

# Python Project

## Python Code for User-Defined function

main.py	Output
<pre>1 def square(num): 2     return num**2 3 object_ = square(6) 4 print("The square of the given number is: ", object_)</pre>	<pre>The square of the given number is: 36 === Code Execution Successful ===</pre>

## Python Code for calling a function

main.py	Output
<pre>1 def a_function(string): 2     return len(string) 3 print("Length of the string Functions is: ", a_function 4       ("Functions")) 5 print("Length of the string Python is: ", a_function("Python"))</pre>	<pre>Length of the string Functions is: 9 Length of the string Python is: 6 === Code Execution Successful ===</pre>

## Python Code for Pass by Reference vs. Value

main.py	Output
<pre>1 def square(item_list): 2     squares = [] 3     for l in item_list: 4         squares.append(l**2) 5     return squares 6 my_list = [17, 52, 8] 7 my_result = square(my_list) 8 print("Squares of the list are: ", my_result)</pre>	<pre>Squares of the list are: [289, 2704, 64] === Code Execution Successful ===</pre>

## Python code to demonstrate the use of default arguments

main.py	Output
<pre>1 def function(n1, n2=20): 2     print("number 1 is: ", n1) 3     print("number 2 is: ", n2) 4 print("Passing only one argument") 5 function(30) 6 print("Passing two arguments") 7 function(50, 30)</pre>	<pre>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 ===</pre>

## Python code to demonstrate the use of keyword arguments

main.py	Output
<pre>1 def function(n1, n2): 2     print("number 1 is: ", n1) 3     print("number 2 is: ", n2) 4 print("Without using keyword") 5 function(50, 30) 6 print("With using keyword") 7 function(n2=50, n1=30)</pre>	<pre>Without using keyword number 1 is: 50 number 2 is: 30 With using keyword number 1 is: 30 number 2 is: 50 === Code Execution Successful ===</pre>

# Python Project

Python code to demonstrate the use of default arguments

main.py	Run	Output
<pre>1 def function(n1, n2): 2     print("number 1 is: ", n1) 3     print("number 2 is: ", n2) 4 print("Passing out of order arguments") 5 function(30, 20) 6 print("Passing only one argument") 7 try: 8     function(30) 9 except: 10    print("Function needs two positional arguments")</pre>		<pre>Passing out of order arguments number 1 is: 30 number 2 is: 20 Passing only one argument Function needs two positional arguments  === Code Execution Successful ===</pre>

Python code to demonstrate the use of variable-length arguments

main.py	Run	Output	Clear
<pre>1 def function(*args_list): 2     ans = [] 3     for l in args_list: 4         ans.append(l.upper()) 5     return ans 6 object = function('Python', 'Functions', 'tutorial') 7 print(object) 8 def function(**kargs_list): 9     ans = [] 10    for key, value in kargs_list.items(): 11        ans.append([key, value]) 12    return ans 13 object = function(First="Python", Second="Functions", Third 14                  ="Tutorial") 14 print(object)</pre>		<pre>['PYTHON', 'FUNCTIONS', 'TUTORIAL'] [['First', 'Python'], ['Second', 'Functions'], ['Third', 'Tutorial']]  === Code Execution Successful ===</pre>	

Python code to demonstrate the use of return statements

main.py	Run	Output
<pre>1 def square(num): 2     return num**2 3 print("With return statement") 4 print(square(52)) 5 def square(num): 6     num**2 7 print("Without return statement") 8 print(square(52))</pre>		<pre>With return statement 2704 Without return statement None  === Code Execution Successful ===</pre>

Python code to demonstrate anonymous functions

main.py	Run	Output
<pre>1 lambda_ = lambda argument1, argument2: argument1 + argument2 2 print("Value of the function is : ", lambda_(20, 30)) 3 print("Value of the function is : ", lambda_(40, 50))</pre>		<pre>Value of the function is : 50 Value of the function is : 90  === Code Execution Successful ===</pre>

# Python Project

Python code to demonstrate scope and lifetime of variables

main.py	Output
<pre>1 def number(): 2     num = 50 3     print("Value of num inside the function: ", num) 4 num = 10 5 number() 6 print("Value of num outside the function:", num)</pre>	<pre>Value of num inside the function: 50 Value of num outside the function: 10  === Code Execution Successful ===</pre>

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

main.py	Output
<pre>1 def word(): 2     string = 'Python functions tutorial' 3     x = 5 4     def number(): 5         print(string) 6         print(x) 7     number() 8 word()</pre>	<pre>Python functions tutorial 5  === Code Execution Successful ===</pre>

Python abs() Function

main.py	Output
<pre>1 integer = -20 2 print('Absolute value of -40 is:', abs(integer)) 3 floating = -20.83 4 print('Absolute value of -40.83 is:', abs(floating))</pre>	<pre>Absolute value of -40 is: 20 Absolute value of -40.83 is: 20.83  === Code Execution Successful ===</pre>

Python all() Function

main.py	Output
<pre>1 k = [1, 3, 4, 6] 2 print(all(k)) 3 k = [0, False] 4 print(all(k)) 5 k = [1, 3, 7, 0] 6 print(all(k)) 7 k = [0, False, 5] 8 print(all(k)) 9 k = [] 10 print(all(k))</pre>	<pre>True False False False True  === Code Execution Successful ===</pre>

Python bin() Function

main.py	Output
<pre>1 x = 10 2 y = bin(x) 3 print(y)</pre>	<pre>0b1010  === Code Execution Successful ===</pre>

# Python Project

## Python bool()

main.py	Output
<pre>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))</pre>	<pre>[] is False [0] is True 0.0 is False None is False True is True Easy string is True  === Code Execution Successful ===</pre>

## Python bytes()

main.py	Output
<pre>1 string = "Hello World." 2 array = bytes(string, 'utf-8') 3 print(array)</pre>	<pre>b'Hello World.'  === Code Execution Successful ===</pre>

## Python callable()

main.py	Output
<pre>1 x = 8 2 print(callable(x))</pre>	<pre>False  === Code Execution Successful ===</pre>

## Python compile()





main.py	Output
<pre>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)</pre>	<pre>&lt;class 'code'&gt; sum = 15  === Code Execution Successful ===</pre>

## Python exec()





main.py	Output
<pre>1 x = 8 2 exec('print(x==8)') 3 exec('print(x+4)')</pre>	<pre>True 12  === Code Execution Successful ===</pre>

# Python Project





## Python sum()

main.py		Output
<pre>1 a = 1 2 b = 2 3 c = 4 4 initial_value = 10 5 s = sum([a, b, c]) 6 print(s) 7 s = sum([a, b, c], initial_value) 8 print(s)</pre>	   Share 	<pre>7 17  === Code Execution Successful ===</pre>





## Python any() Function

main.py		Output
<pre>1 l = [4, 3, 2, 0] 2 print(any(l)) 3 l = [0, False] 4 print(any(l)) 5 l = [0, False, 5] 6 print(any(l)) 7 l = [] 8 print(any(l))</pre>	   Share 	<pre>True False True False  === Code Execution Successful ===</pre>





## Python ascii() Function

main.py		Output
<pre>1 normalText = 'Python is interesting' 2 print(ascii(normalText)) 3 otherText = 'Pythøn is interesting' 4 print(ascii(otherText)) 5 print('Pyth\x66n is interesting')</pre>	   Share 	<pre>'Python is interesting' 'Pyth\u00f6dn is interesting' Pythön is interesting  === Code Execution Successful ===</pre>

## Python bytearray()

main.py		Output
<pre>1 string = "Python is a programming language." 2 arr = bytearray(string, 'utf-8') 3 print(arr)</pre>	   Share 	<pre>bytearray(b'Python is a programming language.')  === Code Execution Successful ===</pre>

## Python eval()

main.py		Output
<pre>1 x = 8 2 print(eval('x + 1'))</pre>	   Share 	<pre>9  === Code Execution Successful ===</pre>

# Python Project

## Python float()

main.py		Output
<pre>1 print(float(9)) 2 print(float(8.19)) 3 print(float("-24.27")) 4 print(float(" -17.19\n")) 5 print(float("xyz"))</pre>	<div>Run</div>	<div>ERROR!</div> <div>9.0</div> <div>8.19</div> <div>-24.27</div> <div>-17.19</div> <div>Traceback (most recent call last):</div> <div>File "&lt;main.py&gt;", line 5, in &lt;module&gt;</div> <div>ValueError: could not convert string to float: 'xyz'</div> <div>=== Code Exited With Errors ===</div>

## Python format() Function

main.py		Output
<pre>1 print(format(123, "d")) 2 print(format(123.4567898, "f")) 3 print(format(12, "b"))</pre>	<div>Run</div>	<div>123</div> <div>123.456790</div> <div>1100</div> <div>=== Code Execution Successful ===</div>

## Python frozenset()

main.py		Output
<pre>1 letters = ('m', 'r', 'o', 't', 's') 2 fSet = frozenset(letters) 3 print('Frozen set is:', fSet) 4 print('Empty frozen set is:', frozenset())</pre>	<div>Run</div>	<div>Frozen set is: frozenset({'m', 's', 't', 'o', 'r'})</div> <div>Empty frozen set is: frozenset()</div> <div>=== Code Execution Successful ===</div>

## Python getattr() Function

main.py		Output
<pre>1 class Details: 2     age = 22 3     name = "Phill" 4 details = Details() 5 print('The age is:', getattr(details, "age")) 6 print('The age is:', details.age)</pre>	<div>Run</div>	<div>The age is: 22</div> <div>The age is: 22</div> <div>=== Code Execution Successful ===</div>

## Python globals() Function

main.py		Output
<pre>1 age = 22 2 globals()['age'] = 22 3 print('The age is:', age)</pre>	<div>Run</div>	<div>The age is: 22</div> <div>=== Code Execution Successful ===</div>

# Python Project

## Python hasattr() Function

main.py	Output
<pre>1 l = [4, 3, 2, 0] 2 print(any(l)) 3 l = [0, False] 4 print(any(l)) 5 l = [0, False, 5] 6 print(any(l)) 7 l = [] 8 print(any(l))</pre>	<pre>True False True False  === Code Execution Successful ===</pre>

## Python iter() Function

main.py	Output
<pre>1 list = [1, 2, 3, 4, 5] 2 listIter = iter(list) 3 print(next(listIter)) 4 print(next(listIter)) 5 print(next(listIter)) 6 print(next(listIter)) 7 print(next(listIter))</pre>	<pre>1 2 3 4 5  === Code Execution Successful ===</pre>

## Python len() Function

main.py	Output
<pre>1 strA = 'Python' 2 print(len(strA))</pre>	<pre>6  === Code Execution Successful ===</pre>

## Python list()

main.py	Output
<pre>1 print(list()) 2 String = 'abcde' 3 print(list(String)) 4 Tuple = (1, 2, 3, 4, 5) 5 print(list(Tuple)) 6 List = [1, 2, 3, 4, 5] 7 print(list(List))</pre>	<pre>[] ['a', 'b', 'c', 'd', 'e'] [1, 2, 3, 4, 5] [1, 2, 3, 4, 5]  === Code Execution Successful ===</pre>

## Python locals() Function

main.py	Output
<pre>1 def localsAbsent(): 2     return locals() 3 def localsPresent(): 4     present = True 5     return locals() 6 print('localsNotPresent:', localsAbsent()) 7 print('localsPresent:', localsPresent())</pre>	<pre>localsNotPresent: {} localsPresent: {'present': True}  === Code Execution Successful ===</pre>

# Python Project

## Python map() Function

main.py	Output
<pre>1 def calculateAddition(n): 2     return n + n 3 numbers = (1, 2, 3, 4) 4 result = map(calculateAddition, numbers) 5 print(result) 6 numbersAddition = set(result) 7 print(numbersAddition)</pre>	<pre>&lt;map object at 0x7dbd2b00d540&gt; {8, 2, 4, 6}  === Code Execution Successful ===</pre>

## Python memoryview () Function

main.py	Output
<pre>1 randomByteArray = bytearray('ABC', 'utf-8') 2 mv = memoryview(randomByteArray) 3 print(mv[0]) 4 print(bytes(mv[0:2])) 5 print(list(mv[0:3]))</pre>	<pre>65 b'AB' [65, 66, 67]  === Code Execution Successful ===</pre>

## Python object()

main.py	Output	Clear
<pre>1 python = object() 2 print(type(python)) 3 print(dir(python))</pre>	<pre>&lt;class 'object'&gt; ['__class__', '__delattr__', '__dir__', '__doc__', '__eq__',  '__format__', '__ge__', '__getattr__', '__getstate__',  '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__',  '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',  '__repr__', '__setattr__', '__sizeof__', '__str__',  '__subclasshook__']  === Code Execution Successful ===</pre>	

## Python chr() Function

main.py	Output
<pre>1 result = chr(102) 2 result2 = chr(112) 3 print(result) 4 print(result2) 5 print("is it string type:", type(result) is str)</pre>	<pre>f p is it string type: True  === Code Execution Successful ===</pre>

## Python complex() function

main.py	Output
<pre>1 a = complex(1) 2 b = complex(1, 2) 3 print(a) 4 print(b)</pre>	<pre>(1+0j) (1+2j)  === Code Execution Successful ===</pre>



# Python Project

## Python delattr() Function

main.py	Output
<pre>1 class Student: 2     id = 101 3     name = "Pranshu" 4     email = "pranshu@abc.com" 5     def getinfo(self): 6         print(self.id, self.name, self.email) 7 s = Student() 8 s.getinfo() 9 delattr(Student, 'course') 10 s.getinfo()</pre>	<pre>101 Pranshu pranshu@abc.com ERROR! Traceback (most recent call last):   File "&lt;main.py&gt;", line 9, in &lt;module&gt; AttributeError: type object 'Student' has no attribute 'course'  === Code Exited With Errors ===</pre>

## Python dir() Function

main.py	Output
<pre>1 att = dir() 2 print(att)</pre>	<pre>['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'traceback']  === Code Execution Successful ===</pre>

## Python divmod() Function

main.py	Output
<pre>1 result = divmod(10,2) 2 print(result)</pre>	<pre>(5, 0)  === Code Execution Successful ===</pre>

## Python enumerate() Function

main.py	Output
<pre>1 result = enumerate([1,2,3]) 2 print(result) 3 print(list(result))</pre>	<pre>&lt;enumerate object at 0x7912188a36a0&gt; [(0, 1), (1, 2), (2, 3)]  === Code Execution Successful ===</pre>

## Python dict()





main.py	Output
<pre>1 result = dict() 2 result2 = dict(a=1, b=2) 3 print(result) 4 print(result2)</pre>	<pre>{ } {'a': 1, 'b': 2}  === Code Execution Successful ===</pre>

## Python filter() Function





main.py	Output
<pre>1 def filterdata(x): 2     if x &gt; 5: 3         return x 4 result = filter(filterdata, (1, 2, 6)) 5 print(list(result))</pre>	<pre>[6]  === Code Execution Successful ===</pre>

# Python Project





## Python hash() Function

main.py		Output
<pre>1 result = hash(21) 2 result2 = hash(22.2) 3 print(result) 4 print(result2)</pre>	   Share 	21 461168601842737174  === Code Execution Successful ===





## Python help() Function

main.py		Output
<pre>1 info = help() 2 print(info)</pre>	   Share 	Welcome to Python 3.12's help utility! If this is your first time using Python, you should definitely check out the tutorial at <a href="https://docs.python.org/3.12/tutorial/">https://docs.python.org/3.12/tutorial/</a> .  Enter the name of any module, keyword, or topic to get help on it. You can enter multiple modules, keywords, or topics, separated by a space. For example, enter 'modules spam'. To get a list of the modules whose name or summary contain a given string, enter 'modules spam'.  To quit this help utility and return to the interpreter, enter 'q' or 'quit'.  help>





## Python min() Function

main.py		Output
<pre>1 small = min(2225, 325, 2025) 2 small2 = min(1000.25, 2025.35, 5625.36, 10052.50) 3 print(small) 4 print(small2)</pre>	   Share 	325 1000.25  === Code Execution Successful ===

## Python set() Function

main.py		Output
<pre>1 result = set() 2 result2 = set('12') 3 result3 = set('javatpoint') 4 print(result) 5 print(result2) 6 print(result3)</pre>	   Share 	set() {'2', '1'} {'j', 'n', 't', 'p', 'i', 'o', 'v', 'a'}  === Code Execution Successful ===

## Python hex() Function

main.py		Output
<pre>1 result = hex(1) 2 result2 = hex(342) 3 print(result) 4 print(result2)</pre>	   Share 	0x1 0x156  === Code Execution Successful ===

# Python Project

## Python id() Function

main.py	Output
<pre>1 val = id("Javatpoint") 2 val2 = id(1200) 3 val3 = id([25, 336, 95, 236, 92, 3225]) 4 print(val) 5 print(val2) 6 print(val3)</pre>	<pre>139086715130608 139086716481040 139086718323136  === Code Execution Successful ===</pre>

## Python setattr() Function

main.py	Output
<pre>1 class Student: 2     id = 0 3     name = "" 4     def __init__(self, id, name): 5         self.id = id 6         self.name = name 7 student = Student(102, "Sohan") 8 print(student.id) 9 print(student.name) 10 setattr(student, 'email', 'sohan@abc.com') 11 print(student.email)</pre>	<pre>102 Sohan sohan@abc.com  === Code Execution Successful ===</pre>

## Python slice() Function

main.py	Output
<pre>1 result = slice(5) 2 result2 = slice(0, 5, 3) 3 print(result) 4 print(result2)</pre>	<pre>slice(None, 5, None) slice(0, 5, 3)  === Code Execution Successful ===</pre>

## Python sorted() Function




main.py	Output
<pre>1 str = "javatpoint" 2 sorted1 = sorted(str) 3 print(sorted1)</pre>	<pre>['a', 'a', 'i', 'j', 'n', 'o', 'p', 't', 't', 'v']  === Code Execution Successful ===</pre>

## Python next() Function




main.py	Output
<pre>1 number = iter([256, 32, 82]) 2 item = next(number) 3 print(item) 4 item = next(number) 5 print(item) 6 item = next(number) 7 print(item)</pre>	<pre>256 32 82  === Code Execution Successful ===</pre>

# Python Project




## Python int() Function

main.py		Output
<pre>1 val = int(10) 2 val2 = int(10.52) 3 val3 = int('10') 4 print("integer values :", val, val2, val3)</pre>	  	<pre>integer values : 10 10 10  === Code Execution Successful ===</pre>




## Python isinstance() function

main.py		Output
<pre>1 class Student: 2     id = 101 3     name = "John" 4 def __init__(self, id, name): 5     self.id = id 6     self.name = name 7 student = Student(1010, "John") 8 lst = [12, 34, 5, 6, 767] 9 print(isinstance(student, Student)) 10 print(isinstance(lst, Student))</pre>	  	<pre>True False  === Code Execution Successful ===</pre>




## Python oct() function

main.py		Output
<pre>1 val = oct(10) 2 print("Octal value of 10:",val)</pre>	  	<pre>Octal value of 10: 0o12  === Code Execution Successful ===</pre>

## Python ord() function

main.py		Output
<pre>1 print(ord('8')) 2 print(ord('R')) 3 print(ord('&amp;'))</pre>	  	<pre>56 82 38  === Code Execution Successful ===</pre>

## Python pow() function

main.py		Output
<pre>1 print(pow(4, 2)) 2 print(pow(-4, 2)) 3 print(pow(4, -2)) 4 print(pow(-4, -2))</pre>	  	<pre>16 16 0.0625 0.0625  === Code Execution Successful ===</pre>

# Python Project

## Python print() function

main.py		Output
<pre>1 print("Python is programming language.") 2 x = 7 3 print("x =", x) 4 y = x 5 print('x =', x, '= y')</pre>		Python is programming language. x = 7 x = 7 = y  === Code Execution Successful ===

## Python range() function

main.py		Output
<pre>1 print(list(range(0))) 2 print(list(range(4))) 3 print(list(range(1, 7)))</pre>		[] [0, 1, 2, 3] [1, 2, 3, 4, 5, 6]  === Code Execution Successful ===

## Python reversed() function

main.py		Output
<pre>1 String = 'Java' 2 print(list(reversed(String))) 3 Tuple = ('J', 'a', 'v', 'a') 4 print(list(reversed(Tuple))) 5 Range = range(8, 12) 6 print(list(reversed(Range))) 7 List = [1, 2, 7, 5] 8 print(list(reversed(List)))</pre>		['a', 'v', 'a', 'J'] ['a', 'v', 'a', 'J'] [11, 10, 9, 8] [5, 7, 2, 1]  === Code Execution Successful ===

## Python round() Function

main.py		Output
<pre>1 print(round(10)) 2 print(round(10.8)) 3 print(round(6.6))</pre>		10 11 7  === Code Execution Successful ===

## Python isinstance() Function

main.py		Output
<pre>1 class Rectangle: 2     def __init__(rectangleType): 3         print('Rectangle is a ', rectangleType) 4 class Square(Rectangle): 5     def __init__(self): 6         Rectangle.__init__('square') 7 print(isinstance(Square, Rectangle)) 8 print(isinstance(Square, list)) 9 print(isinstance(Square, (list, Rectangle))) 10 print(isinstance(Rectangle, (list, Rectangle)))</pre>		True False True True  === Code Execution Successful ===

# Python Project

## Python tuple() Function

main.py	Output
<pre>1 t1 = tuple() 2 print('t1=', t1) 3 t2 = tuple([1, 6, 9]) 4 print('t2=', t2) 5 t1 = tuple('Java') 6 print('t1=', t1) 7 t1 = tuple({4: 'four', 5: 'five'}) 8 print('t1=', t1)</pre>	<pre>t1= () t2= (1, 6, 9) t1= ('J', 'a', 'v', 'a') t1= (4, 5)  === Code Execution Successful ===</pre>

## Python type() Function

main.py	Output
<pre>1 List = [4, 5] 2 print(type(List)) 3 Dict = {4: 'four', 5: 'five'} 4 print(type(Dict)) 5 class Python: 6     a = 0 7 InstanceOfPython = Python() 8 print(type(InstanceOfPython))</pre>	<pre>&lt;class 'list'&gt; &lt;class 'dict'&gt; &lt;class '__main__.Python'&gt;  === Code Execution Successful ===</pre>

## Python vars() Function

main.py	Output
<pre>1 class Python: 2     def __init__(self, x = 7, y = 9): 3         self.x = x 4         self.y = y 5 InstanceOfPython = Python() 6 print(vars(InstanceOfPython))</pre>	<pre>{'x': 7, 'y': 9}  === Code Execution Successful ===</pre>

## Python zip() Function

main.py	Output
<pre>1 numList = [4, 5, 6] 2 strList = ['four', 'five', 'six'] 3 result = zip() 4 resultList = list(result) 5 print(resultList) 6 result = zip(numList, strList) 7 resultSet = set(result) 8 print(resultSet)</pre>	<pre>[] {(4, 'four'), (6, 'six'), (5, 'five')}  === Code Execution Successful ===</pre>

# Python Project

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

main.py	Output
<pre>1 def reciprocal(num): 2     return 1 / num 3 lambda_reciprocal = lambda num: 1 / num 4 print("Def keyword: ", reciprocal(6)) 5 print("Lambda keyword: ", lambda_reciprocal(6))</pre>	<pre>Def keyword:  0.16666666666666666 Lambda keyword:  0.16666666666666666  === Code Execution Successful ===</pre>

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

main.py	Output
<pre>1 squares = [lambda num = num: num ** 2 for num in range(0, 11)] 2 for square in squares: 3     print('The square value of all numbers from 0 to 10:', square 4         (), end=" ")</pre>	<pre>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  === Code Execution Successful ===</pre>

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

main.py	Output
<pre>1 my_List = [[3, 5, 8, 6], [23, 54, 12, 87], [1, 2, 4, 12, 5]] 2 sort_List = lambda num: (sorted(n) for n in num) 3 third_Largest = lambda num, func: [l[len(l) - 2] for l in func(num 4     )] 5 result = third_Largest(my_List, sort_List) 6 print('The third largest number from every sub list is:', result)</pre>	<pre>The third largest number from every sub list is: [6, 54, 5]  === Code Execution Successful ===</pre>

## Python Modules

The screenshot shows a Python IDE with two tabs: 'example\_module.py' and 'main\_program.py'. The 'main\_program.py' tab is active, showing the following code:

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

Below the code editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'OUTPUT' tab is selected, showing the following output:

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

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

# Python Project

## Importing and also Renaming

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

The value of euler's number is 2.718281828459045

## Python from...import Statement

```
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

## Import all Names - From import \* Statement

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

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

## Locating Path of Modules

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

Path of the sys module in the system is: ['/home', '/usr/lib/python3.12.zip', '/usr/lib/python3.12', '/usr/lib/python3.12/lib-dynload', '/usr/local/lib/python3.12/dist-packages', '/usr/lib/python3/dist-packages']

## The dir() Built-in Function

```
1 print( "List of functions:\n ", dir( str ), end="," )
2
```

List of functions:  
['\_add', '\_class', '\_contains', '\_delattr', '\_dir', '\_doc', '\_eq', '\_format', '\_ge', '\_getattr', '\_getitem', '\_getnewargs', '\_getstate', '\_gt', '\_hash', '\_init', '\_init\_subclass', '\_iter', '\_le', '\_len', '\_lt', '\_mod', '\_mul', '\_ne', '\_new', '\_red', '\_reduce\_ex', '\_repr', '\_rmod', '\_rmul', '\_setattr', '\_sizeof', '\_str', '\_subclasshook', '\_capitalize', '\_casefold', '\_center', '\_count', '\_encode', '\_endswith', '\_expandtabs', '\_find', '\_format', '\_format\_map', '\_index', '\_isalnum', '\_isalpha', '\_isascii', '\_isdecimal', '\_isdigit', '\_isidentifier', '\_islower', '\_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']



# Python Project

## Namespaces and Scoping

```
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)
10
```

The number is: 204  
The number is: 404

## PYTHON ARRAYS

### 1. Accessing array elements

```
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

### Deleting the elements from Array

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

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

# Python Project

## 1. Adding or changing the elements in Array

```
File (Ctrl+M)
main.py
1 import array as arr
2 numbers = arr.array('i', [1, 2, 3, 5, 7, 10])
3 numbers[0] = 0
4 print(numbers)
5 numbers[5] = 8
6 print(numbers)
7 numbers[2:5] = arr.array('i', [4, 6, 8])
8 print(numbers)

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

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

To find the length of array

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

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

# Python Project

## Python Decorator

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

input

Hii, welcome to function  
Hii, welcome to function

## Inner Function

```
main.py
1 def func(): # here, we are creating a function and passing the parameter
2     print("We are in first function") # Here, we are printing the data of function
3     def func1(): # here, we are creating a function and passing the parameter
4         print("This is first child function") # Here, we are printing the data of function 1
5     def func2(): # here, we are creating a function and passing the parameter
6         print("This is second child function") # Here, we are printing the data of
7     func1()
8     func2()
9 func()
```

input

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

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

9  
21

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

Hello

# Python Project

## Decorating functions with parameters

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

Hello

## Syntactic Decorator

```
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
9 @outer_div
10 def divide(x, y):
11     print(x / y)
12 divide(5, 10)
13
```

2.0

## Reusing Decorator

```
Welcome  mod_decorator.py U  do_twice.py U X
123 > do_twice.py > ...
1 from mod_decorator import do_twice
2 @do_twice
3 def say_hello():
4     print("Hello There")
5 say_hello()
```

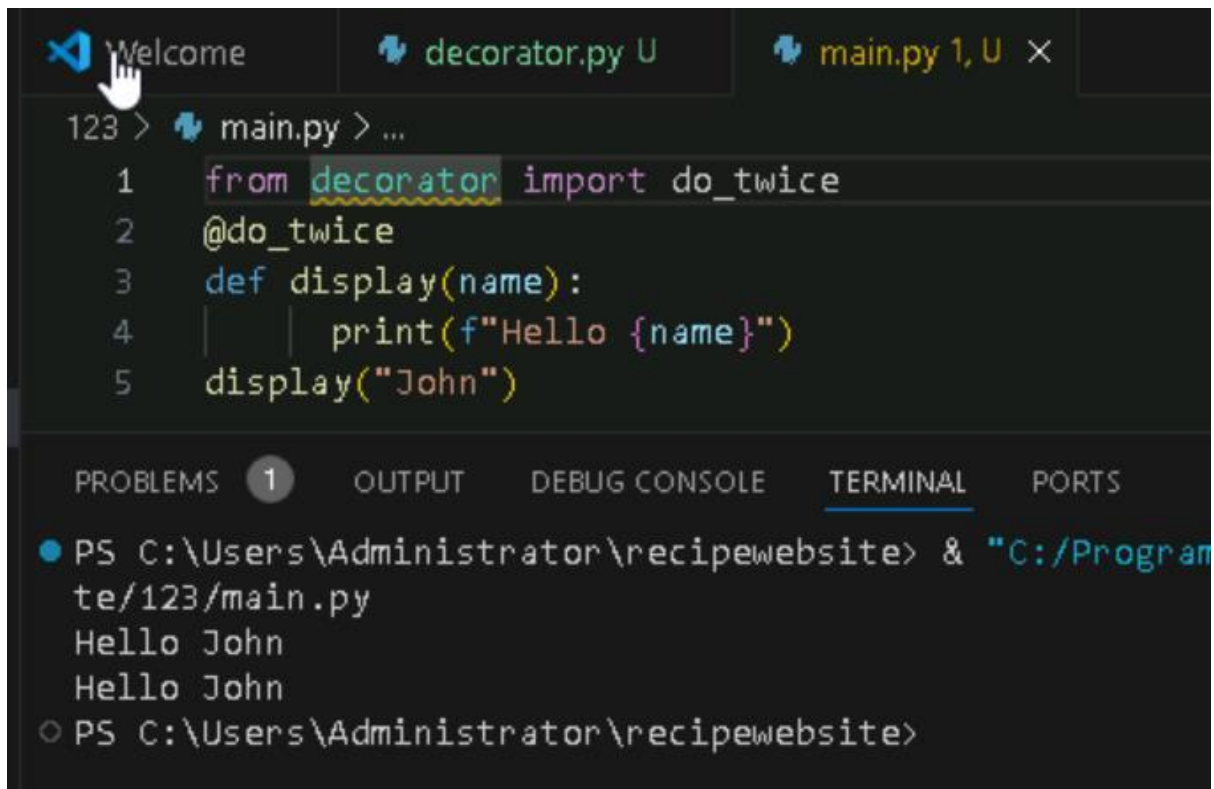
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Administrator\recipewebsite> & "C:/Program Files/Python37/Python.exe" C:/Users/Administrator/recipewebsite/123/do_twice.py
Hello There
Hello There
PS C:\Users\Administrator\recipewebsite>
```



## Python Project

### Python Decorator with Argument



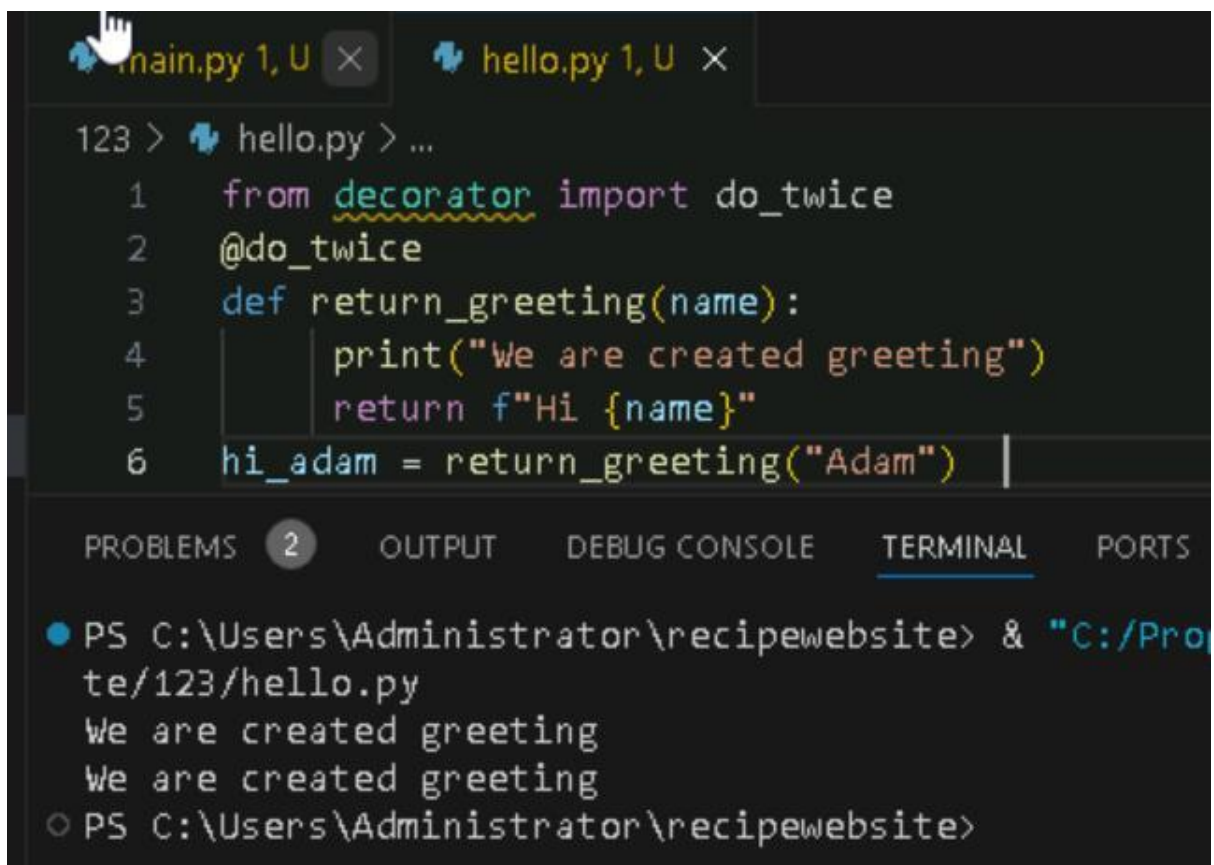
The screenshot shows the Visual Studio Code interface. The top bar has tabs for 'Welcome', 'decorator.py U', and 'main.py 1, U X'. The 'main.py' tab is active. The editor displays the following code:

```
123 > main.py > ...
1  from decorator import do_twice
2  @do_twice
3  def display(name):
4      print(f"Hello {name}")
5  display("John")
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS C:\Users\Administrator\recipewebsite> & "C:/Program Files/Python311/Python.exe" C:/Program Files/Python311/Scripts/python C:/Users/Administrator/recipewebsite/123/main.py
Hello John
Hello John
PS C:\Users\Administrator\recipewebsite>
```

### Returning Values from Decorated Functions



The screenshot shows the Visual Studio Code interface. The top bar has tabs for 'main.py 1, U X' and 'hello.py 1, U X'. The 'hello.py' tab is active. The editor displays the following code:

```
123 > hello.py > ...
1  from decorator import do_twice
2  @do_twice
3  def return_greeting(name):
4      print("We are created greeting")
5      return f"Hi {name}"
6  hi_adam = return_greeting("Adam")
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS C:\Users\Administrator\recipewebsite> & "C:/Program Files/Python311/Python.exe" C:/Program Files/Python311/Scripts/python C:/Users/Administrator/recipewebsite/123/hello.py
We are created greeting
We are created greeting
PS C:\Users\Administrator\recipewebsite>
```

# Python Project

## Fancy Decorators

```
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
```

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

```
1 class Person:      # here, we are creating a class with the name Student
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()
```

Hello Peter  
Hello Peter

## Decorator with Arguments

```
1 import functools # Importing functools into the program
2
3 def repeat(num): # Defining the repeat function that takes 'num'
4     # Creating and returning the decorator function
5     def decorator_repeat(func):
6         @functools.wraps(func) # Using functools.wraps to pre
7         def wrapper(*args, **kwargs):
8             for _ in range(num): # Looping 'num' times to rep
9                 value = func(*args, **kwargs) # Calling the d
10            return value # Returning the value after the loop
11        return wrapper # Returning the wrapper function
12
13    return decorator_repeat
14
15 @repeat(num=5)
16 def function1(name):
17     print(f"{name}")
18
19 function1("John")
20
```

John  
John  
John  
John  
John

# Python Project

## Stateful Decorators

```
1 import functools # Importing functools into the program
2
3 def count_function(func):
4     # Defining the decorator function that counts the number of calls
5     @functools.wraps(func) # Preserving the metadata of the original function
6     def wrapper_count_calls(*args, **kwargs):
7         wrapper_count_calls.num_calls += 1 # Increment the call count
8         print(f"Call {wrapper_count_calls.num_calls} of {func.__name__!r}")
9         return func(*args, **kwargs) # Call the original function with the argument
10
11     wrapper_count_calls.num_calls = 0 # Initialize the call counter
12     return wrapper_count_calls # Return the wrapper function
13
14 # Applying the decorator to the function say_hello
15 @count_function
16 def say_hello():
17     print("Say Hello")
18
19 # Calling the decorated function twice
20 say_hello() # First call
21 say_hello() # Second call
22
```

input

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

## Classes as Decorators

```
1 import functools # Importing functools into the program
2
3 class Count_Calls:
4     # Class to count the number of times a function is called
5     def __init__(self, func):
6         functools.update_wrapper(self, func) # To update the wrapper with the original
7         self.func = func # Store the original function
8         self.num_calls = 0 # Initialize call counter
9
10     def __call__(self, *args, **kwargs):
11         # Increment the call counter each time the function is called
12         self.num_calls += 1
13         print(f"Call {self.num_calls} of {self.func.__name__!r}")
14         return self.func(*args, **kwargs) # Call the original function
15
16 # Applying the Count_Calls class as a decorator
17 @Count_Calls
18 def say_hello():
19     print("Say Hello")
20
21 # Calling the decorated function multiple times
22 say_hello() # First call
23 say_hello() # Second call
24 say_hello() # Third call
25
```

input

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

# Python Project

## How to Create Generator function in Python

main.py	Output
<pre>1 def simple(): 2     for i in range(10): 3         if(i%2==0): 4             yield i 5 for i in simple(): 6     print(i)</pre>	<pre>0 2 4 6 8  === Code Execution Successful ===</pre>

## Using multiple yield Statement

main.py	Output
<pre>1 def multiple_yield(): 2     str1 = "First String" 3     yield str1 4     str2 = "Second string" 5     yield str2 6     str3 = "Third String" 7     yield str3 8 obj = multiple_yield() 9 print(next(obj)) 10 print(next(obj)) 11 print(next(obj))</pre>	<pre>First String Second string Third String  === Code Execution Successful ===</pre>

## Generator expression

main.py	Output
<pre>1 list = [1, 2, 3, 4, 5, 6, 7] 2 z = [x**3 for x in list] 3 a = (x**3 for x in list) 4 print(a) 5 print(z)</pre>	<pre>&lt;generator object &lt;genexpr&gt; at 0x7f34852edd80&gt; [1, 8, 27, 64, 125, 216, 343]  === Code Execution Successful ===</pre>

## Next function

main.py	Output
<pre>1 list = [1, 2, 3, 4, 5, 6] 2 z = (x**3 for x in list) 3 print(next(z)) 4 print(next(z)) 5 print(next(z)) 6 print(next(z))</pre>	<pre>1 8 27 64  === Code Execution Successful ===</pre>



# Python Project

program to print the table of the given number using the generator

main.py	Output
<pre>1- def table(n): 2-     for i in range(1, 11): 3-         yield n * i 4-         i = i + 1 5- for i in table(15): 6-     print(i)</pre>	15 30 45 60 75 90 105 120 135 150  === Code Execution Successful ===

sys.getsizeof() function

```
main.py
1 import sys
2 nums_squared_list = [i * 2 for i in range(1000)]
3 print(sys.getsizeof("Memory in Bytes:", nums_squared_list))
4 nums_squared_gc = (i ** 2 for i in range(1000))
5 print(sys.getsizeof("Memory in Bytes:", nums_squared_gc))
```

input

57  
57

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

Generate Infinite Sequence

main.py	Output
<pre>1- def infinite_sequence(): 2-     num = 0 3-     while True: 4-         yield num 5-         num += 1 6- for i in infinite_sequence(): 7-     print(i)</pre>	713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728  === Execution Halted ===