

# Python Functions

The screenshot shows the PyCharm IDE interface. The top bar has icons for PC, file, PythonProject, and Version control. The left sidebar shows a folder icon and the file demo.py. The main code editor window contains the following Python code:

```
# Example Python Code for User-Defined Function
def square(num):
    return num ** 2
object_ = square(6)
print("The square of the given number is:", object_)
```

The bottom right panel is the terminal window titled "Run" with the tab "demo". It shows the command run and the output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
The square of the given number is: 36
Process finished with exit code 0
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar has icons for PC, file, PythonProject, and Version control. The right side of the top bar shows "Current File" with a dropdown arrow. The left sidebar shows a folder icon and the file demo.py. The main code editor window contains the following Python code:

```
def a_function(string):
    return len(string)

print("Length of the string 'Functions' is:", a_function("Functions"))
print("Length of the string 'Python' is:", a_function("Python"))
```

The bottom right panel is the terminal window titled "Run" with the tab "demo". It shows the command run and the output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Length of the string 'Functions' is: 9
Length of the string 'Python' is: 6
```

Q.

```
def square(item_list): 1 usage
    squares = []
    for l in item_list:
        squares.append(l ** 2)
    return squares

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

Run demo

C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProject

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

Q.

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

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

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

Output

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

== Code Execution Successful ==

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The main area shows a file named "demo.py" with the following content:

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

print("Without using keyword")
function( n1: 50, n2: 30)

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

Below the code editor is a "Run" tool window with a single entry: "demo". The run output shows two sets of printed statements:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Without using keyword
number 1 is: 50
number 2 is: 30
With using keyword
number 1 is: 30
number 2 is: 50
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The main area shows a file named "demo.py" with the following content:

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

print("Passing out of order arguments")
function( n1: 30, n2: 20)

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

Below the code editor is a "Run" tool window with a single entry: "demo". The run output shows three distinct messages:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Passing out of order arguments
number 1 is: 30
number 2 is: 20
Passing only one argument
Function needs two positional arguments
```

Q.

The screenshot shows a PyCharm interface with a dark theme. The top bar displays 'PP PythonProject' and 'Version control'. The main area shows a file named 'demo.py' with the following code:

```
def function(*args_list): 1 usage
    ans = []
    for l in args_list:
        ans.append(l.upper())
    return ans

object_ = function( *args_list: 'Python', 'Functions', 'tutorial')
print(object_)

def function(**kwargs_list): 1 usage
    ans = []
    for key, value in kwargs_list.items():
        ans.append([key, value])
    return ans

object_ = function(First="Python", Second="Functions", Third="Tutorial")
print(object_)
```

The 'Run' tab is selected, showing the output:

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

Q.

The screenshot shows a PyCharm interface with a dark theme. The top bar displays 'PP PythonProject' and 'Version control'. The main area shows a file named 'demo.py' with the following code:

```
def square(num): 1 usage
    return num ** 2

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

def square(num): 1 usage
    num ** 2

print("Without return statement")
pi=int(square(52))
```

The 'Run' tab is selected, showing the output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProject
With return statement
2704
Without return statement
None
```

**Q.**

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the file "demo.py". The code editor contains the following Python code:

```
lambda_ = lambda argument1, argument2: argument1 + argument2
print("Value of the function is:", lambda_(argument1=20, argument2=30))
print("Value of the function is:", lambda_(argument1=40, argument2=50))
```

The run tool window at the bottom shows the output of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Value of the function is: 50
Value of the function is: 90
```

**Q.**

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the file "demo.py". The code editor contains the following Python code:

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

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

The run tool window at the bottom shows the output of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Value of num inside the function: 50
Value of num outside the function: 10
Process finished with exit code 0
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python script:

```
def word():
    string = 'Python functions tutorial'
    x = 5
    def number():
        print(string)
        print(x)
    number()
word()
```

The run tool window at the bottom shows the output of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
Python functions tutorial
5
Process finished with exit code 0
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains the following Python script:

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

# floating number
floating = -40.83
print('Absolute value of -40.83 is:', abs(floating))
```

The run tool window at the bottom shows the output of running the script:

```
Absolute value of -20
Absolute value of -40.83 is: 40.83
Process finished with exit code 0
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains a script named "demo.py" with the following content:

```
1 k = [1, 3, 4, 6]
2 print(all(k))
3
...
4 k = [0, False]
5 print(all(k))
6
7 k = [1, 3, 7, 0]
8 print(all(k))
9
10 k = [0, False, 5]
11 print(all(k))
12
13 k = []
14 print(all(k))
```

The "Run" tab is selected, showing the output of the script. The output window displays the results of the `all()` function for different lists:

Input List	Output
[1, 3, 4, 6]	True
[0, False]	False
[1, 3, 7, 0]	False
[0, False, 5]	False
[]	True

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the current file "demo.py". The code editor contains a script named "demo.py" with the following content:

```
1 x = 10
2 y = bin(x)
3 print(y)
```

The "Run" tab is selected, showing the output of the script. The output window displays the result of the `bin()` function:

Output
0b1010

Below the output, a message indicates the process finished with exit code 0.

Q.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for file types like Python, SQL, and JSON. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is as follows:

```
1 test1 = []
2 print(test1, 'is', bool(test1))
3
4 test1 = [0]
5 print(test1, 'is', bool(test1))
6
7 test1 = 0.0
8 print(test1, 'is', bool(test1))
9
10 test1 = None
11 print(test1, 'is', bool(test1))
12
13 test1 = True
14 print(test1, 'is', bool(test1))
15
16 test1 = 'Easy string'
17 print(test1, 'is', bool(test1))
18
```

The 'Output' tab shows the results of running the code:

```
[] is False
[0] is True
0.0 is False
None is False
True is True
Easy string is True
--- Code Execution Successful ---
```

At the bottom right of the window, there is a message: "Activate Windows" and "Go to Settings to activate Windows."

Q.

The screenshot shows the PyCharm IDE interface. The top bar shows 'PythonProject' and 'Version control'. The left sidebar shows a project tree with 'demo.py' selected. The main area has tabs for 'Run' and 'Output'. The code in 'demo.py' is:

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

The 'Output' tab shows the results of running the code:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\da
[] is False
[0] is True
0.0 is False
None is False
True is True
Easy string is True
Process finished with exit code 0
```

Q.

The screenshot shows the PyCharm IDE interface. At the top, there's a dark header bar with icons for file operations, a project named "PythonProject", and "Version control". Below the header is a file browser showing a folder icon and a file named "demo.py". The main editor area contains the following code:

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

Below the editor is a "Run" tool bar with a "Run" button and a dropdown menu set to "demo". The bottom half of the screen is the "Output" window, which displays the command prompt and its output:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\pyt
False

Process finished with exit code 0
```

Q.

The screenshot shows a Jupyter Notebook cell interface. On the left, there's a sidebar with various icons. The main area has a tab labeled "main.py". The code in the cell is:

```
1 code_str = 'x=5\ny=10\nprint("sum =", x+y)'
2 code = compile(code_str, 'sum.py', 'exec')
3
4 print(type(code))
5 exec(code)
6
7
8 print(x)
9
```

To the right of the code is a "Run" button and an "Output" section. The output shows the results of the executed code:

```
<class 'code'>
sum = 15
5
==== Code Execution Successful ===
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The left sidebar shows a file tree with "demo.py" selected. The main editor window contains the following code:

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

The run output window at the bottom shows the results of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\Pycharm...
True
12
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays "PP PythonProject" and "Version control". The right side of the top bar includes "Current File", a green arrow, and other icons. The left sidebar shows a file tree with "demo.py" selected. The main editor window contains the following code:

```
1 s = sum([1, 2, 4])
2 print(s)
3
4 s = sum([1, 2, 4], 10)
5 print(s)
6
```

The run output window at the bottom shows the results of running the script:

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\Pycharm...
7
17
```

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the file "demo.py". The code editor contains the following Python code:

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

...
l = [0, False]
print(any(l))

l = [0, False, 5]
print(any(l))

l = []
print(any(l))
```

The run tool window at the bottom shows the output of the script:

Run	demo	x
Run	demo	x
Up	C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe	C:\Users\Administrator\PycharmProj
Down	True	
Right	False	
Left	True	
Equal	False	

Q.

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PythonProject" and the file "demo.py". The code editor contains the following Python code:

```
normalText = 'Python is interesting'
print(ascii(normalText))

...
otherText = 'Pythön is interesting'
print(ascii(otherText))

print('Pyth\xf6n is interesting')
```

The run tool window at the bottom shows the output of the script:

Run	demo	x
Run	demo	x
Up	C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe	C:\Users\Administrator\PycharmProj
Down	'Python is interesting'	
Right	'Pyth\xf6n is interesting'	
Left	'Pythön is interesting'	
Equal	Pythön is interesting	

**Q.**

The screenshot shows a Jupyter Notebook interface. On the left is a sidebar with icons for file operations. The main area has a tab for "main.py". The code cell contains:

```
1 string = "Python is a programming language."
2 arr = bytearray(string, 'utf-8')
3 print(arr)
```

The "Run" button is highlighted in blue. To the right, the "Output" section shows the result of the code execution:

```
bytearray(b'Python is a programming language.')
==== Code Execution Successful ====
```

**Q.**

The screenshot shows a Jupyter Notebook interface. On the left is a sidebar with icons for file operations. The main area has a tab for "main.py". The code cell contains:

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

The "Run" button is highlighted in blue. To the right, the "Output" section shows the result of the code execution:

```
9
==== Code Execution Successful ====
```

**Q.**

The screenshot shows a Jupyter Notebook interface. On the left is a sidebar with icons for file operations. The main area has a tab for "main.py". The code cell contains:

```
1 print(float(9))
2 print(float(8.19))
3 print(float("-24.27"))
4 print(float(" -17.19\n"))
5 print(float("xyz"))
6
```

The "Run" button is highlighted in blue. To the right, the "Output" section shows the result of the code execution, which includes an error message:

```
ERROR!
9.0
8.19
-24.27
-17.19
Traceback (most recent call last):
  File "<main.py>", line 5, in <module>
ValueError: could not convert string to float: 'xyz'
==== Code Exited With Errors ====
```

**Q.**

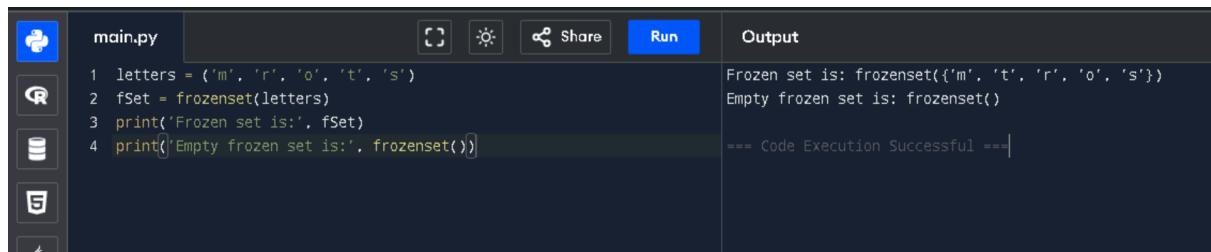
The screenshot shows a Jupyter Notebook interface. On the left is a sidebar with icons for file operations. The main area has a tab for "main.py". The code cell contains:

```
1 print(format(123, "d"))
2 print(format(123.4567898, "f"))
3 print(format(12, "b"))
4
```

The "Run" button is highlighted in blue. To the right, the "Output" section shows the result of the code execution:

```
123
123.456790
1100
==== Code Execution Successful ====
```

Q.



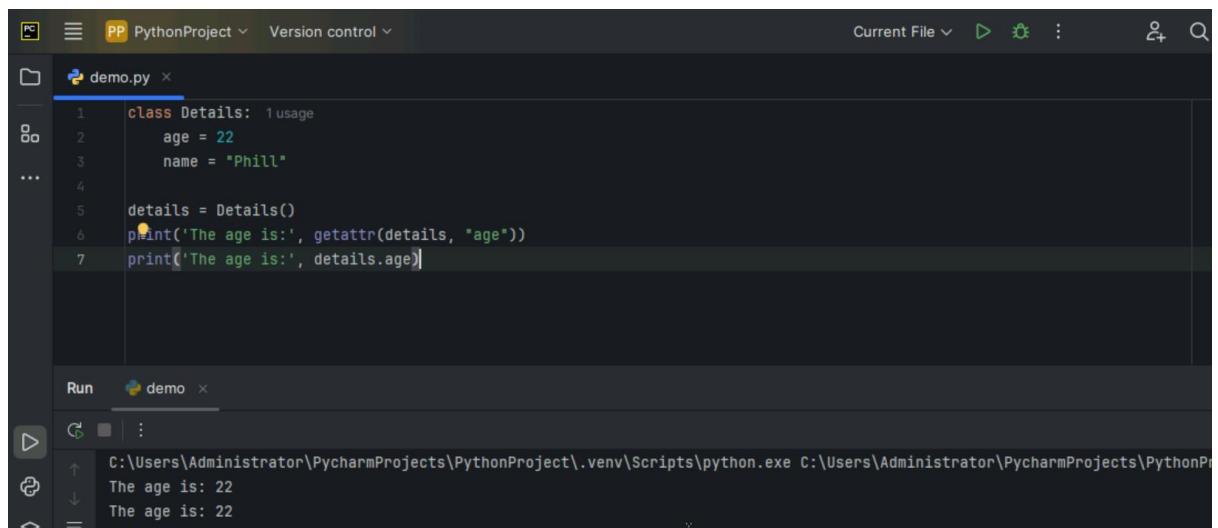
main.py

```
1 letters = {'m', 'r', 'o', 't', 's'}
2 fSet = frozenset(letters)
3 print('Frozen set is:', fSet)
4 print('Empty frozen set is:', frozenset{})
```

Output

```
Frozen set is: frozenset({'m', 't', 'r', 'o', 's'})
Empty frozen set is: frozenset()
== Code Execution Successful ==
```

Q.



demo.py

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

Run demo

```
C:\Users\Administrator\PycharmProjects\PythonProject\.venv\Scripts\python.exe C:\Users\Administrator\PycharmProjects\PythonProject\demo.py
The age is: 22
The age is: 22
```

Q.



main.py

```
1 age = 22
2
3 globals()['age'] = 22
4 print('The age is:', age)
5
```

Output

```
The age is: 22
== Code Execution Successful ==
```

**Q.**

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
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))
12
```

On the right, there is an "Output" section with the following results:

```
True
False
True
False
==== Code Execution Successful ===
```

**Q.**

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
1 list = [1, 2, 3, 4, 5]
2
3 listIter = iter(list)
4
5 print(next(listIter))
6 print(next(listIter))
7 print(next(listIter))
8 print(next(listIter))
9 print(next(listIter))
10
```

On the right, there is an "Output" section with the following results:

```
1
2
3
4
5
==== Code Execution Successful ===
```

**Q.**

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there is a sidebar with various icons. The main area has a file named "main.py" containing the following code:

```
1 strA = 'Python'
2 print(len(strA))
```

On the right, there is an "Output" section with the following results:

```
6
==== Code Execution Successful ===
```

**Q.**

The screenshot shows a Python code editor interface with a dark theme. On the left, there is a vertical toolbar with various icons. The main area is titled "main.py" and contains the following code:

```
1 print(list())
2
3 String = 'abcde'
4 print(list(String))
5
6 Tuple = (1, 2, 3, 4, 5)
7 print(list(Tuple))
8
9 List = [1, 2, 3, 4, 5]
10 print(list(List))
11
```

At the top right, there are buttons for "Run" and "Output". The "Output" panel displays the results of the code execution:

```
[] ['a', 'b', 'c', 'd', 'e'] [1, 2, 3, 4, 5] [1, 2, 3, 4, 5]
--- Code Execution Successful ---
```

**Q.**

The screenshot shows a Python code editor interface with a dark theme. On the left, there is a vertical toolbar with various icons. The main area is titled "main.py" and contains the following code:

```
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())
10
```

At the top right, there are buttons for "Run" and "Output". The "Output" panel displays the results of the code execution:

```
localsNotPresent: {}
localsPresent: {'present': True}
--- Code Execution Successful ---
```

**Q.**

The screenshot shows a Python code editor interface with a dark theme. On the left, there is a vertical toolbar with various icons. The main area is titled "main.py" and contains the following code:

```
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 numbersAddition = set(result)
9 print(numbersAddition)
10
```

At the top right, there are buttons for "Run" and "Output". The "Output" panel displays the results of the code execution:

```
<map object at 0x79348d4dd540>
{8, 2, 4, 6}
--- Code Execution Successful ---
```

Q.

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons for file operations like Open, Save, and Share. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 randomByteArray = bytearray('ABC', 'utf-8')
2
3 mv = memoryview(randomByteArray)
4
5 print(mv[0])
6 print(bytes(mv[0:2]))
7 print(list(mv[0:3]))
8
```

The 'Output' tab shows the results of running the code:

```
65
b'AB'
[65, 66, 67]
== Code Execution Successful ==
```

Q.

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons for file operations like Open, Save, and Share. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 python = object()
2
3 print(type(python))
4 print(dir(python))
5
```

The 'Output' tab shows the results of running the code:

```
<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__']

== Code Execution Successful ==
```

Q.

The screenshot shows a Jupyter Notebook interface with a dark theme. On the left is a sidebar with various icons for file operations like Open, Save, and Share. The main area has tabs for 'main.py' and 'Output'. The code in 'main.py' is:

```
1 result = chr(102)
2 result2 = chr(112)
3
4 print(result)
5 print(result2)
6 print("is it string type:", type(result) is str)
7
```

The 'Output' tab shows the results of running the code:

```
f
p
is it string type: True
== Code Execution Successful ==
```

Q.

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Run. The main area is titled "main.py" and contains the following code:

```
1 a = complex()
2 b = complex(1, 2)
3
4 print(a)
5 print(b)
6
```

At the top right, there are buttons for "Run" and "Output". The "Run" button is blue. The "Output" section shows the results of the code execution:

```
(1+0j)
(1+2j)
==== Code Execution Successful ===
```

Q.

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Run. The main area is titled "main.py" and contains the following code:

```
1 class Student:
2     id = 101
3     name = "Pranshu"
4     email = "pranshu@abc.com"
5
6     def getinfo(self):
7         print(self.id, self.name, self.email)
8
9 s = Student()
10 s.getinfo()
11 delattr(Student, 'course')
12 s.getinfo()
13
```

At the top right, there are buttons for "Run" and "Output". The "Run" button is blue. The "Output" section shows the results of the code execution:

```
101 Pranshu pranshu@abc.com
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 11, in <module>
AttributeError: type object 'Student' has no attribute 'course'

==== Code Exited With Errors ===
```

Q.

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Run. The main area is titled "main.py" and contains the following code:

```
1 att = dir()
2 print(att)
3
```

At the top right, there are buttons for "Run" and "Output". The "Run" button is blue. The "Output" section shows the results of the code execution:

```
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__',
 '__package__', '__spec__', 'traceback']
==== Code Execution Successful ===
```

Q.

The screenshot shows a Python code editor interface. On the left is a sidebar with various icons. The main area has a tab labeled "main.py". The code in the editor is:

```
1 result = divmod(10, 2)
2 print(result)
3
```

The "Output" panel on the right shows the results of running the code:

```
(5, 0)
--- Code Execution Successful ---
```

Q.

The screenshot shows a Python code editor interface. On the left is a sidebar with various icons. The main area has a tab labeled "main.py". The code in the editor is:

```
1 result = enumerate([1, 2, 3])
2 print(result)
3 print(list(result))
4
```

The "Output" panel on the right shows the results of running the code:

```
<enumerate object at 0x7e6f94ff76a0>
[(0, 1), (1, 2), (2, 3)]
--- Code Execution Successful ---
```

Q.

The screenshot shows a Python code editor interface. On the left is a sidebar with various icons. The main area has a tab labeled "main.py". The code in the editor is:

```
1 result = dict()
2 result2 = dict(a=1, b=2)
3
4 print(result)
5 print(result2)
6
```

The "Output" panel on the right shows the results of running the code:

```
{}
{'a': 1, 'b': 2}
--- Code Execution Successful ---
```

**Q.**

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Share. The main area is titled "main.py" and contains the following code:

```
1 def filterdata(x):
2     if x > 5:
3         return x
4
5 result = filter(filterdata, (1, 2, 6))
6 print(list(result))
7
```

At the top right, there are buttons for "Run" and "Share". Below the code, the "Output" section shows the results of the execution:

```
[6]
==== Code Execution Successful ====

```

**Q.**

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Share. The main area is titled "main.py" and contains the following code:

```
1 result = hash(21)
2 result2 = hash(22,2)
3
4 print(result)
5 print(result2)
6
```

At the top right, there are buttons for "Run" and "Share". Below the code, the "Output" section shows the results of the execution:

```
21
461168601842737174
==== Code Execution Successful ====

```

**Q.**

The screenshot shows a Python code editor interface. On the left, there is a sidebar with various icons for file operations like Open, Save, and Share. The main area is titled "main.py" and contains the following code:

```
1 info = help()
2 print(info)
3
```

At the top right, there are buttons for "Run" and "Share". Below the code, the "Output" section shows the results of the execution:

```
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/.

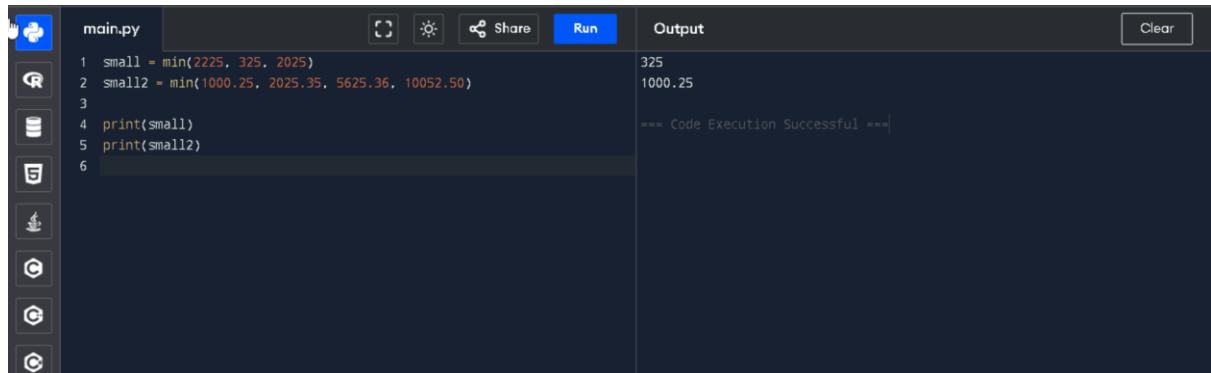
Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

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".

To quit this help utility and return to the interpreter,
enter "q" or "quit".

help> |
```

**Q.**



```
main.py Run Output Clear
1 small = min(2225, 325, 2025)
2 small2 = min(1000.25, 2025.35, 5625.36, 10052.50)
3
4 print(small)
5 print(small2)
6
```

```
325
1000.25
*** Code Execution Successful ***
```

**Q.**



```
main.py Run Output Clear
1 result = set()
2 result2 = set('12')
3 result3 = set('javatpoint')
4
5 print(result)
6 print(result2)
7 print(result3)
8
```

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

```
*** Code Execution Successful ***
```

**Q.**



```
main.py Run Output Clear
1 result = hex(1)
2 result2 = hex(342)
3
4 print(result)
5 print(result2)
6
```

```
0x1
0x156
```

```
*** Code Execution Successful ***
```

**Q.**

The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with various icons. The main area has a tab labeled "main.py". The code is as follows:

```
1 val = id("Javatpoint")
2 val2 = id(1200)
3 val3 = id([25, 336, 95, 236, 92, 3225])
4
5 print(val)
6 print(val2)
7 print(val3)
8
```

On the right, under the "Output" tab, the results are displayed:

```
137612486677616
137612488027664
137612489853312
*** Code Execution Successful ***
```

**Q.**

The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with various icons. The main area has a tab labeled "main.py". The code is as follows:

```
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
13 setattr(student, 'email', 'sohan@abc.com')
14 print(student.email)
15
```

On the right, under the "Output" tab, the results are displayed:

```
102
Sohan
sohan@abc.com
*** Code Execution Successful ***
```

**Q.**

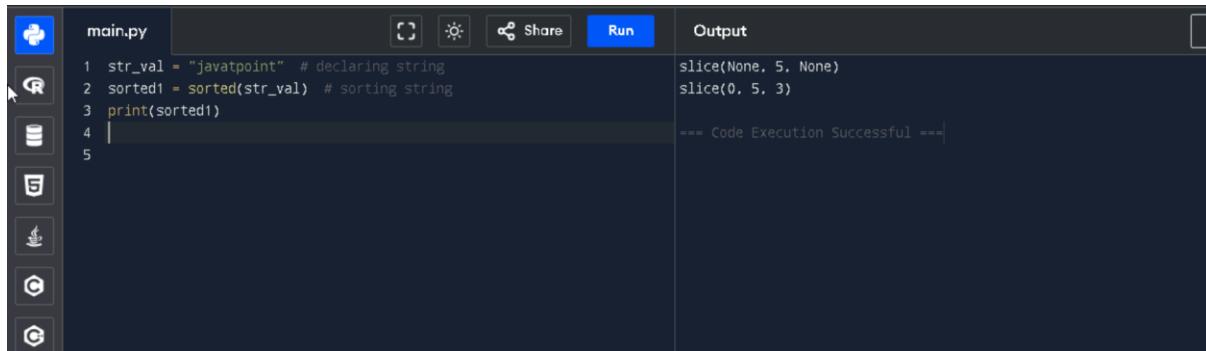
The screenshot shows a Python code editor interface with a dark theme. On the left is a sidebar with various icons. The main area has a tab labeled "main.py". The code is as follows:

```
1 result = slice(5)
2 result2 = slice(0, 5, 3)
3
4 print(result)
5 print(result2)
6
```

On the right, under the "Output" tab, the results are displayed:

```
slice(None, 5, None)
slice(0, 5, 3)
*** Code Execution Successful ***
```

**Q.**



The screenshot shows a Python code editor interface. On the left, there is a vertical toolbar with various icons. The main area is titled "main.py" and contains the following code:

```
1 str_val = "javatpoint" # declaring string
2 sorted1 = sorted(str_val) # sorting string
3 print(sorted1)
4
5
```

At the top right, there are several buttons: a refresh icon, a light mode icon, a "Share" icon, and a "Run" button, which is highlighted in blue. To the right of the code area is an "Output" panel. The output shows the result of the execution:

```
slice(None, 5, None)
slice(0, 5, 3)
==== Code Execution Successful ===
```

Q.

The screenshot shows a dark-themed code editor window. In the top-left, there's a file tree with several Python files (46.py, 47.py, 48.py, 49.py, 50.py, 51.py, 52.py, 53.py, 54.py) under a 'functions' folder. The main editor area contains the following code:

```
functions > 54.py
1 # Python next() Function Example
2 number = iter([256, 32, 82])
3 print(next(number))
4 print(next(number))
5 print(next(number))
```

Below the editor is a terminal window showing the execution of the code:

```
[Done] exited with code=0 in 0.09 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\54.py"
256
32
82
[Done] exited with code=0 in 0.096 seconds
```

The bottom status bar indicates the code is in 'Code' mode, with Ln 5, Col 20, Spaces: 4, UTF-8, CRLF, Python, Go Live, and other system information like ENG, US, 12:41, and 08-02-2025.

Q.

The screenshot shows a dark-themed code editor window. In the top-left, there's an 'EXPLORER' sidebar with a 'functions' folder containing several Python files (44.py, 45.py, 46.py, 47.py, 48.py, 49.py, 50.py, 51.py, 52.py, 53.py, 54.py, 55.py). The file '55.py' is selected and highlighted in blue. The main editor area contains the following code:

```
functions > 55.py
1 # Python int() Function Example
2 val = int(10)
3 val2 = int(10.52)
4 val3 = int('10')
5 print("integer values:", val, val2, val3)
```

Below the editor is a terminal window showing the execution of the code:

```
B2
[Done] exited with code=0 in 0.096 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\55.py"
integer values: 10 10 10
[Done] exited with code=0 in 0.09 seconds
```

The bottom status bar indicates the code is in 'Code' mode, with Ln 5, Col 42, Spaces: 4, UTF-8, CRLF, Python, Go Live, and other system information like ENG, US, 12:41, and 08-02-2025.

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 45.py through 56.py. File 56.py is currently selected and open in the main editor area. The code in 56.py is:

```
functions > 56.py
1 # Python isinstance() Function Example
2 class Student:
3     id = 101
4     name = "John"
5
6     def __init__(self, id, name):
7         self.id = id
8         self.name = name
9
10 student = Student(1010, "John")
11 lst = [12, 34, 5, 6, 767]
12 print(isinstance(student, Student))
13 print(isinstance(lst, Student))
```

The Output tab shows the results of running the code:

```
[Done] exited with code=0 in 0.09 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\56.py"
True
False
[Done] exited with code=0 in 0.092 seconds
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 3, Col 32 Spaces: 4 UTF-8 CRLF Python Go Live", and "12:41 ^ ⌂ ENG US 08-02-2025".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 46.py through 57.py. File 57.py is currently selected and open in the main editor area. The code in 57.py is:

```
functions > 57.py
1 # Python oct() Function Example
2 val = oct(10)
3 print("Octal value of 10:", val)
```

The Output tab shows the results of running the code:

```
False
[Done] exited with code=0 in 0.092 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\57.py"
Octal value of 10: 0o12
[Done] exited with code=0 in 0.122 seconds
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 3, Col 33 Spaces: 4 UTF-8 CRLF Python Go Live", and "12:42 ^ ⌂ ENG US 08-02-2025".

Q.

The screenshot shows a code editor interface with a dark theme. In the Explorer sidebar, there is a 'PYTHON' folder containing several files: 47.py, 48.py, 49.py, 50.py, 51.py, 52.py, 53.py, 54.py, 55.py, 56.py, 57.py, and 58.py. The file 58.py is currently selected and open in the main editor area. The code in 58.py is:

```
# Python ord() Function Example
print(ord('8'))
print(ord('R'))
print(ord('&'))
```

The Output tab shows the results of running the code:

```
[Done] exited with code=0 in 0.122 seconds
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/58.py"
56
82
38
```

The status bar at the bottom right indicates the code is in Python mode, with settings like 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'. A watermark for 'Activate Windows' is visible in the center of the screen.

Q.

The screenshot shows a code editor interface with a dark theme. In the Explorer sidebar, there is a 'PYTHON' folder containing several files: 48.py, 49.py, 50.py, 51.py, 52.py, 53.py, 54.py, 55.py, 56.py, 57.py, 58.py, and 59.py. The file 59.py is currently selected and open in the main editor area. The code in 59.py is:

```
# Python pow() Function Example
print(pow(4, 2))
print(pow(-4, 2))
print(pow(4, -2))
print(pow(-4, -2))
```

The Output tab shows the results of running the code:

```
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/59.py"
16
16
0.0625
0.0625
```

The status bar at the bottom right indicates the code is in Python mode, with settings like 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'. A watermark for 'Activate Windows' is visible in the center of the screen.

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The left sidebar shows a file tree under "EXPLORER" with a "PYTHON" folder containing files like 49.py through 60.py, python.txt, if\_else, loops, reverse\_string, string, eight.py, five.py, four.py, one.py, seven.py, six.py, OUTLINE, and TIMELINE. The main editor area contains a Python script named 60.py:

```
functions > 60.py
1 # Python print() Function Example
2 print("Python is a programming language.")
3 x = 7
4 print("x =", x)
5 y = x
6 print('x =', x, 'y =', y)
```

The "OUTPUT" tab at the bottom shows the results of running the script:

```
[Done] exited with code=0 in 0.12 seconds
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/60.py"
Python is a programming language.
x = 7
x = 7 = y

[Done] exited with code=0 in 0.127 seconds
```

The status bar at the bottom right shows "Ln 6, Col 23" and "12:43".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The left sidebar shows a file tree under "EXPLORER" with a "PYTHON" folder containing files like 50.py through 61.py, python.txt, if\_else, loops, reverse\_string, string, eight.py, five.py, four.py, one.py, seven.py, six.py, OUTLINE, and TIMELINE. The main editor area contains a Python script named 61.py:

```
functions > 61.py
1 # Python range() Function Example
2 print(list(range(0)))
3 print(list(range(4)))
4 print(list(range(1, 7)))
```

The "OUTPUT" tab at the bottom shows the results of running the script:

```
[Done] exited with code=0 in 0.127 seconds
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/61.py"
[]
[0, 1, 2, 3]
[1, 2, 3, 4, 5, 6]

[Done] exited with code=0 in 0.251 seconds
```

The status bar at the bottom right shows "Ln 4, Col 25" and "12:43".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 51.py through 61.py, and 62.py is selected. The main editor pane displays the following Python code:

```
functions > 62.py
1 # Python reversed() Function Example
2 String = 'Java'
3 print(list(reversed(String)))
4 Tuple = ('J', 'a', 'v', 'a')
5 print(list(reversed(Tuple)))
6 Range = range(8, 12)
7 print(list(reversed(Range)))
8 List = [1, 2, 7, 5]
9 print(list(reversed(List)))
```

The Output tab shows the results of running the code:

```
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/62.py"
['a', 'v', 'a', 'J']
['a', 'v', 'a', 'J']
[11, 10, 9, 8]
[5, 7, 2, 1]
[Done] exited with code=0 in 0.114 seconds
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 9, Col 28 Spaces: 4 UTF-8 CRLF Python Go Live", and "12:43 US 08-02-2025".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 52.py through 61.py, and 63.py is selected. The main editor pane displays the following Python code:

```
functions > 63.py
1 # Python round() Function Example
2 print(round(10))
3 print(round(10.8))
4 print(round(6.6))
```

The Output tab shows the results of running the code:

```
[Done] exited with code=0 in 0.114 seconds
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/63.py"
10
11
7
[Done] exited with code=0 in 0.161 seconds
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 4, Col 18 Spaces: 4 UTF-8 CRLF Python Go Live", and "12:44 US 08-02-2025".

Q.

The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files in a 'functions' folder, including 64.py, which is currently selected. The code editor displays the following Python script:

```
functions > 64.py
1 # Python issubclass() Function Example
2 class Rectangle:
3     def __init__(self, rectangleType):
4         print('Rectangle is a', rectangleType)
5
6 class Square(Rectangle):
7     def __init__(self):
8         Rectangle.__init__(self, 'square')
9
10 print(issubclass(Square, Rectangle))
11 print(issubclass(Square, list))
12 print(issubclass(Square, (list, Rectangle)))
13 print(issubclass(Rectangle, (list, Rectangle)))
```

The Output tab shows the results of running the script:

```
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\64.py"
True
False
True
True
```

The status bar at the bottom right indicates the file is 'Code', the line is 13, column is 48, and the date is 08-02-2025.

Q.

The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files in a 'functions' folder, including 65.py, which is currently selected. The code editor displays the following Python script:

```
functions > 65.py
1 # Python str() Function Example
2 print(str(4))
```

The Output tab shows the results of running the script:

```
True
```

The status bar at the bottom right indicates the file is 'Code', the line is 2, column is 14, and the date is 08-02-2025.

Q.

A screenshot of the Visual Studio Code interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 55.py through 65.py, and 66.py is selected. The main editor pane displays the following Python code:

```
functions > 66.py
1 # Python tuple() Function Example
2 t1 = tuple()
3 print('t1=', t1)
4 t2 = tuple([1, 6, 9])
5 print('t2=', t2)
6 t1 = tuple('Java')
7 print('t1=', t1)
8 t1 = tuple({4: 'four', 5: 'five'})
9 print['t1=', t1]
```

The Output tab shows the results of running the code:

```
[Running] python -u "c:/Users/Administrator/Desktop/python/functions/66.py"
t1= ()
t2= (1, 6, 9)
t1= ('J', 'a', 'v', 'a')
t1= (4, 5)

[Done] exited with code=0 in 0.095 seconds
```

The status bar at the bottom right shows "Ln 9, Col 17" and "Activate Windows Go to Settings to activate Windows."

Q.

A screenshot of the Visual Studio Code interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 56.py through 65.py, and 66.py, followed by 67.py, which is selected. The main editor pane displays the following Python code:

```
functions > 67.py
1 # Python type() Function Example
2 List = [4, 5]
3 print(type(List))
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)]
```

The Output tab shows the results of running the code:

```
[Done] exited with code=0 in 0.095 seconds

[Running] python -u "c:/Users/Administrator/Desktop/python/functions/67.py"
<class 'list'>
<class 'dict'>
<class '__main__.Python'>

[Done] exited with code=0 in 0.009 seconds
```

The status bar at the bottom right shows "Ln 11, Col 30" and "Activate Windows Go to Settings to activate Windows."

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The left sidebar shows a file tree under "EXPLORER" with a "PYTHON" folder containing files like 57.py through 68.py. The main editor area contains the following Python code:

```
functions > 68.py
1 # Python vars() Function Example
2 class Python:
3     def __init__(self, x=7, y=9):
4         self.x = x
5         self.y = y
6
7 InstanceOfPython = Python()
8 print(vars(InstanceOfPython))
```

The "OUTPUT" tab in the bottom right shows the terminal output:

```
<class '__main__.Python'>
[Done] exited with code=0 in 0.009 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\68.py"
{'x': 7, 'y': 9}
[Done] exited with code=0 in 0.097 seconds
```

At the bottom, the status bar shows "Ln 8, Col 30" and "12:46".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The left sidebar shows a file tree under "EXPLORER" with a "PYTHON" folder containing files like 59.py through 70.py. The main editor area contains the following Python code:

```
functions > 69.py
1 # Python zip() Function Example
2 numlist = [4, 5, 6]
3 strlist = ['four', 'five', 'six']
4
5 result = zip()
6 resultlist = list(result)
7 print(resultlist)
8
9 result = zip(numlist, strlist)
10 resultSet = set(result)
11 print(resultSet)
```

The "OUTPUT" tab in the bottom right shows the terminal output:

```
[Done] exited with code=0 in 0.087 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\69.py"
[]
{('6', 'six'), (5, 'five'), (4, 'four')}
[Done] exited with code=0 in 0.004 seconds
```

At the bottom, the status bar shows "Ln 11, Col 17" and "12:47".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 59.py through 70.py. File 70.py is open, displaying the following Python code:

```
functions > 70.py
1 # Python Lambda Functions Example
2 add = lambda num: num + 4
3 print(add(6))
4
5 a = lambda x, y: (x * y)
6 print(a(4, 5))
7
8 a = lambda x, y, z: (x + y + z)
9 print(a(4, 5))
```

The Output panel shows the results of running the code:

```
[Done] exited with code=0 in 0.084 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\70.py"
10
20
14
[Done] exited with code=0 in 0.117 seconds
```

At the bottom right, there is an "Activate Windows" message: "Go to Settings to activate Windows." The status bar at the bottom shows "Ln 9, Col 18" and "Python".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 60.py through 71.py. File 71.py is open, displaying the following Python code:

```
functions > 71.py
1 # Python lambda vs def Example
2 def reciprocal(num):
3     return 1 / num
4
5 lambda_reciprocal = lambda num: 1 / num
6
7 print("Def keyword:", reciprocal(6))
8 print("Lambda keyword:", lambda_reciprocal(6))
```

The Output panel shows the results of running the code:

```
[Done] exited with code=0 in 0.117 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\71.py"
Def keyword: 0.1666666666666666
Lambda keyword: 0.1666666666666666
[Done] exited with code=0 in 0.091 seconds
```

At the bottom right, there is an "Activate Windows" message: "Go to Settings to activate Windows." The status bar at the bottom shows "Ln 8, Col 47" and "Python".

Q.

File Edit Selection View Go Run ... ⇐ → python [Administrator]

EXPLORER ...

PYTHON

functions > 72.py

```
1 # Using Lambda Function with filter()
2 list_ = [35, 12, 69, 55, 75, 14, 73]
3 odd_list = list(filter(lambda num: (num % 2 != 0), list_))
4 print("The list of odd numbers is:", odd_list)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Lambda keyword: 0.16666666666666666

[Done] exited with code=0 in 0.091 seconds

[Running] python -u "c:\Users\Administrator\Desktop\python\functions\72.py"

The list of odd numbers is: [35, 69, 55, 75, 73]

[Done] exited with code=0 in 0.124 seconds

Activate Windows  
Go to Settings to activate Windows.

Ln 4, Col 47 Spaces: 4 UTF-8 CRLF Python Go Live

Type here to search

12:48 US 08-02-2025

Q.

File Edit Selection View Go Run ... ⇐ → python [Administrator]

EXPLORER ...

PYTHON

functions > 73.py

```
1 # Using Lambda Function with map()
2 num_list = [1, 2, 3, 4, 5]
3 squared_list = list(map(lambda x: x ** 2, num_list))
4 print("Squared List:", squared_list)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

The list of odd numbers is: [35, 69, 55, 75, 73]

[Done] exited with code=0 in 0.124 seconds

[Running] python -u "c:\Users\Administrator\Desktop\python\functions\73.py"

Squared List: [1, 4, 9, 16, 25]

[Done] exited with code=0 in 0.121 seconds

Activate Windows  
Go to Settings to activate Windows.

Ln 4, Col 37 Spaces: 4 UTF-8 CRLF Python Go Live

Type here to search

12:48 US 08-02-2025

Q.

A screenshot of a code editor window titled "python [Administrator]". The code in the editor is:

```
functions > 74.py
1 # Lambda function with list comprehension
2 squares=[lambda num=num:num**2 for num in range(0,11)]
3
4 for square in squares:
5     print("The square value of all numbers from 0 to 10:",square(),end=" ")
```

The output window shows the results of running the code:

```
[Done] exited with code=0 in 0.113 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\74.py"
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
[Done] exited with code=0 in 0.094 seconds
```

The status bar at the bottom right indicates: Ln 5, Col 74 Spaces: 4 UTF-8 CRLF Python Go Live Q. The date and time are 08-02-2025 12:51 ENG US.

Q.

A screenshot of a code editor window titled "python [Administrator]". The code in the editor is:

```
EXPLORER ... 68.py 69.py 70.py 71.py 74.py 75.py 72.py 73.py
PYTHON functions > 75.py
1 # Lambda function with if-else
2
3 Minimum=lambda x,y:x if(x<y) else y
4 print("The greater number is:",Minimum(35,74))
```

The output window shows the results of running the code:

```
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
[Done] exited with code=0 in 0.094 seconds

[Running] python -u "c:\Users\Administrator\Desktop\python\functions\75.py"
The greater number is: 35

[Done] exited with code=0 in 0.101 seconds
```

The status bar at the bottom right indicates: Ln 4, Col 45 Spaces: 4 UTF-8 CRLF Python Go Live Q. The date and time are 08-02-2025 12:52 ENG US.

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "python [Administrator]". The Explorer sidebar shows a folder named "functions" containing files 69.py through 76.py. The file "76.py" is open in the editor, displaying the following Python code:

```
functions > 76.py
1 # Using lambda with multiple statements
2
3 my_list=[[3,5,8,6],[23,54,12,87],[1,2,4,12,5]]
4
5 sort_list= lambda num: (sorted(n) for n in num)
6
7 third_Largest = lambda num, func: [l[len(l)-1] for l in func(num)]
8
9 result= third_Largest(my_list, sort_list)
10 print("The third largest number from every sub list is:",result)
```

The Output tab shows the results of running the code:

```
The greater number is: 35
[Done] exited with code=0 in 0.101 seconds
[Running] python -u "c:\Users\Administrator\Desktop\python\functions\76.py"
The third largest number from every sub list is: [8, 87, 12]
[Done] exited with code=0 in 0.095 seconds
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 10, Col 65 Spaces: 4 UTF-8 CRLF Python Go Live", and "12:55 US 09-02-2025".

## Python Modules

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "py\_ust". The Explorer sidebar shows a folder named "example\_module" containing files 1.py and example\_module.py. The file "1.py" is open in the editor, displaying the following Python code:

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

The file "example\_module.py" is also open in the editor, showing its contents:

```
example_module.py > square
1 def square(number):
2     result = number ** 2
3     return result
4
```

The Output tab shows the results of running the code:

```
PS D:\py_ust> python -u "d:\py_ust\1.py"
By using the module square of number is: 16
PS D:\py_ust>
```

The status bar at the bottom right shows "Launchpad", "Live Share", "Ln 3, Col 60 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store) Go Live Quokka Prettier", and "17:58 IN 09-02-2025".

Q.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a folder named 'PY\_UST' containing files: \_\_pycache\_\_, 1.py, 2.py, and example\_module.py. The '2.py' tab is active, showing the following Python code:

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

The terminal at the bottom shows the command 'python -u "d:\py\_ust\2.py"' being run, followed by the output 'The value of Euler's number is 2.718281828459045'. The status bar at the bottom right indicates the date as 09-02-2025.

Q.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a folder named 'PY\_UST' containing files: \_\_pycache\_\_, 1.py, 2.py, 3.py, and example\_module.py. The '3.py' tab is active, showing the following Python code:

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

The terminal at the bottom shows the command 'python -u "d:\py\_ust\3.py"' being run, followed by the output 'The value of Euler's number is 2.718281828459045'. The status bar at the bottom right indicates the date as 09-02-2025.

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The workspace is named 'py\_ust'. In the Explorer sidebar, there are files: 'example\_module.py', '1.py', '2.py', '3.py', '4.py', and '4.py' (the active file). The code in '4.py' is:

```
1 from math import e, tau
2
3 print("The value of tau constant is:", tau)
4 print("The value of Euler's number is:", e)
```

The terminal shows the output of running '4.py':

```
PS D:\py_ust> python -u "d:\py_ust\4.py"
The value of tau constant is: 6.283185307179586
The value of Euler's number is: 2.718281828459045
○ PS D:\py_ust>
```

The status bar at the bottom indicates Python 3.11.9 64-bit (Microsoft Store), ENG IN, and the date 09-02-2025.

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The workspace is named 'py\_ust'. In the Explorer sidebar, there are files: 'example\_module.py', '1.py', '2.py', '3.py', '4.py', and '5.py' (the active file). The code in '5.py' is:

```
1 from math import *
2
3 print("Calculating square root:", sqrt(25))
4 print("Calculating tangent of an angle:", tan(pi/6))
```

The terminal shows the output of running '5.py':

```
PS D:\py_ust> python -u "d:\py_ust\5.py"
Calculating square root: 5.0
Calculating tangent of an angle: 0.5773502691896257
○ PS D:\py_ust>
```

The status bar at the bottom indicates Python 3.11.9 64-bit (Microsoft Store), ENG IN, and the date 09-02-2025.

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UT' containing files 1.py through 6.py, and an 'example\_module.py' file. The code editor displays a script named '6.py' with the following content:

```
1 import sys
2
3 print("Path of the sys module in the system is:", sys.path)
```

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\6.py"
Path of the sys module in the system is: ['d:\\py_ust', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\python311.zip', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\DLLs', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\Lib', 'C:\\\\Users\\\\Kumar\\\\anaconda3', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\Lib\\\\site-packages', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\Lib\\\\site-packages\\\\win32', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\Lib\\\\site-packages\\\\win32\\\\lib', 'C:\\\\Users\\\\Kumar\\\\anaconda3\\\\Lib\\\\site-packages\\\\Pythonwin']
```

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UT' containing files 1.py through 7.py, and an 'example\_module.py' file. The code editor displays a script named '7.py' with the following content:

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

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\7.py"
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__', '__setattro__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

Q.

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists files in a folder named 'PY\_UST'. The main editor area displays the contents of '8.py'. The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\8.py"
The number is: 404
The number is: 404
PS D:\py_ust>
```

Q.

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists files in a folder named 'PY\_UST'. The main editor area displays the contents of '9.py'. The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\9.py"
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
PS D:\py_ust> []
```

Q.

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists files in a directory named 'PY\_UST'. The current file is '10.py', which contains the following code:

```
10.py > ...
1 num = [3, 4, 5, 7]
2
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)}")
```

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\10.py"
Traceback (most recent call last):
  File "d:\py_ust\10.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
```

Q.

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists files in a directory named 'PY\_UST'. The current file is '11.py', which contains the following code:

```
11.py > ...
1 def square_root(Number):
2     assert (Number < 0), "Give a positive integer"
3     return Number ** (1/2)
4
5 print(square_root(36))
6 print(square_root(-36))
```

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\11.py"
Traceback (most recent call last):
  File "d:\py_ust\11.py", line 5, in <module>
    print(square_root(36))
    ^^^^^^^^^^
  File "d:\py_ust\11.py", line 2, in square_root
    assert (Number < 0), "Give a positive integer"
    ^^^^^^^^^^
AssertionError: Give a positive integer
```

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "PY\_UST" containing files 1.py through 12.py, and an "example\_module.py".
- Code Editor:** Displays the contents of "12.py". The code defines a function "reciprocal" that attempts to divide 1 by the input number. It handles a ZeroDivisionError by printing a message and returns None for zero.
- Terminal:** Shows the command "PS D:\py\_ust> python -u "d:\py\_ust\12.py"" being run, followed by the output "0.25" and "We cannot divide by zero".
- Bottom Status Bar:** Shows the Python version as 3.11.9 64-bit (Microsoft Store), along with other system information like battery level and time.

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "PY\_UST" containing files 1.py through 13.py, and an "example\_module.py".
- Code Editor:** Displays the contents of "13.py". The code contains a try-except block where it attempts to divide 4 by 0, prints the result, and then handles a ZeroDivisionError by printing a message. A finally block is present to print a message about the finally clause.
- Terminal:** Shows the command "PS D:\py\_ust> python -u "d:\py\_ust\13.py"" being run, followed by the output "Attempting to divide by zero", "This is code of finally clause", and "PS D:\py\_ust>".
- Bottom Status Bar:** Shows the Python version as 3.11.9 64-bit (Microsoft Store), along with other system information like battery level and time.

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UST' containing several Python files (1.py through 14.py) and an 'example\_module.py' file. The current file is '14.py'. The code defines a class 'EmptyError' that inherits from 'RuntimeError'. It has an \_\_init\_\_ method that sets self.arguments to the argument passed to it. A try block attempts to execute var = " ". If an 'EmptyError' is raised, it prints the value of 'var.arguments'. The terminal window shows the command 'python -u "d:\py\_ust\14.py"' being run, followed by a traceback indicating that 'EmptyError' does not inherit from 'BaseException', which is required for catching it. The status bar at the bottom right shows the date and time as 09-02-2025.

```
14.py > ...
1  class EmptyError(RuntimeError):
2      def __init__(self, argument):
3          self.arguments = argument
4
5  var = " "
6  try:
7      raise EmptyError("The variable is empty")
8  except (EmptyError, var):
9      print(var.arguments)
10
11
12
13
14
example_module.py
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\py_ust> python -u "d:\py_ust\14.py"
Traceback (most recent call last):
  File "d:\py_ust\14.py", line 7, in <module>
    raise EmptyError("The variable is empty")
EmptyError: The variable is empty

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "d:\py_ust\14.py", line 8, in <module>
    except (EmptyError, var):
TypeError: catching classes that do not inherit from BaseException is not allowed
PS D:\py_ust>
```

Launchpad Live Share

Search

File Edit Selection View Go Run ...

10 Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store) Go Live Quokka Prettier ENG IN 18:04 09-02-2025

## Python Arrays

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UST' containing several Python files (1.py through 14.py) and an 'example\_module.py' file. The current file is '1.py'. The code defines a list 'a' with values [1, 2, 3, 4, 5, 6]. It then uses print statements to output various indices of the list: 'a[0]', 'a[1]', 'a[2]', 'a[3]', 'a[4]', 'a[5]', 'a[-1]', 'a[-2]', 'a[-3]', and 'a[-4]'. The terminal window shows the command 'python -u "d:\py\_ust\python\_arrays\1.py"' being run, and the output is the list of elements from index 0 to -4. The status bar at the bottom right shows the date and time as 09-02-2025.

```
python_arrays > 1.py > ...
4
5  print("First element is:", a[0])
6  print("Second element is:", a[1])
7  print("Third element is:", a[2])
8  print("Fourth element is:", a[3])
9  print("Last element is:", a[-1])
10 print("Second last element is:", a[-2])
11 print("Third last element is:", a[-3])
12 print("Fourth last element is:", a[-4])
13 print(a[0], a[1], a[2], a[3], a[-1], a[-2], a[-3], a[-4])
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\py_ust> python -u "d:\py_ust\python_arrays\1.py"
First element is: 2
Second element is: 4
Third element is: 5
Fourth element is: 6
Last element is: 6
Second last element is: 4
Third last element is: 2
2 4 5 6 6 5 4 2
PS D:\py_ust>
```

Launchpad Live Share

Search

File Edit Selection View Go Run ...

14 Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store) Go Live Quokka Prettier ENG IN 18:07 09-02-2025

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The Explorer sidebar shows a folder named 'PY\_UST' containing files like 1.py through 14.py, 10.py, 11.py, 12.py, 13.py, 14.py, and example\_module.py. The current file is '2.py' located in the 'python\_arrays' subfolder. The code in '2.py' uses the `arr.array` function to create an array of integers [1, 2, 3, 5, 7, 10], then prints it, changes the first element to 0, prints it again, changes the 6th element to 8, prints it again, and finally changes the elements from index 2 to 5 to [4, 6, 8] and prints the array. The terminal below shows the execution of the script and its output. The status bar at the bottom indicates the file is 13 lines long, has 4 spaces per tab, is using UTF-8 encoding, and is running on Python 3.11.9 64-bit (Microsoft Store). The date and time shown are 09-02-2025.

```
python_arrays > 2.py > ...
3     numbers = arr.array('i', [1, 2, 3, 5, 7, 10])
4
5     numbers[0] = 0
6     print(numbers)
7
8     numbers[5] = 8
9     print(numbers)
10
11    numbers[2:5] = arr.array('i', [4, 6, 8])
12    print(numbers)
13

Second element is: 4
Third element is: 5
Fourth element is: 6
Last element is: 6
Second last element is: 5
Third last element is: 4
Fourth last element is: 2
2 4 5 6 6 5 4 2
PS D:\py_ust> python -u "d:\py_ust\python_arrays\2.py"
array('i', [0, 2, 3, 5, 7, 10])
array('i', [0, 2, 3, 5, 7, 8])
array('i', [0, 2, 4, 6, 8, 8])
PS D:\py_ust> []
```

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The Explorer sidebar shows a folder named 'PY\_UST' containing files like 1.py through 14.py, 11.py, 12.py, 13.py, 14.py, and example\_module.py. The current file is '3.py' located in the 'python\_arrays' subfolder. The code imports the `array` module and creates an array 'number' of integers [1, 2, 3, 3, 4]. It then uses the `del` keyword to delete the second element (index 1) and prints the array. The terminal below shows the execution of the script and its output. The status bar at the bottom indicates the file is 7 lines long, has 4 spaces per tab, is using UTF-8 encoding, and is running on Python 3.11.9 64-bit (Microsoft Store). The date and time shown are 09-02-2025.

```
python_arrays > 3.py > ...
1 import array as arr
2
3 number = arr.array('i', [1, 2, 3, 3, 4])
4
5 del number[2]
6 print(number)
7

PS D:\py_ust> python -u "d:\py_ust\python_arrays\3.py"
PS D:\py_ust> []
```

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure under "PY\_UST" containing "\_\_pycache\_\_" and "python\_arrays". Inside "python\_arrays", there are files 1.py through 14.py and an "example\_module.py".
- Code Editor:** Displays the content of "4.py":

```
import array as arr
a = arr.array('d', [1.1, 2.1, 3.1, 2.6, 7.8])
b = arr.array('d', [3.7, 8.6])
c = arr.array('d')
c = a + b
print("Array c =", c)
```
- Terminal:** Shows the output of running "4.py":

```
PS D:\py_ust> python -u "d:\py_ust\python_arrays\4.py"
Array c = array('d', [1.1, 2.1, 3.1, 2.6, 7.8, 3.7, 8.6])
PS D:\py_ust>
```
- Bottom Status Bar:** Includes icons for Launchpad, Live Share, and system status like ENG IN, 18:09, and 09-02-2025.

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure under "PY\_UST" containing "\_\_pycache\_\_" and "python\_arrays". Inside "python\_arrays", there are files 1.py through 14.py and an "example\_module.py".
- Code Editor:** Displays the content of "5.py":

```
import array as arr
x = arr.array('i', [4, 7, 19, 22])
print("First element:", x[0])
print("Second element:", x[1])
print("Second last element:", x[-1])
```
- Terminal:** Shows the output of running "5.py":

```
PS D:\py_ust> python -u "d:\py_ust\python_arrays\5.py"
First element: 4
Second element: 7
Second last element: 22
PS D:\py_ust>
```
- Bottom Status Bar:** Includes icons for Launchpad, Live Share, and system status like ENG IN, 18:09, and 09-02-2025.

## Python Decorators

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The Explorer sidebar shows a project structure with files like 1.py, 2.py, 3.py, 4.py, 5.py, 6.py, 7.py, 8.py, 9.py, 10.py, 11.py, 12.py, 13.py, 14.py, and example\_module.py. The 1.py file is open in the editor, containing the following code:

```
def func1(msg):
    print(msg)

func1("Hii, welcome to function")

func2 = func1
func2("Hii, welcome to function")
```

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\1.py"
Hii, welcome to function
Hii, welcome to function
PS D:\py_ust>
```

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The Explorer sidebar shows a project structure with files like 1.py, 2.py, 3.py, 4.py, 5.py, 6.py, 7.py, 8.py, 9.py, 10.py, 11.py, 12.py, 13.py, 14.py, and example\_module.py. The 2.py file is open in the editor, containing the following code:

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

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

func1()
func2()
func()
```

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\2.py"
This is first child function
This is second child function
We are in first function
PS D:\py_ust>
```

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UST' containing several Python files (1.py through 14.py) and a file named 'example\_module.py'. The current file being edited is '3.py' under the 'python\_decorators' subfolder. The code in '3.py' defines two functions: 'sub' and 'operator'. The 'sub' function takes a single argument 'x' and returns 'x - 1'. The 'operator' function takes two arguments, 'func' and 'x', and returns a new function that prints the result of calling 'func(x)' and then adds 10 to it. The terminal at the bottom shows the output of running the script: 'PS D:\py\_ust> python -u "d:\py\_ust\python\_decorators\3.py"' followed by the numbers 9 and 21, indicating the results of the two operations defined in the code.

```
def sub(x):
    return x - 1

def operator(func, x):
    temp = func(x)
    return temp + 10

print(operator(sub, 10))
print(operator(add, 20))
```

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UST' containing several Python files (1.py through 14.py) and a file named 'example\_module.py'. The current file being edited is '4.py' under the 'python\_decorators' subfolder. The code in '4.py' defines a function named 'hello' that prints the string 'Hello' and then returns a new function named 'new'. When this new function is called, it prints 'Hello' again. The terminal at the bottom shows the output of running the script: 'PS D:\py\_ust> python -u "d:\py\_ust\python\_decorators\4.py"' followed by 'Hello' and 'Hello', indicating the function's behavior.

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

new = hello()
new()
```

Q.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PY\_UST' containing several Python files (1.py through 14.py) and a file named 'example module.ov'. The current file is '5.py', which contains the following code:

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

The terminal at the bottom shows the output of running the code:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\5.py"
2.0
PS D:\py_ust>
```

Q.

A screenshot of the Visual Studio Code interface, similar to the previous one but with a different file selected. The Explorer sidebar shows the same structure. The current file is '6.py', which contains the following code:

```
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 divide(50,10)
```

The terminal at the bottom shows the output of running the code twice:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\6.py"
PS D:\py_ust> python -u "d:\py_ust\python_decorators\6.py"
5.0
PS D:\py_ust>
```

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure under "PY\_UST" containing "\_\_pycache\_\_", "python\_arrays", and "python\_decorators". Inside "python\_decorators", there are files 1.py through 7.py and a "decorator.py" file.
- Code Editor:** Displays the contents of file 7.py, which contains:

```
from decorator import do_twice # type: ignore
@do_twice
def say_hello():
    print("Hello There")
say_hello()
```
- Terminal:** Shows the output of running the code:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\7.py"
Hello There
Hello There
```
- Bottom Status Bar:** Shows the Python version (3.11.9 64-bit (Microsoft Store)), status indicators (ENG IN), and the date (09-02-2025).

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure under "PY\_UST" containing "\_\_pycache\_\_", "python\_arrays", and "python\_decorators". Inside "python\_decorators", there are files 1.py through 8.py and a "mod\_decorator.py" file.
- Code Editor:** Displays the contents of file 8.py, which contains:

```
from mod_decorator import do_twice
@do_twice
def display(name):
    print(f"Hello {name}")
display()
```
- Terminal:** Shows the output of running the code:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\8.py"
Traceback (most recent call last):
  File "d:\py_ust\python_decorators\8.py", line 7, in <module>
    display()
  File "d:\py_ust\python_decorators\mod_decorator.py", line 3, in wrapper_do_twice
    func()
TypeError: display() missing 1 required positional argument: 'name'
```
- Bottom Status Bar:** Shows the Python version (3.11.9 64-bit (Microsoft Store)), status indicators (ENG IN), and the date (09-02-2025).

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "File Edit Selection View Go Run ...". The search bar contains "py\_ust". The Explorer sidebar on the left shows a project structure under "PY\_UST" with files like mod\_decorator.py, 1.py through 9.py, and 9.ov. The main editor window displays "mod\_decorator.py" with the following code:

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

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\9.py"
Hello John
Hello John
PS D:\py_ust>
```

The status bar at the bottom right shows "1914 ENG IN 09-02-2025".

Q.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "File Edit Selection View Go Run ...". The search bar contains "py\_ust". The Explorer sidebar on the left shows a project structure under "PY\_UST" with files like mod\_decorator.py, 1.py through 9.py, and 10.py. The main editor window displays "10.py" with the following code:

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

The terminal at the bottom shows the output of running the script:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\10.py"
Hello John
Hello John
PS D:\py_ust> python -u "d:\py_ust\python_decorators\9.py"
We are created greeting
We are created greeting
PS D:\py_ust>
```

The status bar at the bottom right shows "1915 ENG IN 09-02-2025".

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure for "PY\_UST" containing "\_\_pycache\_\_", "python\_arrays", "python\_decorators", and "1.py" through "10.py".
- Code Editor:** Displays "11.py" with the following Python code:

```
1  class Student:
2      def __init__(self, name, grade):
3          self.name = name
4          self.grade = grade
5
6      @property
7          def display(self):
8              return self.name + " got grade " + self.grade
9
10 stu = Student("John", "B")
11 print("Name of the student: ", stu.name)
12 print("Grade of the student: ", stu.grade)
13 print(stu.display)
```
- Terminal:** Shows the output of running the code:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\11.py"
Name of the student: John
Grade of the student: B
John got grade B
PS D:\py_ust>
```
- Bottom Status Bar:** Includes icons for Launchpad, Live Share, and various system status indicators like battery level and signal strength.

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure for "PY\_UST" containing "python\_decorators", "1.py" through "10.py", and "12.py".
- Code Editor:** Displays "12.py" with the following Python code:

```
1  class Person:
2      @staticmethod
3          def hello():
4              print("Hello Peter")
5
6 per = Person()
7 per.hello()
8 Person.hello()
```
- Terminal:** Shows the output of running the code:

```
PS D:\py_ust> python -u "d:\py_ust\python_decorators\12.py"
Hello Peter
Hello Peter
PS D:\py_ust>
```
- Bottom Status Bar:** Includes icons for Launchpad, Live Share, and various system status indicators like battery level and signal strength.

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "PY\_UST" containing subfolders like "\_\_pycache\_\_" and "python\_decorators". Inside "python\_decorators", there are files 1.py through 13.py.
- Code Editor:** The active file is "13.py", which contains Python code for a "repeat" decorator. The code defines a "repeat" function that takes a parameter "num" and returns a decorator. The decorator uses the `functools.wraps` function and a wrapper function to repeat the original function "num" times. A test call to `function1("John")` is shown.
- Terminal:** The terminal shows two command-line executions of the script. Both executions result in the output "John" being printed five times, corresponding to the value of "num" in the decorator.
- Status Bar:** The status bar at the bottom indicates the file is "16, Col 18", the encoding is "UTF-8", and the Python version is "3.11.9 64-bit (Microsoft Store)".

Q.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "PY\_UST" containing a "python\_decorators" folder. Inside "python\_decorators", there are files 6.py through 14.py and "mod\_decorator.py".
- Code Editor:** The active file is "14.py", which contains Python code for a "count\_function" decorator. The code defines a "count\_function" function that uses the `functools.wraps` function and a wrapper function to count the number of calls to the decorated function. It prints the count each time the function is called. A test call to `say\_hello()` is shown.
- Terminal:** The terminal shows two command-line executions of the script. The first execution shows the initial state with "Call 1 of 'say\_hello'". The second execution shows the state after two calls, with "Call 2 of 'say\_hello'" and "Say Hello" printed.
- Status Bar:** The status bar at the bottom indicates the file is "Ln 1, Col 17", the encoding is "UTF-8", and the Python version is "3.11.9 64-bit (Microsoft Store)".

Q.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The Explorer sidebar on the left displays a file tree for a directory named 'PY\_UST' containing several Python files (1.py through 15.py) and a folder 'python\_decorators'. The 'python\_decorators' folder contains 'mod\_decorator.py' and '15.py'. The 'TERMINAL' tab at the bottom shows the output of running '15.py' in a PowerShell window (PS D:\py\_ust> python -u "d:\py\_ust\python\_decorators\15.py"). The terminal output shows three calls to the 'say\_hello' function, each printing 'Say Hello'. The status bar at the bottom right indicates the system language is English (ENG), the date is 09-02-2025, and the time is 19:21.

```
File Edit Selection View Go Run ... ← → ⌂ py_ust
EXPLORER PY_UST python_decorators
1.py 2.py 3.py 4.py 5.py 6.py 7.py 8.py 9.py 10.py 11.py 12.py 14.py 15.py mod_decorator.py
python_decorators > 15.py > Count_Calls > __call__
1 import functools
2
3 class Count_Calls:
4     def __init__(self, func):
5         functools.update_wrapper(self, func)
6         self.func = func
7         self.num_calls = 0
8     def __call__(self, *args, **kwargs):
9         self.num_calls += 1
10        print(f"Call {self.num_calls} of {self.func.__name__}!")
11        return self.func(*args, **kwargs)
12 @Count_Calls
13 def say_hello():
14     print("Say Hello")
15 say_hello()
16 say_hello()
17 say_hello()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\py_ust> python -u "d:\py_ust\python_decorators\15.py"
Call 1 of 'say_hello'
Say Hello
Call 2 of 'say_hello'
Say Hello
Call 3 of 'say_hello'
Say Hello
PS D:\py_ust>

... OUTLINE Count_Calls __init__ func num_calls
TIMELINE
RUNNING TASKS
Launchpad Live Share
Search
Ln 9, Col 28 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store) Go Live Quokka Prettier ENG IN 19:21 09-02-2025
```