code 0 Press ENTER to exit console."

47. Python help() Function Example

The screenshot shows a terminal window with the title bar "input". The terminal displays the Python help utility output:

```
Welcome to Python 3.12's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.12/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list
the modules whose name or summary contain a given string such as "spam",
enter "modules spam".
```

48. Python min() Function Example

The screenshot shows a terminal window with the title bar "input". The terminal displays the output of a Python script named "main.py" which uses the min() function:

```
#Calling function
small = min(2225,325,2025) # returns smallest element
small2 = min(1000.25,2025.35,5625.36,10052.50)
# Displaying result
print(small)
print(small2)
```

```
325
1000.25

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

49. Python set() Function Example

```
main.py
1 # Calling function
2 result = set() # empty set
3 result2 = set('12')
4 result3 = set('javatpoint')
5 # Displaying result
6 print(result)
7 print(result2)
8 print(result3)
```

```
set()
{'1', '2'}
{'n', 'o', 'v', 'p', 'j', 'i', 'a', 't'}
```

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

50. Python hex() Function Example

```
main.py
1 # Calling function
2 result = hex(1)
3 # integer value
4 result2 = hex(342)
5 # Displaying result
6 print(result)
7 print(result2)
```

```
0x1
0x156
```

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

51. Python id() Function Example

```
main.py
1 # Calling function
2 val = id("javatpoint") # string object
3 val2 = id(1200) # integer object
4 val3 = id([25,336,95,236,92,3225]) # List object
5 # Displaying result
6 print(val)
7 print(val2)
8 print(val3)

125814745936112
125814746863664
125814747466560

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

52. Python setattr() Function Example

```
main.py
1 class Student:
2     id = 0
3     name = ""
4
5     def __init__(self, id, name):
6         self.id = id
7         self.name = name
8
9 student = Student(102,"Sohan")
10 print(student.id)
11 print(student.name)
12 #print(student.email) product error
13 setattr(student, 'email','sohan@abc.com') # adding new attribute
14 print(student.email)

102
Sohan
sohan@abc.com

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

53. Python slice() Function Example

```
main.py
1 #Calling function
2 result = slice(5) # returns slice object
3 result2 = slice(0,5,3) # returns slice object
4 # Displaying result
5 print(result)
6 print(result2)

input
slice(None, 5, None)
slice(0, 5, 3)

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

54. Python sorted() Function Example

```
main.py
1 str = "javatpoint" # declaring string
2 # Calling function
3 sorted1 = sorted(str) # sorting string
4 # Displaying result
5 print(sorted1)

input
['a', 'a', 'i', 'j', 'n', 'o', 'p', 't', 'e', 'v']

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

55. Python next() Function Example

The screenshot shows a Jupyter Notebook interface. The code cell contains:

```
1 number = iter([256, 32, 82]) # Creating iterator
2 # Calling function
3 item = next(number)
4 # Displaying result
5 print(item)
6 # second item
7 item = next(number)
8 print(item)
9 # third item
10 item = next(number)
11 print(item)
```

The output cell shows the results of the `next()` calls:

```
256
32
82
...Program finished with exit code 0
Press ENTER to exit console.
```

56. Python input() Function Example

The screenshot shows a Jupyter Notebook interface. The code cell contains:

```
1 # Calling function
2 val = input("Enter a value: ")
3 # Displaying result
4 print("You entered:",val)
```

The output cell shows the interaction with the user:

```
Enter a value: 311
You entered: 311
...Program finished with exit code 0
Press ENTER to exit console.
```

57. Python int() Function Example

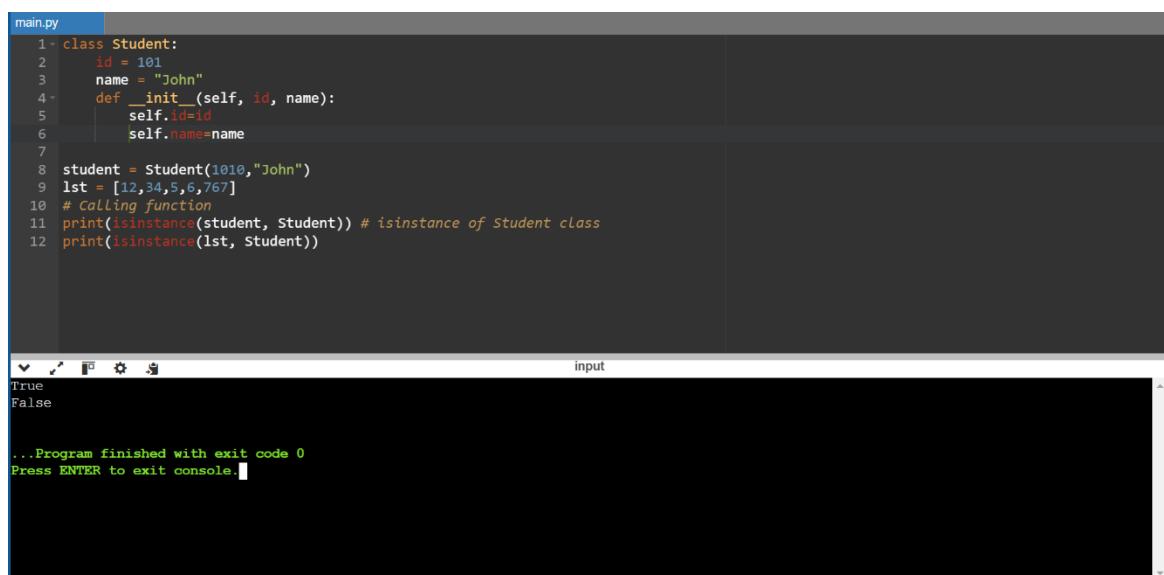


```
main.py
1 | # Calling function
2 val = int(10) # integer value
3 val2 = int(10.52) # float value
4 val3 = int('10') # string value
5 # Displaying result
6 print("integer values :",val, val2, val3)

integer values : 10 10 10

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

58. Python isinstance() function Example



```
main.py
1 - class Student:
2     id = 101
3     name = "John"
4     def __init__(self, id, name):
5         self.id=id
6         self.name=name
7
8 student = Student(1010,"John")
9 lst = [12,34,5,6,767]
10 # Calling function
11 print(isinstance(student, Student)) # isinstance of Student class
12 print(isinstance(lst, Student))

True
False

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

59. Python oct() function Example

```
main.py
1 # Calling function
2 val = oct(10)
3 # Displaying result
4 print("Octal value of 10:",val)

Octal value of 10: 0o12

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

60. Python ord() function Example

```
main.py
1 # Code point of an integer
2 print(ord('8'))
3
4 # Code point of an alphabet
5 print(ord('R'))
6
7 # Code point of a character
8 print(ord('&'))

56
82
38

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

61. Python pow() function Example

```
main.py
1 # positive x, positive y (x**y)
2 print(pow(4, 2))
3
4 # negative x, positive y
5 print(pow(-4, 2))
6
7 # positive x, negative y (x**-y)
8 print(pow(4, -2))
9
10 # negative x, negative y
11 print(pow(-4, -2))

16
16
0.0625
0.0625

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

62. Python print() function Example

```
main.py
1 print("Python is programming language.")
2
3 x = 7
4 # Two objects passed
5 print("x =", x)
6
7 y = x
8 # Three objects passed
9 print('x =', x, 'y')

Python is programming language.
x = 7
x = y

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

63. Python range() function Example

```
main.py
1 # empty range
2 print(list(range(0)))
3
4 # using the range(stop)
5 print(list(range(4)))
6
7 # using the range(start, stop)
8 print(list(range(1,7 )))

[1]
[0, 1, 2, 3]
[1, 2, 3, 4, 5, 6]

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

64. Python reversed() function Example

```
main.py
1 # for string
2 String = 'Java'
3 print(list(reversed(String)))
4
5 # for tuple
6 Tuple = ('J', 'a', 'v', 'a')
7 print(list(reversed(Tuple)))
8
9 # for range
10 Range = range(8, 12)
11 print(list(reversed(Range)))
12
13 # for list
14 List = [1, 2, 7, 5]
15 print(list(reversed(List)))

[1]
['a', 'v', 'a', 'J']
['a', 'v', 'a', 'J']
[11, 10, 9, 8]
[5, 7, 2, 1]

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

65. Python round() Function Example

```
main.py
1 # for integers
2 print(round(10))
3
4 # for floating point
5 print(round(10.8))
6
7 # even choice
8 print(round(6.6))

10
11
7

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

66. Python issubclass() Function Example

```
main.py
1 class Rectangle:
2     def __init__(rectangleType):
3         print('Rectangle is a ', rectangleType)
4
5 class Square(Rectangle):
6     def __init__(self):
7         Rectangle.__init__('square')
8
9 print(issubclass(Square, Rectangle))
10 print(issubclass(Square, list))
11 print(issubclass(Square, (list, Rectangle)))
12 print(issubclass(Rectangle, (list, Rectangle)))

True
False
True
True

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

67. Python str() Function Example



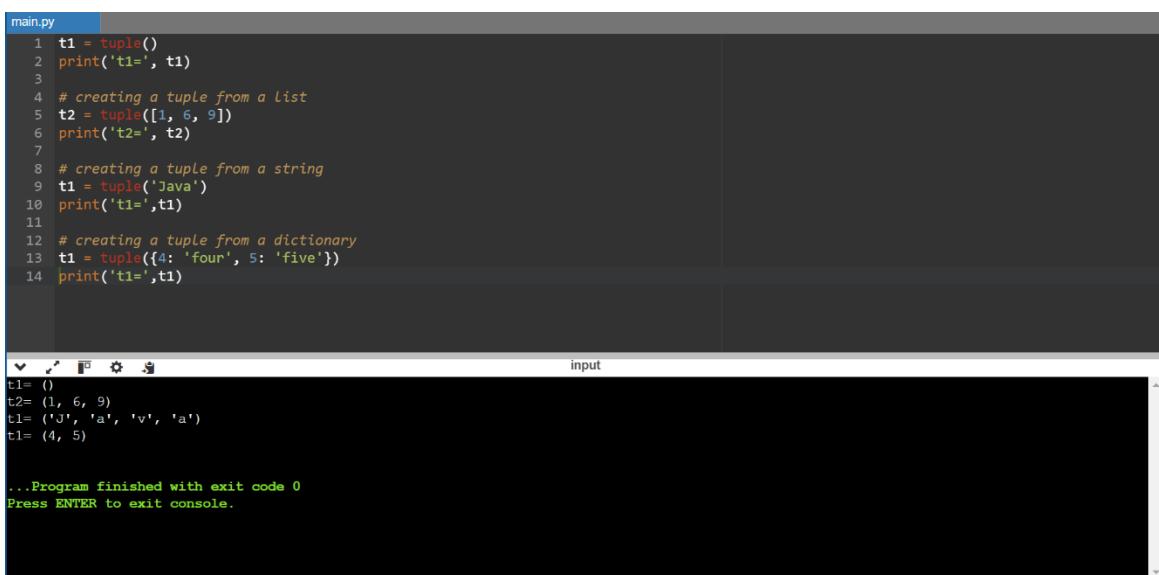
```
main.py
1 print(str('4'))
```

input

```
4

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

68. Python tuple() Function Example



```
main.py
1 t1 = tuple()
2 print('t1=', t1)
3
4 # creating a tuple from a list
5 t2 = tuple([1, 6, 9])
6 print('t2=', t2)
7
8 # creating a tuple from a string
9 t1 = tuple('Java')
10 print('t1=', t1)
11
12 # creating a tuple from a dictionary
13 t1 = tuple({4: 'four', 5: 'five'})
14 print('t1=', t1)
```

input

```
t1= ()
t2= (1, 6, 9)
t1= ('J', 'a', 'v', 'a')
t1= (4, 5)

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

69. Python type() Function Example

```
main.py
1 List = [4, 5]
2 print(type(List))
3
4 Dict = {4: 'four', 5: 'five'}
5 print(type(Dict))
6
7 class Python:
8     a = 0
9
10 InstanceOfPython = Python()
11 print(type(InstanceOfPython))
```

input

```
<class 'list'>
<class 'dict'>
<class '__main__.Python'>

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

70. Python vars() Function Example

```
main.py
1 class Python:
2     def __init__(self, x = 7, y = 9):
3         self.x = x
4         self.y = y
5
6 InstanceOfPython = Python()
7 print(vars(InstanceOfPython))
```

input

```
{'x': 7, 'y': 9}

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

71. Python zip() Function Example

The screenshot shows a Jupyter Notebook cell with the following code:

```
main.py
1 numList = [4,5, 6]
2 strList = ['four', 'five', 'six']
3
4 # No iterables are passed
5 result = zip()
6
7 # Converting iterator to list
8 resultList = list(result)
9 print(resultList)
10
11 # Two iterables are passed
12 result = zip(numList, strList)
13
14 # Converting iterator to set
15 resultSet = set(result)
16 print(resultSet)
```

The output of the code is:

```
[]
```

```
{(6, 'six'), (5, 'five'), (4, 'four')}
```

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

72. Code to demonstrate how we can use a lambda function for adding 4 numbers

The screenshot shows a Jupyter Notebook cell with the following code:

```
main.py
1 add = lambda num: num + 4
2 print( add(5) )
```

The output of the code is:

```
10
```

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

73. Example of a lambda function that adds 4 to the input number using the add function.

The screenshot shows a terminal window with a dark background. At the top, there is a blue header bar with the text "main.py". The main area contains the following Python code:

```
1 def add( num ):
2     return num + 4
3 print( add(6) )
```

Below the code, the terminal prompt "input" is visible. The output of the program is displayed in green text:

```
10
...Program finished with exit code 0
Press ENTER to exit console.
```

74. Example of a lambda function that multiply 2 numbers and return one result.

The screenshot shows a terminal window with a dark background. At the top, there is a blue header bar with the text "main.py". The main area contains the following Python code:

```
1 a = lambda x, y : (x * y)
2 print(a(4, 5))
```

Below the code, the terminal prompt "input" is visible. The output of the program is displayed in green text:

```
20
...Program finished with exit code 0
Press ENTER to exit console.
```

75. another example of a lambda function that adds 2 numbers and return one result.

The screenshot shows a terminal window titled "main.py". The code in the editor is:

```
1 a = lambda x, y, z : (x + y + z)
2 print(a(4, 5, 5))
```

The terminal output is:

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

76. Python code to show the reciprocal of the given number to highlight the difference between def() and lambda().

The screenshot shows a terminal window titled "main.py". The code in the editor is:

```
1 # Python code to show the reciprocal of the given number to highlight the difference between def() and Lambda().
2 def reciprocal( num ):
3     return 1 / num
4
5 lambda_reciprocal = lambda num: 1 / num
6
7 # using the function defined by def keyword
8 print( "Def keyword: ", reciprocal(5) )
9
10 # using the function defined by Lambda keyword
11 print( "Lambda keyword: ", lambda_reciprocal(5) )
```

The terminal output is:

```
Def keyword: 0.1666666666666666
Lambda keyword: 0.1666666666666666

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

77. example of lambda function with filter() in Python.

```
main.py
1 list_ = [35, 12, 69, 55, 75, 14, 73]
2 odd_list = list(filter( lambda num: (num % 2 != 0) , list_ ))
3 print('The list of odd number is:',odd_list)
```

```
input
The list of odd number is: [35, 69, 55, 75, 73]

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

78. Code to calculate the square of each number of a list using the map() function

```
main.py
1 #Code to calculate the square of each number of a List using the map() function
2 numbers_list = [2, 4, 5, 1, 3, 7, 8, 9, 10]
3 squared_list = list(map( lambda num: num ** 2 , numbers_list ))
4 print( 'Square of each number in the given list:' ,squared_list )
```

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

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

79. Code to calculate square of each number of lists using list comprehension

The screenshot shows a terminal window titled "input" with the following content:

```
main.py
1 #Code to calculate square of each number of lists using list comprehension
2 squares = [lambda num = num: num ** 2 for num in range(0, 11)]
3 for square in squares:
4     print('The square value of all numbers from 0 to 10:',square(), end = "\n")
```

The output of the program is displayed below the code:

```
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 square 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
```

At the bottom of the terminal window, it says "...Program finished with exit code 0" and "Press ENTER to exit console".

80. Code to use lambda function with if-else

The screenshot shows a terminal window titled "input" with the following content:

```
main.py
1 Minimum = lambda x, y : x if (x < y) else y
2 print('The greater number is:', Minimum( 35, 74 ))
```

The output of the program is displayed below the code:

```
The greater number is: 35
```

At the bottom of the terminal window, it says "...Program finished with exit code 0" and "Press ENTER to exit console".

81. Code to print the third largest number of the given list using the lambda function

The screenshot shows a terminal window with a dark background. At the top, there's a blue header bar with the text "main.py". The main area contains the following Python code:

```
1 # Code to print the third Largest number of the given List using the Lambda function
2
3 my_List = [ [3, 5, 8, 6], [23, 54, 12, 87], [1, 2, 4, 12, 5] ]
4 # sorting every sublist of the above list
5 sort_List = lambda num : ( sorted(n) for n in num )
6 # Getting the third Largest number of the sublist
7 third_Largest = lambda num, func : [ l[ len(l) - 2 ] for l in func(num) ]
8 result = third_Largest( my_List, sort_List )
9 print('The third largest number from every sub list is:', result )
```

Below the code, the terminal output is displayed:

```
The third largest number from every sub list is: [6, 54, 5]
...Program finished with exit code 0
Press ENTER to exit console.
```

MODULES

82. Python import Statement

The screenshot shows a terminal window with a dark background. At the top, there's a blue header bar with the text "main.py". The main area contains the following Python code:

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

Below the code, the terminal output is displayed:

```
The value of euler's number is 2.718281828459045
...Program finished with exit code 0
Press ENTER to exit console []
```

83. Importing and also Renaming

The screenshot shows a terminal window with a dark background. In the top left corner, there is a tab labeled "main.py". The main area of the terminal contains the following Python code:

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

Below the code, the terminal displays the output of the program:

```
The value of euler's number is 2.718281828459045
...Program finished with exit code 0
Press ENTER to exit console.
```

84. Python from...import Statement

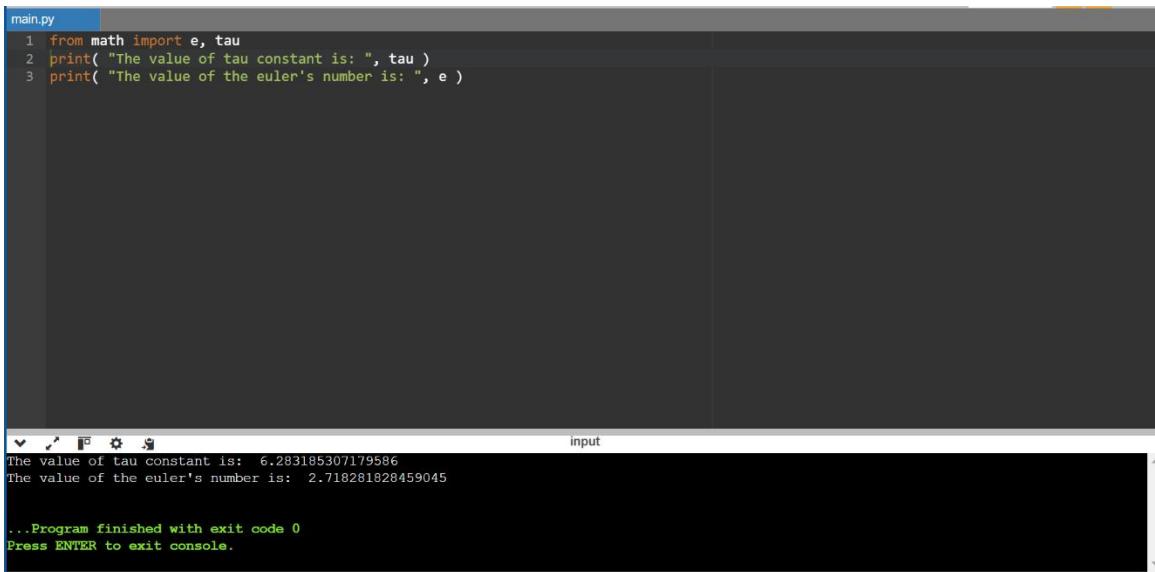
The screenshot shows a terminal window with a dark background. In the top left corner, there is a tab labeled "main.py". The main area of the terminal contains the following Python code:

```
1 from math import e
2 # here, the e value represents the euler's number
3 print( "The value of euler's number is", e )
```

Below the code, the terminal displays the output of the program:

```
The value of euler's number is 2.718281828459045
...Program finished with exit code 0
Press ENTER to exit console.
```

85. we are creating a simple Python program to show how to import multiple

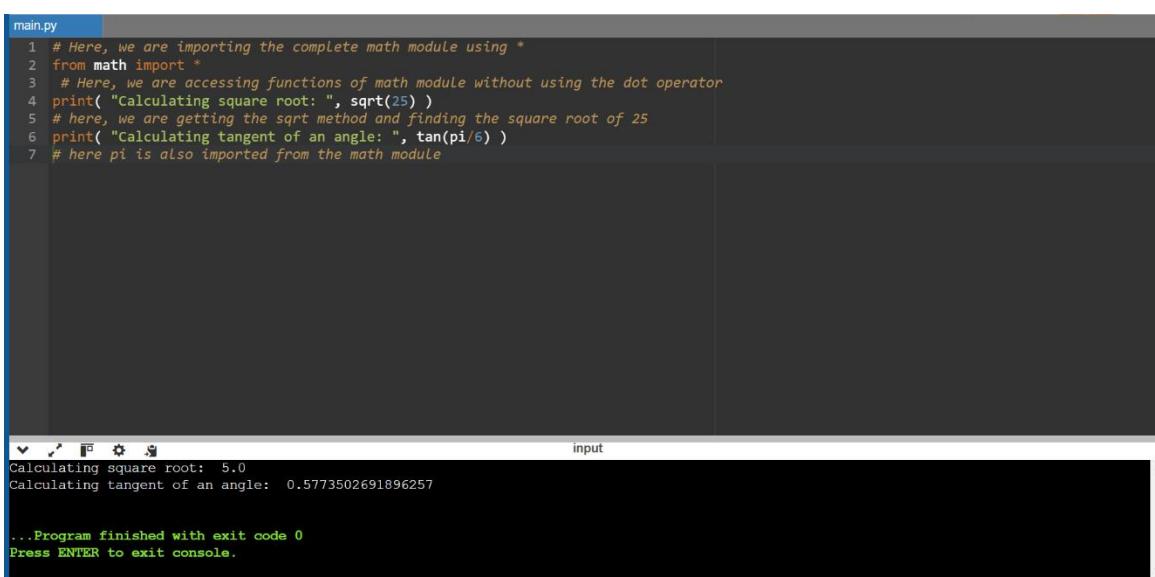


```
main.py
1 from math import e, tau
2 print( "The value of tau constant is: ", tau )
3 print( "The value of the euler's number is: ", e )
```

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

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

86. we are importing the complete math module using *



```
main.py
1 # Here, we are importing the complete math module using *
2 from math import *
3 # Here, we are accessing functions of math module without using the dot operator
4 print( "Calculating square root: ", sqrt(25) )
5 # here, we are getting the sqrt method and finding the square root of 25
6 print( "Calculating tangent of an angle: ", tan(pi/6) )
7 # here pi is also imported from the math module
```

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

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

87. Example to print the path.

```
main.py
1 # Here, we are importing the sys module
2 import sys
3 # Here, we are printing the path using sys.path
4 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/dist-packages', '/usr/lib/python3/dist-packages']

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

88. A simple Python program to print the directory of a module

```
main.py
1 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__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']]

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

89. Namespaces and scoping.

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 Number = 204
2 def AddNumber(): # here, we are defining a function with the name Add Number
3 # Here, we are accessing the global namespace
4     global Number
5     Number = Number + 200
6 print("The number is:", Number) |
7 # here, we are printing the number after performing the addition
8 AddNumber() # here, we are calling the function
9 print("The number is:", Number)
```

Below the code editor is a terminal window titled "input" showing the execution of the program:

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

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

EXCEPTION

90. Exceptions versus Syntax Errors.

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 #Python code after removing the syntax error
2 string = "Python Exceptions"
3
4 for s in string:
5     if (s != o:
6         print( s )
```

Below the code editor is a terminal window titled "input" showing the execution of the program, which results in a syntax error:

```
File "/home/main.py", line 5
    if (s != o:
        ^
SyntaxError: invalid syntax

...Program finished with exit code 1
Press ENTER to exit console.
```

91. Python code to catch an exception and handle it using try and except code blocks.

The screenshot shows a terminal window with the file 'main.py' open. The code attempts to print elements from index 0 to 4 of a list 'a'. It includes a try-except block to catch an IndexError if the loop goes beyond the array's length. The output shows the first three elements and then an error message 'Index out of range'.

```
main.py
1 # Python code to catch an exception and handle it using try and except code blocks
2
3 a = ["Python", "Exceptions", "try and except"]
4 try:
5     for i in range( 4 ):
6         print( "The index and element from the array is", i, a[i] )
7     #if an error occurs in the try block, then except block will be executed by the Python interpreter
8 except:
9     print ("Index out of range")
10
```

```
input
The index and element from the array is 0 Python
The index and element from the array is 1 Exceptions
The index and element from the array is 2 try and except
Index out of range

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

92. Python code to show how to raise an exception in Python

The screenshot shows a terminal window with the file 'main.py' open. The code defines a list 'num' with four elements and then checks if its length is greater than 3. If so, it raises an Exception with a message about the length being too long. The output shows a traceback of the exception being raised and caught, followed by the error message.

```
main.py
1 #Python code to show how to raise an exception in Python
2 num = [3, 4, 5, 7]
3 if len(num) > 3:
4     raise Exception( f"Length of the given list must be less than or equal to 3 but is {len(num)}" )

input
Traceback (most recent call last):
  File "/home/main.py", line 4, in <module>
    raise Exception( f"Length of the given list must be less than or equal to 3 but is {len(num)}" )
                                                               ^
Exception: Length of the given list must be less than or equal to 3 but is 4

...Program finished with exit code 1
Press ENTER to exit console.
```

93. The assert Statement

```
main.py
1 #Python program to show how to use assert keyword
2 # defining a function
3 def square_root( Number ):
4     assert ( Number < 0 ), "Give a positive integer"
5     return Number**(1/2)
6
7 #Calling function and passing the values
8 print( square_root( 36 ) )
9 print( square_root( -36 ) )

input
Traceback (most recent call last):
File "/home/main.py", line 8, in <module>
    print( square_root( 36 ) )
           ^^^^^^^^^^^^^^
File "/home/main.py", line 4, in square_root
    assert ( Number < 0 ), "Give a positive integer"
          ^^^^^^^^^^
AssertionError: Give a positive integer

...Program finished with exit code 1
Press ENTER to exit console.
```

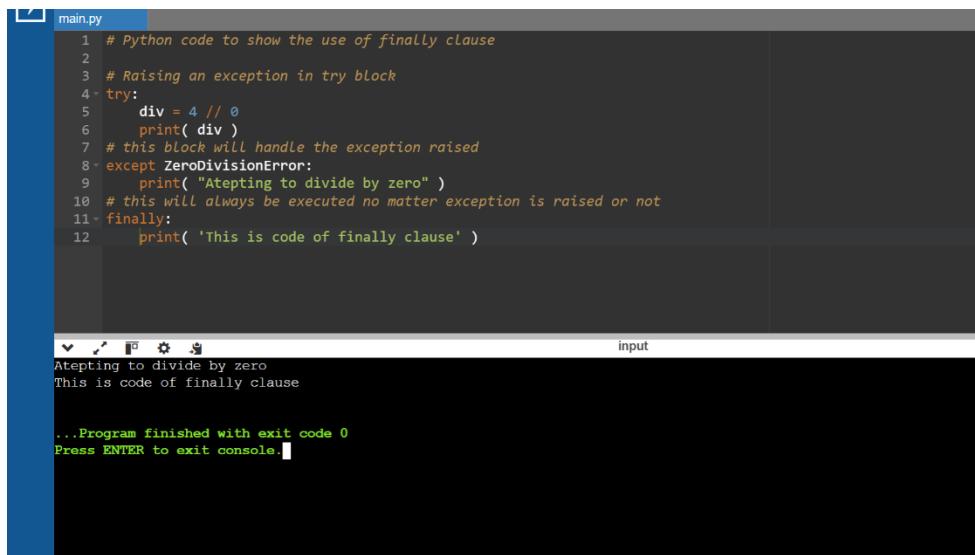
94. Try with Else Clause

```
main.py
1 # Python program to show how to use else clause with try and except clauses
2
3 # Defining a function which returns reciprocal of a number
4 def reciprocal( num1 ):
5     try:
6         reci = 1 / num1
7     except ZeroDivisionError:
8         print( "We cannot divide by zero" )
9     else:
10        print ( reci )
11 # Calling the function and passing values
12 reciprocal( 4 )
13 reciprocal( 0 )

input
0.25
We cannot divide by zero

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

95. Finally Keyword in Python

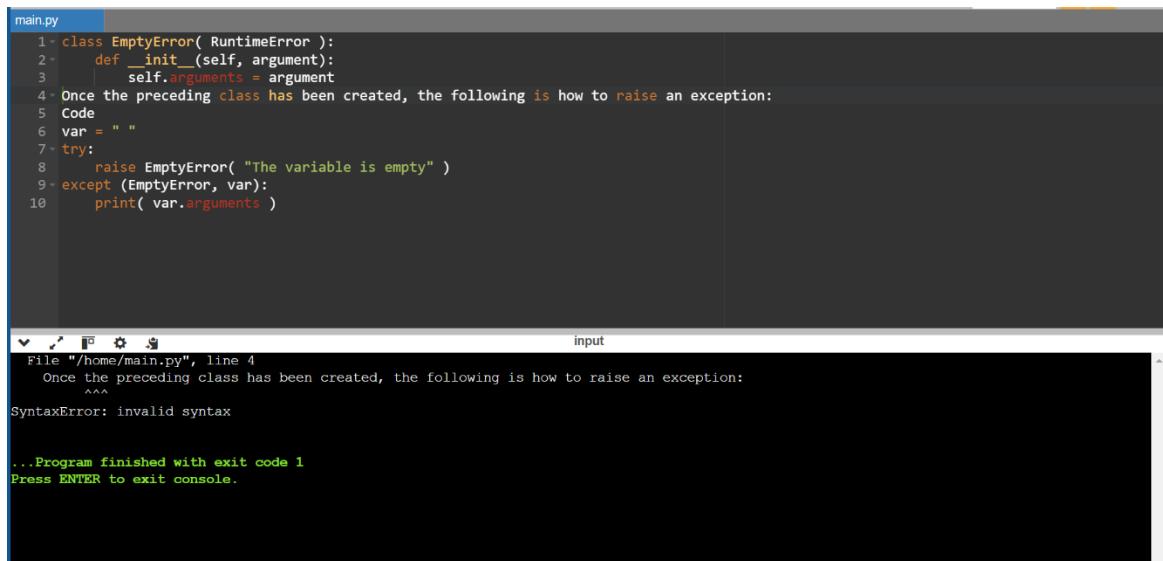


```
main.py
1 # Python code to show the use of finally clause
2
3 # Raising an exception in try block
4 try:
5     div = 4 // 0
6     print( div )
7 # this block will handle the exception raised
8 except ZeroDivisionError:
9     print( "Attempting to divide by zero" )
10 # this will always be executed no matter exception is raised or not
11 finally:
12     print( 'This is code of finally clause' )
```

Attempting to divide by zero
This is code of finally clause

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

96. User-Defined Exceptions



```
main.py
1 class EmptyError( RuntimeError ):
2     def __init__(self, argument):
3         self.arguments = argument
4 Once the preceding class has been created, the following is how to raise an exception:
5 Code
6 var = " "
7 try:
8     raise EmptyError( "The variable is empty" )
9 except (EmptyError, var):
10     print( var.arguments )
```

File "/home/main.py", line 4
Once the preceding class has been created, the following is how to raise an exception:
^ ^
SyntaxError: invalid syntax

...Program finished with exit code 1
Press ENTER to exit console.

ARRAY

97. Accessing array elements

```
main.py
1 import array as arr
2 a = arr.array('i', [2, 4, 5, 6])
3 print("First element is:", a[0])
4 print("Second element is:", a[1])
5 print("Third element is:", a[2])
6 print("Forth element is:", a[3])
7 print("last element is:", a[-1])
8 print("Second last element is:", a[-2])
9 print("Third last element is:", a[-3])
10 print("Forth last element is:", a[-4])
11 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

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

98. How to change or add elements?

```
main.py
1 import array as arr
2 numbers = arr.array('i', [1, 2, 3, 5, 7, 10])
3
4 # changing first element 1 by the value 0.
5 numbers[0] = 0
6 print(numbers)
7
8 # changing last element 10 by the value 8.
9 numbers[5] = 8
10 print(numbers)
11
12 # replace the value of 3rd to 5th element by 4, 6 and 8
13 numbers[2:5] = arr.array('i', [4, 6, 8])
14 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.

99. Delete Elements from an Array

The screenshot shows a terminal window with the following content:

```
main.py
1 import array as arr
2 number = arr.array('i', [1, 2, 3, 3, 4])
3 del number[2] # removing third element
4 print(number)
```

Output:

```
array('i', [1, 2, 3, 4])
...Program finished with exit code 0
Press ENTER to exit console.
```

100. Array Concatenation

The screenshot shows a terminal window with the following content:

```
main.py
1 import array as arr
2 a=arr.array('d',[1.1 , 2.1 ,3.1,2.6,7.8])
3 b=arr.array('d',[3.7,8.6])
4 c=arr.array('d')
5 c=a+b
6 print("Array c = ",c)
```

Output:

```
Array c =  array('d', [1.1, 2.1, 3.1, 2.6, 7.8, 3.7, 8.6])
...Program finished with exit code 0
Press ENTER to exit console.
```

101. Array Concatenation

The screenshot shows a terminal window with two panes. The left pane contains the Python script `main.py`:

```
main.py
1 import array as arr
2 x = arr.array('i', [4, 7, 19, 22]) # Initialize the array elements
3 print("First element:", x[0])
4 print("Second element:", x[1])
5 print("Second last element:", x[-1])
```

The right pane shows the terminal output:

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

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

DECORATORS

102. Sample example

The screenshot shows a terminal window with two panes. The left pane contains the Python script `main.py`:

```
main.py
1 def func1(msg):
2     print(msg)
3 func1("Hii, welcome to function ")
4 func2 = func1
5 func2("Hii, welcome to function ")
```

The right pane shows the terminal output:

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

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

103. Inner Function

The screenshot shows a terminal window with the title "input". The code in the editor is:

```
main.py
1 def func():
2     print("We are in first function")
3 def func1():
4     print("This is first child function")
5
6 def func2():
7     print("This is second child function")
8 func1()
9 func2()
10 func()
```

The terminal output is:

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

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

104. Inner Function

The screenshot shows a terminal window with the title "input". The code in the editor is:

```
main.py
1 def add(x):
2     return x+1
3 def sub(x):
4     return x-1
5 def operator(func, x):
6     temp = func(x)
7     return temp
8 print(operator(sub,10))
9 print(operator(add,20))
```

The terminal output is:

```
9
21

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

105. Inner Function

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 def hello():
2     def hi():
3         print("Hello")
4     return hi
5 new = hello()
6 new()
```

Below the code editor is a terminal window titled "input" showing the output of the program:

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

106. Decorating functions with parameters

The screenshot shows a code editor window titled "main.py" containing the following Python code:

```
1 def divide(x,y):
2     print(x/y)
3 def outer_div(func):
4
5     def inner(x,y):
6         if(x < y):
7             x,y = y,x
8         return func(x,y)
9     return inner
10 divide1 = outer_div(divide)
11 divide1(2,4)
```

Below the code editor is a terminal window titled "input" showing the output of the program:

```
2.0
...Program finished with exit code 0
Press ENTER to exit console
```

107. Syntactic Decorator

The screenshot shows a code editor window with a dark theme. The file is named 'main.py'. The code defines a function 'outer_div' which returns an inner function 'inner'. Inside 'inner', if x < y, it swaps them. Then it prints the result of calling the original function 'func' with the swapped values. The code then calls 'divide(4, 2)'. The output window below shows the program finished with exit code 0 and prompts the user to press Enter to exit.

```
main.py
1 def outer_div(func):
2     def inner(x, y):
3         if x < y:
4             x, y = y, x
5             return func(x, y)
6     return inner
7
8 @outer_div
9 def divide(x, y):
10    print(x / y)
11
12 divide(4, 2)
13
14
```

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

108. Reusing Decorator

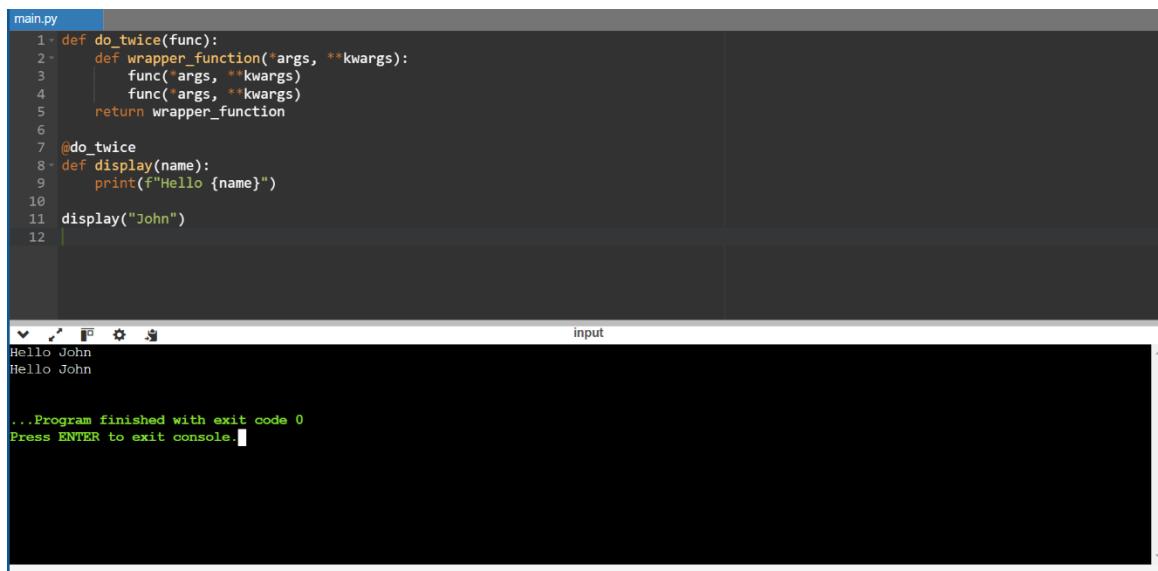
The screenshot shows a code editor window with a dark theme. The file is named 'main.py'. It imports 'functools' and defines a decorator 'do_twice' that wraps a function 'func'. The wrapper calls 'func' twice. Then it defines a function 'say_hello' and applies the 'do_twice' decorator to it. When 'say_hello' is called, it prints 'Hello There' twice. The output window shows the two 'Hello There' prints and the exit message.

```
main.py
1 import functools
2
3 def do_twice(func):
4     @functools.wraps(func)
5     def wrapper_do_twice():
6         func()
7         func()
8     return wrapper_do_twice
9
10 @do_twice
11 def say_hello():
12     print("Hello There")
13
14 say_hello()
15
16
```

Hello There
Hello There

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

109. Python Decorator with Argument

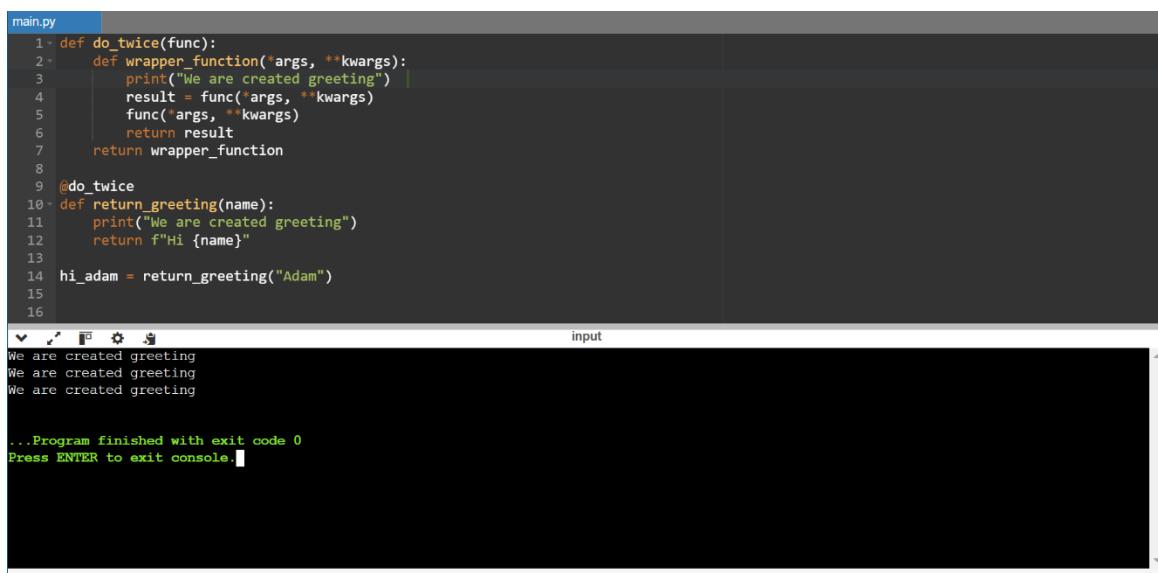


```
main.py
1 def do_twice(func):
2     def wrapper_function(*args, **kwargs):
3         func(*args, **kwargs)
4         func(*args, **kwargs)
5     return wrapper_function
6
7 @do_twice
8 def display(name):
9     print(f"Hello {name}")
10
11 display("John")
12 
```

```
Hello John
Hello John

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

110. Returning Values from Decorated Functions



```
main.py
1 def do_twice(func):
2     def wrapper_function(*args, **kwargs):
3         print("We are created greeting") |
4         result = func(*args, **kwargs)
5         func(*args, **kwargs)
6     return result
7
8 @do_twice
9 def return_greeting(name):
10    print("We are created greeting")
11    return f"Hi {name}"
12
13
14 hi_adam = return_greeting("Adam")
15
16 
```

```
We are created greeting
We are created greeting
We are created greeting

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

111. Class Decorators - @property decorator

The screenshot shows a terminal window with the following content:

```
main.py
1+ class Student: # here, we are creating a class with the name Student
2+     def __init__(self, name, grade):
3+         self.name = name
4+         self.grade = grade
5+     @property
6+     def display(self):
7+         return self.name + " got grade " + self.grade
8+
9+ stu = Student("John", "B")
10 print("Name of the student: ", stu.name)
11 print("Grade of the student: ", stu.grade)
12 print(stu.display)
13
```

input

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

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

112. Class Decorators - @staticmethod decorator

The screenshot shows a terminal window with the following content:

```
main.py
1+ class Person: # here, we are creating a class with the name Person
2+     @staticmethod
3+     def hello(): # here, we are defining a function hello
4+         print("Hello Peter")
5+ per = Person()
6+ per.hello()
7+ Person.hello()
8+
```

input

```
Hello Peter
Hello Peter

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

113. Decorator with Arguments

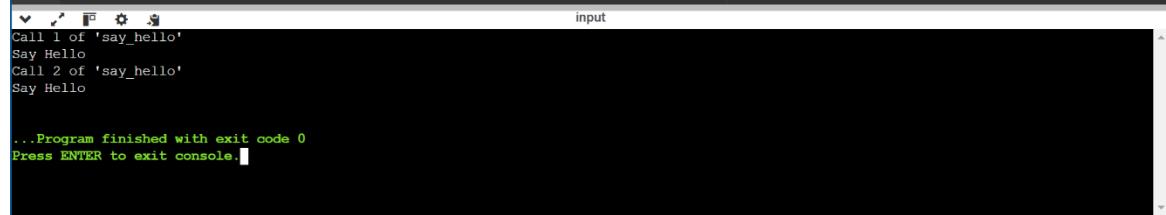


```
main.py
1 import functools
2 def repeat(num):
3     def decorator_repeat(func):
4         @functools.wraps(func)
5         def wrapper(*args, **kwargs):
6             for _ in range(num):
7                 value = func(*args, **kwargs)
8             return value
9         return wrapper
10    return decorator_repeat
11
12 @repeat(num=5)
13 def function1(name):
14     print(f"{{name}}")
15 function1("JavatPoint")
```

JavatPoint
JavatPoint
JavatPoint
JavatPoint
JavatPoint

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

114. Stateful Decorators



```
main.py
1 import functools
2
3 def count_function(func):
4     @functools.wraps(func)
5     def wrapper_count_calls(*args, **kwargs):
6         wrapper_count_calls.num_calls += 1
7         print(f"Call {wrapper_count_calls.num_calls} of {func.__name__!r}")
8         return func(*args, **kwargs)
9     wrapper_count_calls.num_calls = 0
10    return wrapper_count_calls
11
12 @count_function
13 def say_hello():
14     print("Say Hello")
15
16 say_hello()
17 say_hello()
18
```

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

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

115. Classes as Decorators

The screenshot shows a terminal window with two panes. The left pane contains the Python code in a file named `main.py`. The right pane shows the output of running the program.

```
main.py
1 import functools
2 class Count_Calls:
3     def __init__(self, func):
4         functools.update_wrapper(self, func)
5         self.func = func
6         self.num_calls = 0
7
8     def __call__(self, *args, **kwargs):
9         self.num_calls += 1
10        print(f"Call {self.num_calls} of {self.func.__name__!r}")
11        return self.func(*args, **kwargs)
12
13 @Count_Calls
14 def say_hello():
15     print("Say Hello")
16
17 say_hello()
18 say_hello()
19 say_hello()
20 
```

input

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

...Program finished with exit code 0
```