



201 PYTHON PROGRAMMING EXERCISES FOR ALL



EDCORNER LEARNING

201 Python Programming Exercises For All

201 Python Programming Exercises For All

Edcorner Learning

Table of Contents

[Introduction](#)

[Module 1 Tuples](#)

[Module 2 Dictionaries](#)

[Module 3 Data type](#)

[Module 4 Conditionals](#)

[Module 5 Loops](#)

[Module 6 Exceptions](#)

[Module 7 Functions](#)

[Module 8 Regular Expressions](#)

[Module 9 Classes](#)

[Module 10 Strings](#)

[Module 11 Dictionary](#)

[Module 12 Lists](#)

Introduction

Python is a general-purpose interpreted, interactive, object-oriented, and a powerful programming language with dynamic semantics. It is an easy language to learn and become expert. Python is one among those rare languages that would claim to be both easy and powerful. Python's elegant syntax and dynamic typing alongside its interpreted nature makes it an ideal language for scripting and robust application development in many areas on giant platforms.

Python helps with the modules and packages, which inspires program modularity and code reuse. The Python interpreter and thus the extensive standard library are all available in source or binary form for free of charge for all critical platforms and can be freely distributed. Learning Python doesn't require any pre-requisites. However, one should have the elemental understanding of programming languages.

This Book consist of 12 modules to practice different Python exercises on different topics.

In each exercise we have given the exercise coding statement you need to complete and verify your answers. We also attached our own input output screen of each exercise and their solutions.

Learners can use their own python compiler in their system or can use any online compilers available.

We have covered all level of exercises in this book to give all the learners a good and efficient Learning method to do hands on python different scenarios.

Module 1 Tuples

1. Given the code below, use the correct argument(s) for the **range()** function on line 1 in order to return a range of consecutive integers from 0 to 9 inclusively. Use a single argument inside the parentheses of **range()**!
- 2.

```
my_range =
```

```
print(list(my_range)) #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
#Edcorner Learning Python Exercises Vol  
2
```

```
my_range = range(10)
```

```
print(list(my_range)) #[0, 1, 2, 3, 4,  
5, 6, 7, 8, 9]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```


2. Given the code below, use the correct argument(s) for the **range()** function on line 1 in order to return a range of consecutive integers from 0 to 9 inclusively. Use two arguments inside the parentheses of **range()**!

```
my_range =
```

```
print(list(my_range)) #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
my_range = range(0, 10)
```

```
print(list(my_range)) #[0, 1, 2, 3, 4,  
5, 6, 7, 8, 9]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

3. Given the code below, use the correct argument(s) for the `range()` function on line 1 in order to return a range of consecutive integers from 117 to 129 exclusively.

```
my_range =
```

```
print(list(my_range)) #[117, 118, 119, 120, 121, 122, 123, 124, 125,  
126, 127, 128]
```

```
my_range = range(117, 129)
```

```
print(list(my_range)) #[117, 118, 119,  
120, 121, 122, 123, 124, 125, 126, 127,  
128]
```

```
[117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128]
```

4. Given the code below, use the correct argument(s) for the range() function on line 1 in order to return [10,13,16,19] when converted to a list.

```
my_range =
```

```
print(list(my_range)) #[10, 13, 16, 19]
```

#Edcorner Learning Python Exercises

```
my_range = range(10, 20, 3)
```

```
print(list(my_range)) #[10, 13, 16, 19]
```

[10, 13, 16, 19]

5. Given the code below, add the correct step as the third argument of the range() function on line 1 in order to return [115,120] when converted to a list.

```
my_range = range(115, 125, )
```

```
print(list(my_range)) #[115, 120]
```

```
#Edcorner Learning Python Exercises
```

```
my_range = range(115, 125, 5)
```

```
print(list(my_range)) #[115, 120]
```

1.

```
[115, 120]
```


6. Given the code below, add the correct step as the third argument of the **range()** function on line 1 in order to return [-75, -60, -45, -30] when converted to a list.

```
my_range = range(-75, -25, )
```

```
print(list(my_range)) #[-75, -60, -45, -30]
```

```
#Edcorner Learning Python Exercises
```

```
my_range = range(-75, -25, 15)
```

```
print(list(my_range)) #[-75, -60, -45,  
-30]
```

```
[-75, -60, -45, -30]
```

7. Given the code below, add the correct step as the third argument of the `range()` function on line 1 in order to return `[-25, 5, 35, 65, 95, 125]` when converted to a list.

```
my_range = range(-25, 139, )
```

```
print(list(my_range)) #[-25, 5, 35, 65, 95, 125]
```

#Edcorner Learning Python Exercises

```
my_range = range(-25, 139, 30)
```

```
print(list(my_range)) #[-25, 5, 35, 65,  
95, 125]
```

```
[-25, 5, 35, 65, 95, 125]
```

8. Given the code below, write the correct range on line 1 in order to return [-10] when converted to a list.

```
my_range =
```

```
print(list(my_range)) #[-10]
```

#Edcorner Learning Python Exercises

```
my_range = range(-10,-9)  
print(list(my_range)) #[-10]
```

```
[-10]
```

Module 2 Dictionaries

9. Given the code below, use the correct code on line 3 in order to return the value associated with key 4. Do not use a method as a solution for this exercise!

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar",  
5: "XRP"}  
  
value =  
    print(value)
```



```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
value = crypto[4]
```

```
print(value)
```

Stellar

10. Given the code below, use the correct code on line 3 in order to return the value associated with key 4. This time, use a method as a solution for this exercise!

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
value =
```

```
print(value)
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}  
  
value = crypto.get(4)  
  
print(value)
```

Stellar

11. Given the code below, use the correct code on line 3 in order to update the value associated with key 4 to "Cardano".

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4:  
"Stellar", 5: "XRP"}
```

```
print(crypto[4])
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
crypto[4] = "Cardano"
```

```
print(crypto[4])
```

Cardano

12. Given the code below, use the correct code on line 3 in order to add a new key-value pair to the dictionary: 6: "Monero"

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
print(crypto[6])
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
crypto[6] = "Monero"
```

```
print(crypto[6])
```

Monero

13. Given the code below, use the correct code on line 3 to return the number of key-value pairs in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar",  
5: "XRP"}
```

```
number =
```

```
print(number)
```


#Edcorner Learning Python Exercises

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}  
  
number = len(crypto)  
  
print(number)
```

5

14. Given the code below, use the correct code on line 3 to delete the key-value pair associated with key 3. Do not use a method as a solution for this exercise!

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
print(crypto)
```

#Edcorner Learning Python Exercises

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
del crypto[3]
```

```
print(crypto)
```

```
{1: 'Bitcoin', 2: 'Ethereum', 4: 'Stellar', 5: 'XRP'}
```

15. Given the code below, use the correct code on line 3 in order to delete the key-value pair associated with key 3. This time, use a method as a solution for this exercise!

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar",  
5: "XRP"}  
  
print(crypto)
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
crypto.pop(3)
```

```
print(crypto)
```

```
{1: 'Bitcoin', 2: 'Ethereum', 4: 'Stellar', 5: 'XRP'}
```

16. Given the code below, use the correct code on line 3 in order to verify that 7 is not a key in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
check =
```

```
print(check)
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
check = 7 not in crypto
```

```
print(check)
```

True

`17. Given the code below, use the correct method on line 3 in order to delete all the elements in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar",  
5: "XRP"}
```

```
crypto.
```

```
print(crypto)
```


#Edcorner: Learning Python Exercises

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
crypto.clear()
```

```
print(crypto)
```

```
{}
```

18. Given the code below, use the correct code on line 3 in order to get a list of tuples, where each tuple represents a key-value pair in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
result =
```

```
print(list(result))
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}  
  
result = crypto.items()  
  
print(list(result))
```

```
[(1, 'Bitcoin'), (2, 'Ethereum'), (3, 'Litecoin'), (4, 'Stellar'), (5, 'XRP')]
```

19. Given the code below, use the correct function on line 3 in order to get the sum of all the keys in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar",  
5: "XRP"}
```

```
add =
```

```
print(add)
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}  
  
add = sum(crypto)  
  
print(add)
```

20. Given the code below, use the correct method on line 3 in order to get a list of all the values in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
val =
```

```
print(list(val))
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
val = crypto.values()
```

```
print(list(val))
```

```
['Bitcoin', 'Ethereum', 'Litecoin', 'Stellar', 'XRP']
```

21. Given the code below, use the correct function on line 3 in order to get the smallest key in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar",  
5: "XRP"}
```

```
key =  
    print(key)
```



```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}
```

```
key = min(crypto)
```

```
print(key)
```

22. Given the code below, use the correct method on line 3 in order to get a list of all the keys in the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4: "Stellar", 5:  
"XRP"}
```

```
keys =
```

```
print(list(keys))
```

#Edcorner Learning Python Exercises

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}  
  
keys = crypto.keys()  
  
print(list(keys))
```

```
[1, 2, 3, 4, 5]
```

23. Given the code below, use the correct method on line 3 in order to return and remove an arbitrary key-value pair from the dictionary.

```
crypto = {1: "Bitcoin", 2: "Ethereum", 3: "Litecoin", 4:  
"Stellar", 5: "XRP"}  
print(len(crypto))
```

```
crypto = {1: "Bitcoin", 2: "Ethereum",  
3: "Litecoin", 4: "Stellar", 5: "XRP"}  
  
crypto.popitem()  
  
print(len(crypto))
```

Module 3 Data type

24. Given the code below, use the correct function on line 3 in order to convert value to a string.

```
value = 10
```

```
conv =
```

```
print(type(conv))
```

#Edcorner Learning Python Exercises

```
value = 10
```

```
conv = str(value)
```

```
print(type(conv))
```

```
<class 'str'>
```


25. Given the code below, use the correct function on line 3 in order to convert value to an integer.

```
value = "10"
```

```
conv =
```

```
print(type(conv))
```

#Edcorner Learning Python Exercises

```
value = "10"  
conv = int(value)  
print(type(conv))
```

```
<class 'int'>
```

26. Given the code below, use the correct function on line 3 in order to convert value to a floating-point number.

```
value = 10
```

```
conv =
```

```
print(type(conv))
```

#Edcorner Learning Python Exercises

```
value = 10
```

```
conv = float(value)
```

```
print(type(conv))
```

```
<class 'float'>
```

27. Given the code below, use the correct function on line 3 in order to convert value to a list.

```
value = "Hello!"  
conv =  
print(type(conv))
```

```
#Edcorner Learning Python Exercises
```

```
value = "Hello!"
```

```
conv = list(value)
```

```
print(type(conv))
```

```
<class 'list'>
```

28. Given the code below, use the correct function on line 3 in order to convert value to a tuple.

```
value = [1, 2, 3, 10, 20, 30]
```

```
conv =
```

```
print(type(conv))
```

#Edcorner Learning Python Exercises

```
value = [1, 2, 3, 10, 20, 30]
```

```
conv = tuple(value)
```

```
print(type(conv))
```

```
<class 'tuple'>
```


29. Given the code below, use the correct function on line 3 in order to convert value to a frozen set.

```
value = (10, 20, 40, 10, 25, 30, 45)
```

```
conv =
```

```
print(type(conv))
```

#Edcorner Learning Python Exercises

```
value = (10, 20, 40, 10, 25, 30, 45)
```

```
conv = frozenset(value)
```

```
print(type(conv))
```

```
<class 'frozenset'>
```

30. Given the code below, use the correct function on line 3 in order to convert value to a binary representation.

```
value = 10
```

```
conv =
```

```
print(conv)
```

#Edcorner Learning Python Exercises

```
value = 10
```

```
conv = bin(value)
```

```
print(conv)
```

0b1010

31. Given the code below, use the correct function on line 3 in order to convert value to a hexadecimal representation.

```
value = 10
```

```
conv =
```

```
print(conv)
```

#Edcorner Learning Python Exercises

```
value = 10  
conv = hex(10)  
print(conv)
```

0xa

32. Given the code below, use the correct function on line 3 in order to convert value from binary to decimal notation.

```
value = '0b1010'
```

```
conv =
```

```
print(conv)
```

```
value = '0b1010'
```

```
conv = int(value, 2)
```

```
print(conv)
```


33. Given the code below, use the correct function on line 3 in order to convert value from hexadecimal to decimal notation.

```
value = '0xa'
```

```
conv =
```

```
print(conv)
```

```
value = '0xa'
```

```
conv = int(value, 16)
```

```
print(conv)
```

Module 4 Conditionals

34. Considering the code below, write code that prints out True! if x has 50 characters or more.

```
x = "The days of Python 2 are almost over. Python 3 is the king now."
```

#Edcorner Learning Python Exercises

```
x = "The days of Python 2 are almost  
over. Python 3 is the king now."
```

```
if len(x) >= 50:  
    print("True!")
```

True!

35. Considering the code below, write code that prints out True! if x is a string and the first character in the string is T.

```
x = "The days of Python 2 are almost over. Python 3 is the  
king now."
```

#Edcorner Learning Python Exercises

```
x = "The days of Python 2 are almost  
over. Python 3 is the king now."
```

```
if type(x) is str and  
x.startswith("T"):  
    print("True!")
```

True!

36. Considering the code below, write code that prints out True! if at least one of the following conditions occurs:

- the string contains the character z
- the string contains the character y at least twice

```
x = "The days of Python 2 are almost over. Python 3 is the king now."
```


#Edcorner Learning Python Exercises

```
x = "The days of Python 2 are almost  
over. Python 3 is the king now."
```

```
if "z" in x or x.count("y") >= 2:  
    print("True!")
```

True!

37. Considering the code below, write code that prints out True! if the index of the first occurrence of letter f is less than 10 and prints out False! if the same condition is not satisfied.

```
x = "The days of Python 2 are almost over. Python 3 is the  
king now."
```

#Edcorner: Learning Python Exercises

x = "The days of Python 2 are almost over. Python 3 is the king now."

```
if x.index("f") < 10:  
    print("True!")  
else:  
    print("False!")
```

False!

38. Considering the code below, write code that prints out True! if the last 3 characters of the string are all digits and prints out False! otherwise.

Hint! Use the appropriate method to check if the requested string slice contains digits only.

```
x = "The days of Python 2 are almost over. Python 3 is the king now."
```

#Edcorner Learning Python Exercises

```
x = "The days of Python 2 are almost  
over. Python 3 is the king now."
```

```
if x[-3:].isdigit():  
    print("True!")  
else:  
    print("False!")
```

False!

39. Considering the code below, write code that prints out True! if x has at least 8 elements and the element positioned at index 6 is a floating-point number and prints out False! otherwise.

```
x = [115, 115.9, 116.01, ["length", "width", "height"], 109,  
115, 119.5, ["length", "width", "height"]]
```

#Edcorner Learning Python Exercises

```
x = [115, 115.9, 116.01, ["length",  
"width", "height"], 109, 115, 119.5,  
["length", "width", "height"]]  
  
if len(x) >= 8 and type(x[6]) is float:  
    print("True!")  
else:  
    print("False!")
```

True!

40. Considering the code below, write code that prints out True! if the second string of the first list in X ends with the letter h and the first string of the second list in x also ends with the letter h, and prints out False! otherwise.

```
x = [115, 115.9, 116.01, ["length", "width", "height"], 109, 115, 119.5, ["length", "width", "height"]]
```


#Edcorner Learning Python Exercises

```
x = [115, 115.9, 116.01, ["length",  
"width", "height"], 109, 115, 119.5,  
["length", "width", "height"]]
```

```
if x[3][1].endswith("h") and x[7]  
[0].endswith("h"):  
    print("True!")  
else:  
    print("False!")
```

True!

41. Considering the code below, write code that prints out True! if one of the following two conditions are satisfied and prints out False! otherwise.

- the third string of the first list in x ends with the letter h
- the second string of the second list in x also ends with the letter h

```
x = [115, 115.9, 116.01, ["length", "width", "height"], 109, 115,  
119.5, ["length", "width", "height"]]
```

```
x = [115, 115.9, 116.01, ["length",  
"width", "height"], 109, 115, 119.5,  
["length", "width", "height"]]
```

```
if x[3][2].endswith("h") or x[7]  
[1].endswith("h"):  
    print("True!")  
else:  
    print("False!")
```

True!

42. Considering the code below, write code that prints out **True!** if the largest value among the first 3 elements of the list is less than or equal to the smallest value among the next 3 elements of the list. Otherwise, print out **False!**

Hint! Use the appropriate slices to make the comparison.

```
x = [115, 115.9, 116.01, 109, 115, 119.5, ["length", "width",  
"height"]]
```

#Edcorner Learning Python Exercises

```
x = [115, 115.9, 116.01, 109, 115,  
119.5, ["length", "width", "height"]]
```

```
if max(x[:3]) <= min(x[3:6]):  
    print("True!")  
else:  
    print("False!")
```

False!

43. Considering the code below, write code that prints out **True!** if **115** appears at least once inside the list or if it is the first element in the list. Otherwise, print out **False!**

Hint! Use the appropriate method to check if 115 is the first element in the list.

```
x = [115, 115.9, 116.01, 109, 115, 119.5, ["length", "width",  
"height"]]
```

#Edcorner Learning Python Exercises

```
x = [115, 115.9, 116.01, 109, 115,
119.5, ["length", "width", "height"]]

if x.count(115) >= 1 or x.index(115) ==
0:
    print("True!")
else:
    print("False!")
```

True!

44. Considering the code below, write code that prints out **True!** if the value associated with key number **5** is **Perl** or the number of key-value pairs in the dictionary divided by 5 returns a remainder less than 2. Otherwise, print out **False!**

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```



```
#Edcorner Learning Python Exercises
```

```
x = {1: "Python", 2: "Java", 3:  
"Javascript", 4: "Ruby", 5: "Perl", 6:  
"C#", 7: "C++"}
```

```
if x[5] == "Perl" or len(x) % 5 < 2:  
    print("True!")  
else:  
    print("False!")
```

```
True!
```

45. Considering the code below, write code that prints out True! if 3 is a key in the dictionary and the smallest value (alphabetically) in the dictionary is C#. Otherwise, print out False!

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl",  
6: "C#", 7: "C + +"}
```

```
#Edcorner Learning Python Exercises
```

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```

```
if 3 in x and sorted(x.values())[0] == "C#":  
    print("True!")  
else:  
    print("False!")
```

```
True!
```

46. Considering the code below, write code that prints out True! if the last character of the largest (alphabetically) value in the dictionary is n. Otherwise, print out False!

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```

#Edcorner Learning Python Exercises

```
x = {1: "Python", 2: "Java", 3:
"Javascript", 4: "Ruby", 5: "Perl", 6:
"C#", 7: "C++"}

if sorted(x.values())[-1][-1] == "n":
    print("True!")
else:
    print("False!")
```

False!

47. Considering the code below, write code that prints out **True!** if the largest key in the dictionary divided by the second largest key in the dictionary returns a remainder equal to the smallest key in the dictionary. Otherwise, print out **False!**

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl",  
6: "C#", 7: "C++"}
```

#Edcorner Learning Python Exercises

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```

```
if sorted(x.keys())[-1] %  
sorted(x.keys())[-2] ==  
sorted(x.keys())[0]:  
    print("True!")  
else:  
    print("False!")
```

True!

48. Considering the code below, write code that prints out **True!** if the sum of all the keys in the dictionary is less than the number of characters of the string obtained by concatenating the values associated with the first 5 keys in the dictionary. Otherwise, print out **False!**

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C + +"}
```


#Edcorner Learning Python Exercises

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```

```
if sum(x) < len(x[1] + x[2] + x[3] + x[4] + x[5]):  
    print("True!")  
else:  
    print("False!")
```

False!

49. Considering the code below, write code that prints out True! if the 3rd element of the first range is less than 2, prints out False! if the 5th element of the first range is 5, and prints out None! otherwise.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```

#Edcorner Learning Python Exercises

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]
```

```
if x[0][2] < 2:  
    print("True!")  
elif x[0][4] == 5:  
    print("False!")  
else:  
    print("None!")
```

None!

50. Considering the code below, write code that prints out **True!** if the 3rd element of the 3rd range is less than 6, prints out **False!** if the 1 st element of the second range is 5, and prints out **None!** otherwise.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```

#Edcorner Learning Python Exercises

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]
```

```
if x[2][2] < 6:  
    print("True!")  
elif x[1][0] == 5:  
    print("False!")  
else:  
    print("None!")
```

False!

51. Considering the code below, write code that prints out **True!** if the last element of the first range is greater than 3, prints out **False!** if the last element of the second range is less than 9, and prints out **None!** otherwise.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```

#Edcorner Learning Python Exercises

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]
```

```
if x[0][-1] > 3:  
    print("True!")  
elif x[1][-1] < 9:  
    print("False!")  
else:  
    print("None!")
```

True!

52. Considering the code below, write code that prints out **True!** if the length of the first range is greater than or equal to 5, prints out **False!** if the length of the second range is 4, and prints out **None!** otherwise.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```


#Edcorner Learning Python Exercises

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]
```

```
if len(x[0]) >= 5:  
    print("True!")  
elif len(x[1]) == 4:  
    print("False!")  
else:  
    print("None!")
```

True!

53. Considering the code below, write code that prints out True! if the sum of all the elements of the first range is greater than the sum of all the elements of the third range, prints out False! if the largest element of the second range is greater than the largest element of the third range, and prints out None! otherwise.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```

#Edcorner Learning Python Exercises

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]
```

```
if sum(x[0]) > sum(x[2]):  
    print("True!")  
elif max(x[1]) > max(x[2]):  
    print("False!")  
else:  
    print("None!")|
```

False!

54. Considering the code below, write code that prints out True! if the largest element of the first range minus the second element of the 3rd range is equal to the first element of the first range, prints out False! if the length of the first range minus the length of the 2nd range is equal to the first element of the 3rd range, prints out Maybe! if the sum of all the elements of the 3rd range divided by 2 returns a remainder of 0, and prints out None! otherwise.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```

#Edcorner Learning Python Exercises

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]  
  
if max(x[0]) - x[2][1] == x[0][0]:  
    print("True!")  
elif len(x[0]) - len(x[1]) == x[2][0]:  
    print("False!")  
elif sum(x[2]) % 2 == 0:  
    print("Maybe!")  
else:  
    print("None!")
```

True!

55. Considering the code below, write code that prints out True! if the sum of the last 3 elements of the first range plus the sum of the last 3 elements of the 3rd range is equal to the sum of the last 3 elements of the 2nd range, and prints out False! if the length of the first range times 2 is less than the sum of all the elements of the 3rd range.

```
x = [list(range(5)), list(range(5,9)), list(range(1,10,3))]
```

```
x = [list(range(5)), list(range(5,9)),  
list(range(1,10,3))]
```

```
if sum(x[0][-3:]) + sum(x[2][-3:]) ==  
sum(x[1][-3:]):  
    print("True!")  
elif len(x[0]) * 2 < sum(x[2]):  
    print("False!")
```

True!

56. Considering the code below, write code that prints out True! if the 2nd character of the value at key 1 is also present in the value at key 4, and prints out False! otherwise.

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```


#Edcorner Learning Python Exercises

```
x = {1: "Python", 2: "Java", 3:
"Javascript", 4: "Ruby", 5: "Perl", 6:
"C#", 7: "C++"}
```

```
if x[1][1] in x[4]:
    print("True!")
else:
    print("False!")
```

True!

57. Considering the code below, write code that prints out True! if the second to last character of the value at key 3 is the first character of the value at key 5, and prints out False! otherwise.

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl",  
6: "C#", 7: "C++"}
```

#Edcorner Learning Python Exercises

```
x = {1: "Python", 2: "Java", 3:
"Javascript", 4: "Ruby", 5: "Perl", 6:
"C#", 7: "C++"}

if x[3][-2] == x[5][0]:
    print("True!")
else:
    print("False!")
```

False!

58. Considering the code below, write code that prints out True! if the number of characters of the smallest value in the dictionary is equal to the number of occurrences of letter a in the value at key 3, and prints out False! otherwise.

```
x = {1: "Python", 2: "Java", 3: "Javascript", 4: "Ruby", 5: "Perl", 6: "C#", 7: "C++"}
```

#Edcorner Learning Python Exercises

```
x = {1: "Python", 2: "Java", 3:
"Javascript", 4: "Ruby", 5: "Perl", 6:
"C#", 7: "C++"}
```

```
if len(min(x.values())) ==
x[3].count("a"):
    print("True!")
else:
    print("False!")
```

True!

Module 5 Loops

59. Write a for loop that iterates over the x list and prints out all the elements of the list.

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

#Edcorner Learning Python Exercises

```
x = [10, 12, 13, 14, 17, 19, 21, 22,  
25]
```

```
for i in x:  
    print(i)
```

```
10  
12  
13  
14  
17  
19  
21  
22  
25
```


60. Write a for loop that iterates over the x list and prints out the remainders of each element of the list divided by 3.

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

#Edcorner Learning Python Exercises

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

```
for i in x:  
    print(i % 3)
```

```
1  
0  
1  
2  
2  
1  
0  
1  
1
```

61. Write a for loop that iterates over the x list and prints out all the elements of the list in reversed order and multiplied by 10.

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

#Edcorner Learning Python Exercises

```
x = [10, 12, 13, 14, 17, 19, 21, 22,  
25]
```

```
for i in sorted(x, reverse = True):  
    print(i * 10)
```

```
250  
220  
210  
190  
170  
140  
130  
120  
100
```

62. Write a for loop that iterates over the x list and prints out all the elements of the list divided by 2 and the string Great job! after the list is exhausted.

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

#Edcorner Learning Python Exercises

```
x = [10, 12, 13, 14, 17, 19, 21, 22,  
25]
```

```
for i in x:  
    print(i / 2)  
else:  
    print("Great job!")
```

```
5.0  
6.0  
6.5  
7.0  
8.5  
9.5  
10.5  
11.0  
12.5  
Great job!
```

63. Write a for loop that iterates over the x list and prints out the index of each element.

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

```
#Edcorner Learning Python Exercises
```

```
x = [10, 12, 13, 14, 17, 19, 21, 22,  
25]
```

```
for i in x:  
    print(x.index(i))
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8
```


64. Write a while loop that prints out the value of x squared while x is less than or equal to 5. Be careful not to end up with an infinite loop!

`x = 0`

#Edcorner Learning Python Exercises

```
x = 0
```

```
while x <= 5:  
    print(x ** 2)  
    x = x + 1
```

```
0  
1  
4  
9  
16  
25
```

65. Write a while loop that prints out the value of x times 10 while x is less than or equal to 4 and then prints out Done! when x becomes larger than 4. Be careful not to end up with an infinite loop!

`x = 0`

#Edcorner Learning Python Exercises

```
x = 0  
  
while x <= 4:  
    print(x * 10)  
    x = x + 1  
else:  
    print("Done!")
```

```
0  
10  
20  
30  
40  
Done!
```

66. Write a while loop that prints out the value of x plus 10 while x is less than or equal to 15 and the remainder of x divided by 5 is 0. Be careful not to end up with an infinite loop!

x = 10

```
x = 10
```

```
while x <= 15 and x % 5 == 0:  
    print(x + 10)  
    x = x + 1
```

67. Write a while loop that prints out the absolute value of x while x is negative. Be careful not to end up with an infinite loop!

`x = -7`

#Edcorner Learning Python Exercises

```
x = -7
```

```
while x < 0:  
    print(abs(x))  
    x = x + 1
```

```
7  
6  
5  
4  
3  
2  
1
```


68. Write a while loop that prints out the value of x times y while x is greater than or equal to 5 and less than 10, and prints out the result of x divided by y when x becomes 10. Be careful not to end up with an infinite loop!

$$x = 5$$

$$y = 2$$

#Edcorner Learning Python Exercises

```
x = 5  
y = 2
```

```
while x >= 5 and x < 10:  
    print(x * y)  
    x = x + 1  
else:  
    print(x / y)
```

```
10  
12  
14  
16  
18  
5.0
```

69. Write code that will iterate over the x list and multiply by 10 only the elements that are greater than 20 and print them out to the screen.

Hint: use nesting!

```
x = [10, 12, 13, 14, 17, 19, 21, 22, 25]
```

```
#Edcorner Learning Python Exercises
```

```
x = [10, 12, 13, 14, 17, 19, 21, 22,  
25]
```

```
for i in x:  
    if i > 20:  
        print(i * 10)
```

```
210
```

```
220
```

```
250
```

70. Write code that will iterate over the x and y lists and multiply each element of x with each element of y, also printing the results to the screen.

Hint: use nesting!

```
x = [2, 4, 6]
```

```
y = [5, 10]
```

#Edcorner Learning Python Exercises

```
x = [2, 4, 6]
```

```
y = [5, 10]
```

```
for i in x:  
    for j in y:  
        print(i * j)
```

```
10  
20  
20  
40  
30  
60
```

71. Write code that will iterate over the x and y lists and multiply each element of x with each element of y that is less than 12, also printing the results to the screen.

Hint: use nesting!

```
x = [2, 4, 6, 8]
```

```
y = [5, 10, 15, 20]
```

#Edcorner Learning Python Exercises

```
x = [2, 4, 6, 8]
y = [5, 10, 15, 20]

for i in x:
    for j in y:
        if j < 12:
            print(i * j)
```

```
10
20
20
40
30
60
40
80
```


72. Write code that will iterate over the x and y lists and multiply each element of x that is greater than 5 with each element of y that is less than 12, also printing the results to the screen.

Hint: use nesting!

```
x = [2, 4, 6, 8]
```

```
y = [5, 10, 15, 20]
```

#Edcorner Learning Python Exercises

```
x = [2, 4, 6, 8]  
y = [5, 10, 15, 20]
```

```
for i in x:  
    for j in y:  
        if i > 5 and j < 12:  
            print(i * j)
```

```
30  
60  
40  
80
```

73. Write code that will iterate over the x and y lists and multiply each element of x with each element of y that is less than or equal to 10, also printing the results to the screen. For y's elements that are greater than 10, multiply each element of x with y squared.

Hint: use nesting!

```
x = [2, 4, 6, 8]
```

```
y = [5, 10, 15, 20]
```

#Edcorner Learning Python Exercises

```
x = [2, 4, 6, 8]
y = [5, 10, 15, 20]

for i in x:
    for j in y:
        if j <= 10:
            print(i * j)
        else:
            print(i * j ** 2)
```

```
10
20
450
800
20
40
900
1600
30
60
1350
2400
40
80
1800
3200
```

74. Write code that will print out each character in x doubled if that character is also inside y. Hint: use nesting!

```
x = "cryptocurrency"
```

```
y = "blockchain"
```

#Edcorner Learning Python Exercises

```
x = "cryptocurrency"  
y = "blockchain"  
  
for i in x:  
    if i in y:  
        print(i * 2)
```

```
cc  
oo  
cc  
nn  
cc
```

75. Write code that will iterate over the range generated by `range(9)` and for each element that is between 3 and 7 inclusively print out the result of multiplying that element by the second element in the same range.

Hint: use nesting!

```
my_range = range(9)
```

#Edcorner Learning Python Exercises

```
my_range = range(9)

for i in my_range:
    if 3 <= i <= 7:
        print(i * my_range[1])
```



```
3
4
5
6
7
```


76. Write code that will iterate over the range starting at 1, up to but not including 11, with a step of 2, and for each element that is between 3 and 8 inclusively print out the result of multiplying that element by the last element in the same range. For any other element of the range (outside [3-8]) print Outside!

Hint: use nesting!

#Edcorner Learning Python Exercises

```
for i in range(1,11,2):  
    if 3 <= i <= 8:  
        print(i * range(1,11,2)[-1])  
    else:  
        print("Outside!")
```

Outside!

27

45

63

Outside!

77. Write code that will iterate over the range starting at 5, up to but not including 25, with a step of 5, and for each element that is between 10 and 21 inclusively print out the result of multiplying that element by the second to last element of the same range. For any other element of the range (outside [10-21]) print Outside! Finally, after the entire range is exhausted print out The end!

Hint: use nesting!

#Edcorner Learning Python Exercises

```
for i in range(5,25,5):
    if 10 <= i <= 21:
        print(i * range(5,25,5)[-2])
    else:
        print("Outside!")
else:
    print("The end!")
```

```
Outside!
150
225
300
The end!
```

78. Write a while loop that prints out the value of x times 11 while x is less than or equal to 11. When x becomes equal to 10, print out x is 10! Be careful not to end up with an infinite loop!

x = 5

#Edcorner Learning Python Exercises

```
x = 5
```

```
while x <= 11:  
    if x == 10:  
        print("x is 10!")  
        x = x + 1  
    else:  
        print(x * 11)  
        x = x + 1
```

```
55  
66  
77  
88  
99  
x is 10!  
121
```

79. Insert a break statement where necessary in order to obtain the following result:

1

1

100

20

10

```
        x = [1, 2]
        y = [10, 100]
    for i in x:
        for j in y:
            if i % 2 == 0:
print(i * j)
        print(i)
        print(j)
```



```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            print(i * j)
            break
        print(i)
    print(j)
```

```
1
1
100
20
10
```

80. Insert a break statement where necessary in order to obtain the following result:

```
1
1
100
20
10
```

```
x = [1, 2]
y = [10, 100]
for i in x:
    for j in y:
        if i % 2 == 0:
            print(i * j)
            print(i)
            print(j)
```

#Edcorner Learning Python Exercises

```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            print(i * j)
        print(i)
        break
    print(j)
```

```
1
10
20
2
10
```

81. Insert a break statement where necessary in order to obtain the following result:

1

1

100

10

```
x = [1, 2]  
y = [10, 100]
```

```
    for i in x:  
for j in y:  
    if i % 2 == 0:  
        print(i * j)  
        print(i)  
        print(j)
```

#Edcorner Learning Python Exercises

```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            break
        print(i * j)
        print(i)
        print(j)
```

```
1
1
100
10
```

82. Insert a continue statement where necessary in order to obtain the following result:

1

1

100

20

200

```
x = [1, 2]
    y = [10, 100]
        for i in x:
            for j in y:
                if i % 2 == 0:
                    print(i * j)
            print(i)
        print(j)
```



```
#Edcorner Learning Python Exercises
```

```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            print(i * j)
            continue
        print(i)
    print(j)
```

```
1
1
100
20
200
100
```

83. Insert a continue statement where necessary in order to obtain the following result:

1

1

100

100

```
    x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            print(i * j)
        print(i)
    print(j)
```

#Edcorner Learning Python Exercises

```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            continue
        print(i * j)
        print(i)
        print(j)
```

```
1
1
100
100
```

Module 6 Exceptions

84. Fix the code below so that it doesn't generate a `SyntaxError`. Hint! The result should be 20 200

```
x = [1, 2]
y = [10, 100]
    for i in x:
        for j in y:
            if i % 2 == 0
                print(i * j)
```

```
#Edcorner Learning Python Exercises
```

```
x = [1, 2]
```

```
y = [10, 100]
```

```
for i in x:
```

```
    for j in y:
```

```
        if i % 2 == 0:
```

```
            print(i * j)
```

```
20
```

```
200
```

85. Fix the code below so that it doesn't generate a `TypeError`. Hint!
The result should be 20 200

```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if y % 2 == 0:
            print(i * j)
```



```
#Edcorner Learning Python Exercises
```

```
x = [1, 2]
```

```
y = [10, 100]
```

```
for i in x:
```

```
    for j in y:
```

```
        if i % 2 == 0:
```

```
            print(i * j)
```

➤

```
20
```

```
200
```

86. Fix the code below so that it doesn't generate an `IndexError`. Hint! The result should be 200 200

```
x = [1, 2]
y = [10, 100]
for i in x:
    for j in y:
        if i % 2 == 0:
            print(x[1] * y[2])
```

#Edcorner Learning Python Exercises

```
x = [1, 2]
y = [10, 100]

for i in x:
    for j in y:
        if i % 2 == 0:
            print(x[1] * y[1])
```

200

200

87. Add the necessary clause(s) to the code below so that in case the `ZeroDivisionError` exception is raised then the program prints out `Zero!` to the screen.

try:

```
    print(25 % 0)
```

#Edcorner Learning Python Exercises

```
try:  
    print(25 % 0)  
except ZeroDivisionError:  
    print("Zero!")
```

Zero!

88. Add the necessary clause(s) to the code below so that in case the code under try raises no exceptions then the program prints out the result of the math operation and the string Clean! to the screen.

```
try:
```

```
    print(25 % 5 ** 5 + 5)
```

```
except:
```

```
    print("Bug!")
```

#Edcorner Learning Python Exercises

```
try:  
    print(25 % 5 ** 5 + 5)  
except:  
    print("Bug!")  
else:  
    print("Clean!")
```

30
Clean!

89. Add the necessary clause(s) to the code below so that no matter if the code under try raises any exceptions or not, then the program prints out the string Result! to the screen.

```
try:
    print(25 % 0 ** 5 + 5)
except:
    print("Bug!")
```


#Edcorner Learning Python Exercises

```
try:  
    print(25 % 0 ** 5 + 5)  
except:  
    print("Bug!")  
finally:  
    print("Result!")
```



```
Bug!  
Result!
```

90. Add the necessary clause(s) to the code below so that in case the code under try raises the `ZeroDivisionError` exception then the program prints out the string `Zero!` to the screen; additionally, if the code under try raises the `IndexError` exception then the program prints out the string `Index!` to the screen.

```
x = [1, 9, 17, 32]
    try:
print(x[3] % 3 ** 5 + x[4])
```

#Edcorner Learning Python Exercises

```
x = [1, 9, 17, 12]
```

```
try:
```

```
    print(x[3] % 3 ** 5 + x[4])
```

```
except ZeroDivisionError:
```

```
    print("Zero!")
```

```
except IndexError:
```

```
    print("Index!")
```

Index!

91. Add the necessary clause(s) to the code below so that in case the code under try raises no exceptions then the program prints out the result of the math operation and the string Clean! to the screen. If the code under try raises the ZeroDivisionError exception then the program prints Zero! to the screen. Ultimately, regardless of the result generated by the code under try, the program should print out Finish! to the screen.

```
try:
```

```
    print(25 % 5 ** 5 + 5)
```

#Edcorner Learning Python Exercises

```
try:
    print(25 % 5 ** 5 + 5)
except ZeroDivisionError:
    print("Zero!")
else:
    print("Clean!")
finally:
    print("Finish!")
```


```
30
Clean!
Finish!
```

Module 7 Functions

93. Implement a function called `my_func()` that simply prints out Hello Python! to the screen and call the function.

#Edcorner Learning Python Exercises

```
def my_func():  
    print("Hello Python!")  
  
my_func()
```

A screenshot of a terminal window with a black background and white text. The text "Hello Python!" is displayed on the first line.

Hello Python!

94. Implement a function called `my_func()` that creates a variable `add` which stores the result of adding 10 and 20, and prints out the value of `add`. Don't forget to also call the function!


```
def my_func():  
    add = 10 + 20  
    print(add)  
  
my_func()
```

95. Implement a function called `my_func()` that takes a single parameter `x` and multiplies it with 10, also returning the result when the function is called

```
result = my_func(7)
```

```
print(result)
```

```
def my_func(x):  
    return x * 10  
  
result = my_func(7)  
print(result)
```

96. Implement a function called `my_func()` that takes two parameters `x` and `y` and divides `x` by `y`, also returning the result when the function is called.

```
result = my_func(38,19)
```

```
print(result)
```

#Edcorner Learning Python Exercises

```
def my_func(x, y):  
    return x / y  
  
result = my_func(38,19)  
print(result)
```

2.0

97. Implement a function called `my_func()` that takes 3 parameters `x`, `y` and `z` and raises `x` to the power of `y` then adds `z`, also returning the result when the function is called.

```
result = my_func(3,3,3)
print(result)
```

```
def my_func(x,y,z):  
    return x ** y + z  
  
result = my_func(3,3,3)  
print(result)
```

98. Implement a function called `my_func()` that takes a single parameter `x` and multiplies it with each element of `range(5)`, also adding each multiplication result to a new (initially empty) list called `my_new_list`. Finally, the list should be printed out to the screen after the function is called.

```
result = my_func(2)
```

```
print(result)
```



```
def my_func(x):  
    my_new_list = []  
    for i in range(5):  
        my_new_list.append(i * x)  
    return my_new_list  
  
result = my_func(2)  
print(result)
```

```
[0, 2, 4, 6, 8]
```

99. Implement a function called `my_func()` that takes a single parameter `x` (a string) and turns each character of the string to uppercase, also returning the result when the function is called

```
result = my_func("Edcorner Learning")  
print(result)
```

```
#Edcorner Learning Python Exercises
```

```
def my_func(x):  
    return x.upper()
```

```
result = my_func("Edcorner Learning")  
print(result)
```

EDCORNER LEARNING

100. Implement a function called `my_func()` that takes a single parameter `x` (a list) and eliminates all duplicates from the list, also returning the result when the function is called

```
result = my_func([11, 12, 13, 11, 15, 18, 18, 22, 20, 16, 12])  
print(result)
```

#Edcorner Learning Python Exercises

```
def my_func(x):  
    return list(set(x))  
  
result = my_func([11, 12, 13, 11, 15,  
18, 18, 22, 20, 16, 12])  
print(result)
```

```
[11, 12, 13, 15, 16, 18, 20, 22]
```

101. Implement a function called `my_func()` that takes a single parameter `x` (a tuple) and for each element of the tuple that is greater than 4 it raises that element to the power of 2, also adding it to a new (initially empty) list called `my.newlist`. Finally, the code returns the result when the function is called.

```
result = my_func((2, 3, 5, 6, 4, 8, 9))  
print(result)
```

#Edcorner Learning Python Exercises

```
def my_func(x):  
    my_new_list = []  
    for i in x:  
        if i > 4:  
            my_new_list.append(i ** 2)  
    return my_new_list  
  
result = my_func((2, 3, 5, 6, 4, 8, 9))  
print(result)
```

[25, 36, 64, 81]

102. Implement a function called `my_func()` that takes a single parameter `x` (a dictionary) and multiplies the number of elements in the dictionary with the largest key in the dictionary, also returning the result when the function is called.

```
result = my_func({1: 3, 2: 3, 4: 5, 5: 9, 6: 8, 3: 7, 7: 0})  
print(result)
```



```
def my_func(x):  
    return len(x) * sorted(x.keys())  
[-1]  
  
result = my_func([1: 3, 2: 3, 4: 5, 5:  
9, 6: 8, 3: 7, 7: 0])  
print(result)
```



103. Implement a function called `my_func()` that takes a single positional parameter `x` and a default parameter `y` which is equal to 10 and multiplies the two, also returning the result when the function is called.

```
result = my_func(5)
print(result)
```

```
def my_func(x, y = 10):  
    return x * y  
  
result = my_func(5)  
print(result)
```

104. Implement a function called `my_func()` that takes a single positional parameter `x` and two default parameters `y` and `z` which are equal to 100 and 200 respectively, and adds them together, also returning the result when the function is called.

```
result = my_func(50)
print(result)
```

#Edcorner: Learning Python Exercises

```
def my_func(x, y = 100, z = 200):  
    return x + y + z  
  
result = my_func(50)  
print(result)
```

350

105. Implement a function called `my_func()` that takes two default parameters `x` (a list) and `y` (an integer), and returns the element in `x` positioned at index `y`, also printing the result to the screen when called.

```
result = my_func(list(range(2,25,2)), 4)
print(result) #result should be 10
```

```
def my_func(x, y):  
    return x[y]  
  
result = my_func(list(range(2,25,2)),  
4)  
print(result)
```

106. Implement a function called `my_func()` that takes a positional parameter `x` and a variable-length tuple of parameters and returns the result of multiplying `x` with the second element in the tuple, also returning the result when the function is called.

```
result = my_func(5, 10, 20, 30, 50)
print(result)
```


#Edcorner Learning Python Exercises

```
def my_func(x, *args):  
    return x * args[1]  
  
result = my_func(5, 10, 20, 30, 50)  
print(result)
```



100

107. Implement a function called `my_func()` that takes a positional parameter `x` and a variable-length dictionary of (keyword) parameters and returns the result of multiplying `x` with the largest value in the dictionary, also returning the result when the function is called.

```
result = my_func(10, val1 = 10, val2 = 15, val3 = 20, val4  
= 25, val5 = 30)  
print(result)
```

```
def my_func(x, **kwargs):  
    return x * sorted(kwargs.values())  
[-1]  
  
result = my_func(10, val1 = 10, val2 =  
15, val3 = 20, val4 = 25, val5 = 30)  
print(result)
```

108. Add the correct line(s) of code inside the function in order to get 200 as a result of calling myfunc() and have the result printed out to the screen.

```
var = 10  
def my_func(x):  
    my_func(20)
```

#Edcorner Learning Python Exercises

```
var = 10
```

```
def my_func(x):  
    print(x * var)
```

```
my_func(20)
```

200

109. Add the correct line(s) of code inside the function in order to get 100 as a result of calling myfunc() and have the result printed out to the screen.

```
var = 10
def my_func(x):
    print(x * var)
    my_func(20)
```

```
var = 10
```

```
def my_func(x):  
    var = 5  
    print(x * var)
```

```
my_func(20)
```

110. Make the necessary adjustment inside the function in order to get 120 as a result of calling **myfunc()** and have the result printed out to the screen.

```
def my_func(x):  
    print(x * var)  
var = 12  
  
my_func(10)
```


#Edconner Learning Python Exercises

```
def my_func(x):  
    var = 12  
    print(x * var)
```

```
my_func(10)
```

↳

120

111. Add the necessary line of code inside the function in order to get 80 as a result of calling myfunc() and have the result printed out to the screen.

```
var = 8
def my_func(x):
    print(x * var)
var = 12
my_func(10)
```

#Edcorner Learning Python Exercises

```
var = 8
```

```
def my_func(x):  
    global var  
    print(x * var)  
    var = 12
```

```
my_func(10)
```

112. Write code that will import only the pi variable from the math module and then it will format it in order to have only 4 digits after the floating point. Of course, print out the result to the screen using the print() function.

#Edcorner Learning Python Exercises

```
from math import pi  
print("%.4f" % pi)
```

3.1416

113. Add the necessary code in between print's parentheses in order to read the content of test.txt as a string and have the result printed out to the screen.

```
f = open("test.txt", "r")  
    print()
```

Solution:

```
f = open("test.txt", "r")  
    print(f.read())
```

114. Add the necessary code in between print's parentheses in order to read the content of test.txt as a list where each element of the list is a row in the file, and have the result printed out to the screen.

```
f = open("test.txt", "r")  
print()
```


Solution:

```
f = open("test.txt", "r")  
print(f.readlines())
```

115. Add the necessary code on line 5 in order to bring back the cursor at the very beginning of test.txt before reading from the file once again.

```
f = open("test.txt", "r")  
f.read()  
print(f.read())
```

Solution:

```
f = open("test.txt", "r")
```

```
f.read()
```

```
f.seek(0)
```

```
print(f.read())
```

116. Add the necessary code on line 5 (in between the parentheses of `print()`) in order to get the current position of the cursor inside `test.txt` and have the result printed out to the screen.

```
f = open("test.txt", "r")
```

```
f.read(5)
```

```
print()
```

Solution:

```
f = open("test.txt", "r")  
f.read(5)  
    print(f.tell())
```

117. Add the necessary code on line 5 (in between the parentheses of `print()`) in order to get the current mode in which `test.txt` is open (read, write etc.) and have the result printed out to the screen.

```
f = open("test.txt", "r")
```

```
f.read(5)
```

```
print()
```

Solution:

```
f = open("test.txt", "r")
```

```
f.read(5)
```

```
print(f.mode)
```

118. Add the necessary file access mode on line 1 in order to open test.txt for appending and reading at the same time.

```
f = open("test.txt", )
```

```
print(f.mode)
```


Solution:

```
f = open("test.txt", "a + ")
```

```
print(f.mode)
```

119. Add the necessary code on lines 3 and 4 in order to write the string python to test.txt and have the result of reading the file printed out to the screen.

```
f = open("test.txt", "w")  
f = open("test.txt", "r")  
print(f.read())
```

Solution:

```
f = open("test.txt", "w")  
f.write("python")  
f.close()  
f = open("test.txt", "r")  
print(f.read())
```

120. Add the necessary code on lines 3 and 4 in order to write a list of strings ['python', ' ', 'and', ' ' 'java'] to test.txt and have the result of reading the file printed out to the screen.

```
f = open("test.txt", "w")
```

```
f = open("test.txt", "r")
```

```
print(f.read())
```

Solution:

```
f = open("test.txt", "w")  
f.writelines(['python', ' ', 'and', ' ', 'java'])  
f.close()  
f = open("test.txt", "r")  
print(f.read())
```

121. Add the necessary code starting at line 1 in order to write the string python and also close test.txt properly using the with statement.

```
f = open("test.txt", "r")
```

```
print(f.read())
```

Solution :

with open("test.txt", "w") as f:

f.write("python")

f = open("test.txt", "r")

print(f.read())

122. Add the necessary code on lines 4 and 5 in order to delete the entire content of test.txt.

```
    with open("test.txt", "w") as f:  
f.write("python")  
    f = open("test.txt", "r")  
    print(f.read())
```


Solution:

```
with open("test.txt", "w") as f:  
    f.write("python")
```

```
f = open("test.txt", "r+")  
f.truncate()
```

```
f = open("test.txt", "r")  
  
print(f.read())
```

Module 8 Regular Expressions

123. Write code on line 5 in order to match the word Python at the beginning of the string using the match() method.

```
import re
```

```
s = " Python exercises consist of various Modules
```

```
result =
```

```
print(result.group())
```

#Edcorner Learning Python Exercises

```
import re

s = "Python exercises consist of
various Modules."

result = re.match("Python", s)

print(result.group())
```



Python

124. Write code on line 5 in order to match the word Bitcoin at the beginning of the string using the match() method and ignoring the case. This way, no matter if you have bitcoin or Bitcoin, the match is done either way.

```
import re
```

```
s = "Bitcoin was born on Jan 3rd 2009 as an alternative to the failure  
of the current financial system. In 2017, the price of 1 BTC reached  
$20000, with a market cap of over $300B."
```

```
print(result.group())
```

```
import re

s = "Bitcoin was born on Jan 3rd 2009  
as an alternative to the failure of the  
current financial system. In 2017, the  
price of 1 BTC reached $20000, with a  
market cap of over $300B."

result = re.match("Bitcoin", s, re.I)

print(result.group())
```

125. Write code on line 5 in order to match the words Bitcoin was at the beginning of the string using the match() method. Use the dot (.) belonging to regex syntax in your solution.

```
import re

s = "Bitcoin was born on Jan 3rd 2009 as an alternative to the
failure of the current financial system. In 2017, the price of 1 BTC
reached $20000, with a market cap of over $300B."

result =

print(result.group())
```

#Edcorner Learning Python Exercises

```
import re
```

```
s = "Bitcoin was born on Jan 3rd 2009  
as an alternative to the failure of the  
current financial system. In 2017, the  
price of 1 BTC reached $20000, with a  
market cap of over $300B."
```

```
result = re.match(r"B.{6} .{3}", s)
```

```
print(result.group())
```

Bitcoin was

126. Write code on line 5 in order to match the year 2009 in the string using the search() method. Use the \d in your solution.

```
import re
```

```
s = "Bitcoin was born on Jan 3rd 2009 as an alternative to the failure  
of the current financial system. In 2017, the price of 1 BTC reached  
$20000, with a market cap of over $300B."
```

```
result =
```

```
print(result.group(1))
```

```
import re

s = "Bitcoin was born on Jan 3rd 2009
as an alternative to the failure of the
current financial system. In 2017, the
price of 1 BTC reached $20000, with a
market cap of over $300B."

result = re.search(r"(\d{4})\s", s)

print(result.group(1))
```



Module 9 Classes

127. Write a class called ClassOne starting on line 1 containing:

- Theinitmethod with two parameters p1 and p2. Define the corresponding attributes

inside theinitmethod.

- A method called square that takes one parameter p3 and prints out the value of p3 squared.

class

p = ClassOne(1, 2)

print(type(p))


```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

p = ClassOne(1, 2)
print(type(p))
```

```
<class '__main__.ClassOne'>
```

128. Considering the ClassOne class and the p object, write code on line 11 in order to access the p1 attribute for the current instance of the class and print its value to the screen.

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2
    def square(self, p3):
        print(p3 ** 2)
p = ClassOne(1, 2)
```

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

p = ClassOne(1, 2)

print(p.p1)
```

129. Considering the ClassOne class and the p object, write code on line 11 in order to call the square() method for the current instance of the class using 10 as an argument and print the result to the screen.

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
self.p1 = p1  
self.p2 = p2  
    def square(self, p3):  
print(p3 ** 2)  
    p = ClassOne(1, 2)
```



```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

p = ClassOne(1, 2)

p.square(10)
```

130. Considering the ClassOne class and the p object, write code on line 11 in order to set the value of the p2 attribute to 5 for the current instance of the class, without using a function.

```
class ClassOne(object):
```

```
    def __init__(self, p1, p2):
```

```
        self.p1 = p1
```

```
        self.p2 = p2
```

```
    def square(self, p3):
```

```
        print(p3 ** 2)
```

```
p = ClassOne(1, 2)
```

```
print(p.p2)
```

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

p = ClassOne(1, 2)

p.p2 = 5

print(p.p2)
```

131. Considering the ClassOne class and the p object, write code on lines 11 and 12 in order to set the value of the p2 attribute to 50 for the current instance of the class using a function, and then get the new value of p2, again using a function, and print it out to the screen as well.

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
  
        self.p1 = p1  
        self.p2 = p2  
  
def square(self, p3):  
    print(p3 ** 2)  
  
    p = ClassOne(1, 2)
```

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

p = ClassOne(1, 2)

setattr(p, 'p2', 50)
print(getattr(p, 'p2'))
```

132. Considering the ClassOne class and the p object, write code on line 11 in order to check if p2 is an attribute of p, using a function, also printing the result to the screen.

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
        self.p1 = p1  
        self.p2 = p2  
    def square(self, p3):  
        print(p3 ** 2)  
p = ClassOne(1, 2)
```

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
        self.p1 = p1  
        self.p2 = p2  
  
    def square(self, p3):  
        print(p3 ** 2)  
  
p = ClassOne(1, 2)  
print(hasattr(p, 'p2'))
```

True

133. Considering the ClassOne class and the p object, write code on line 11 to check if p is indeed an instance of the ClassOne class, using a function, also printing the result to the screen.

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
        self.p1 = p1  
        self.p2 = p2  
    def square(self, p3):  
        print(p3 ** 2)  
        p = ClassOne(1, 2)
```


#Edcorner Learning Python Exercises

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
        self.p1 = p1  
        self.p2 = p2  
  
    def square(self, p3):  
        print(p3 ** 2)  
  
p = ClassOne(1, 2)  
  
print(isinstance(p, ClassOne))
```

True

134. Considering the ClassOne class, write code starting on line 9 to create a child class called ClassTwo that inherits from ClassOne and also has its own method called times10() that takes a single parameter x and prints out the result of multiplying x by 10.

```
class ClassOne(object):
```

```
    def __init__(self, p1, p2):
```

```
        self.p1 = p1
```

```
        self.p2 = p2
```

```
    def square(self, p3):
```

```
        print(p3 ** 2)
```

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

class ClassTwo(ClassOne):
    def times10(self, x):
        print(x * 10)

y = ClassTwo(10, 20)
print(y.p1)
```

135. Considering the ClassOne and ClassTwo classes, where the latter is a child of the former, write code on line 15 in order to call the times10() method from the child class having x equal to 45, also printing the result to the screen.

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
        self.p1 = p1  
        self.p2 = p2  
  
    def square(self, p3):  
        print(p3 ** 2)  
  
class ClassTwo(ClassOne):  
    def times10(self, x):  
        return x * 10  
  
obj = ClassTwo(15, 25)
```

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

class ClassTwo(ClassOne):
    def times10(self, x):
        return x * 10

obj = ClassTwo(15, 25)

print(obj.times10(45))
```

136. Considering the ClassOne and ClassTwo classes, write code on line 13 to verify that ClassTwo is indeed a child of ClassOne, also printing the result to the screen.

```
class ClassOne(object):  
    def __init__(self, p1, p2):  
self.p1 = p1  
self.p2 = p2
```

```
    def square(self, p3):  
print(p3 ** 2)
```

```
class ClassTwo(ClassOne):  
    def times10(self, x):  
return x * 10
```

```
class ClassOne(object):
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def square(self, p3):
        print(p3 ** 2)

class ClassTwo(ClassOne):
    def times10(self, x):
        return x * 10

print(issubclass(ClassTwo, ClassOne))
```

Module 10 Strings

137. Given the code below, insert the correct negative index on line 3 in order to get the last character in the string.

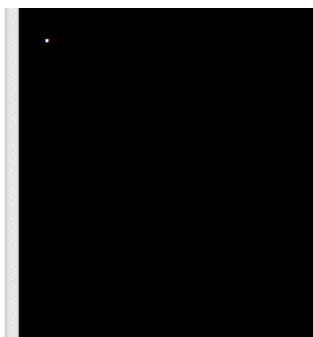
```
my_string = "In 2010, someone paid 10k Bitcoin for two  
pizzas."
```

```
print(my_string[])
```


#Edcorner Learning Python Exercises

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."
```

```
print(my_string[-1])
```



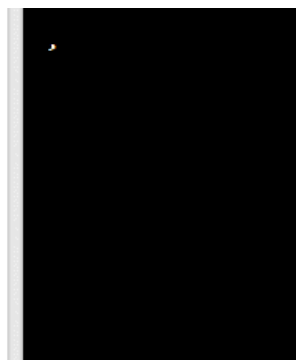
138. Given the code below, insert the correct positive index on line 3 in order to return the comma character from the string.

```
my_string = "In 2010, someone paid 10k Bitcoin for two pizzas."
```

```
print(my_string[])
```

#Edcorner Learning Python Exercises

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."  
print(my_string[7])
```



139. Given the code below, insert the correct negative index on line 3 in order to return the w character from the string.

```
my_string = "In 2010, someone paid 10k Bitcoin for two  
pizzas."  
  
print(my_string[])
```

#tdcorner Learning Python Exercises

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."
```

```
print(my_string[-10])
```

W

140. Given the code below, insert the correct method on line 3 in order to return the index of the B character in the string.

```
my_string = "In 2010, someone paid 10k Bitcoin for two  
pizzas."  
  
print(my_string.)
```

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."  
print(my_string.index("B"))
```

141. Given the code below, insert the correct method on line 3 in order to return the number of occurrences of the letter o in the string.

```
my_string = "In 2010, someone paid 10k Bitcoin for  
two pizzas."  
  
print(my_string.)
```


#Edcorner Learning Python Exercises

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."
```

```
print(my_string.count("o"))
```

5

142. Given the code below, insert the correct method on line 3 in order to convert all letters in the string to uppercase.

```
my_string = "In 2010, someone paid 10k Bitcoin for two  
pizzas."
```

```
print(my_string.)
```

#Edcorner Learning Python Exercises

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."
```

```
print(my_string.upper())
```

```
IN 2010, SOMEONE PAID 10K BITCOIN FOR TWO PIZZAS.
```

143. Given the code below, insert the correct method on line 3 in order to get the index at which the substring Bitcoin starts.

```
my_string = "In 2010, someone paid 10k Bitcoin for two  
pizzas."  
  
print(my_string.)
```

#Edcorner Learning Python Exercises

```
my_string = "In 2010, someone paid 10k  
Bitcoin for two pizzas."
```

```
print(my_string.find("Bitcoin"))
```



Module 11 Dictionary

144. Create the **get_name** function which returns the value of the key **name**

such key does not exist the function should return the string **unknown** name .

def get_name(dictionary: dict) -> str:

```
    # TODO
```

```
arr = [
```

```
    {
```

```
        'name': 'John',
```

```
        'age': 25
```

```
    },
```

```
    {
```

```
        'age': 20
```

```
    },
```

```
    {
```

```
        'name': 'Tom',
```

```
        'age': 38
```

```
    },
```

```
    {}
```

```
]
```

```
for dictionary in arr:  
    print(get_name(dictionary))
```


#Edcorner Learning Python Exercises

```
def get_name(dictionary: dict) -> str:  
    return dictionary.get('name',  
        'Unknown name')
```

```
arr = [  
    {  
        'name': 'John',  
        'age': 25  
    },  
    {  
        'age': 20  
    },  
    {  
        'name': 'Tom',  
        'age': 38  
    },  
    {}  
]
```

```
for dictionary in arr:  
    print(get_name(dictionary))
```

```
John  
Unknown name  
Tom  
Unknown name
```

145. Create the **get_name** function which returns the value of the key **name**

such key does not exist the function should return the string **unknown** name .

```
def get_name(dictionary: dict) -> str:
```

```
    # TODO
```

```
arr = [
```

```
    {
```

```
        'name': 'John',
```

```
        'age': 25
```

```
    },
```

```
    {
```

```
        'age': 20
```

```
    },
```

```
    {
```

```
        'name': 'Tom',
```

```
        'age': 38
```

```
    },
```

```
    {}
```

```
]
```

```
for dictionary in arr:  
    print(get_name(dictionary))
```

#Edcorner Learning Python Exercises

```
def get_name(dictionary: dict) -> str:  
    return  
dictionary.setdefault('name', 'Unknown  
name')
```

```
arr = [  
    {  
        'name': 'John',  
        'age': 25  
    },  
    {  
        'age': 20  
    },  
    {  
        'name': 'Tom',  
        'age': 38  
    },  
    {}  
]
```

```
for dictionary in arr:  
    print(get_name(dictionary))
```

```
John  
Unknown name  
Tom  
Unknown name
```

146. Sort the arr list of dictionaries by a value in the **age** key. Sorted list save in the sorted_arr variable.

Expected result:

```
[{'name': 'Alice', 'age': 20}, {'name': 'lohn', 'age': 25}, {'name': 'Tom', 'age': 38}]
```

```
arr = [  
    {  
        'name': 'John',  
        'age': 25  
    },  
    {  
        'name': 'Alice',  
        'age': 20  
    },  
    {  
        'name': 'Tom',  
        'age': 38  
    }  
]  
  
sorted_arr =
```

Solution:

```
arr = [  
    {  
        'name': 'John',  
        'age': 25  
    },  
    {  
        'name': 'Alice',  
        'age': 20  
    },  
    {  
        'name': 'Tom',  
        'age': 38  
    }  
]  
  
sorted_arr = sorted(arr, key=lambda key:  
key['age'])
```

147. Convert two lists into the **dictionary**. The **keys** list stores the keys of the dictionary, and the **values** list stores their values.

Expected result:

```
{'name': 'john', 'age': 25, 'grade': 'B'}
```

```
keys = ['name', 'age', 'grade']
```

```
values = ['John', 25, 'B']
```

```
dictionary =
```

Solution:

```
keys = ['name', 'age', 'grade']
```

```
values = ['John', 25, 'B']
```

```
dictionary = dict(zip(keys, values))
```


148. Create a list from the **dictionary** keys. Save the result in the **arr**

variable.

Expected result:

['name', 'age', 'grade']

```
dictionary = {  
'name': 'John',  
'age': 25,  
'grade': 'B',  
}  
arr =
```

Solution:

```
dictionary = {  
    'name': 'John',  
    'age': 25,  
    'grade': 'B',  
    }  
arr = list(dictionary.keys())
```

149. Merge two dictionaries: dict_1 and dict_2. Save the result in the new_dict dictionary.

Expected result:

{'name': 'Anna', 'age': 20, 'weight': 55, 'height': 170}

```
dict_1 = {'name': 'Anna', 'age': 20}
```

```
dict_2 = {'weight': 55, 'height': 170}
```

```
new_dict =
```

Solution:

```
dict_1 = {'name': 'Anna', 'age': 20}
```

```
dict_2 = {'weight': 55, 'height': 170}
```

```
new_dict = {**dict_1, **dict_2}
```

```
# Python 3.9+
```

```
# new_dict = dict_1 | dict_2
```

150. Sort the dictionary by the key in ascending order.

Expected result:

```
{'Alice': 10, 'Anna': 20, 'Clinton': 18, 'Mark': 15}
```

```
dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}
```

```
dictionary =
```

Solution:

dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice':

10}

dictionary = dict(sorted(dictionary.items()))

151. Add the `name` key with the value `Mark` to the dictionary.

Expected result:

```
{'weight': 55, 'height': 170, 'name': 'Mark'}
```

```
dictionary = {'weight': 55, 'height': 170}
```

Solution

```
dictionary = {'weight': 55, 'height': 170}
```

```
dictionary['name'] = 'Mark'
```


152. Convert the **dictionary** into a list of tuples. The first element of tuples should be a dictionary key and the second one a dictionary value. Save the result in the **arr_of_tupies** variable.

Expected result:

```
[('name', 'Anna'), ('age', 20), ('weight', 55), ('height', 170)]
```

```
dictionary = {'name': 'Anna', 'age': 20, 'weight': 55, 'height': 170}
```

```
arr_of_tuples =
```

Solution

```
dictionary = {'name': 'Anna', 'age': 20, 'weight': 55, 'height': 170}
```

```
arr_of_tuples = [(key, value) for key, value in dictionary.items()]
```

153. Create the key_exists function which checks if the name key exists in a given dictionary. The function Should return True Or False .

```
def key_exists(dictionary: dict) -> bool:
    arr_of_dicts = [
        {
            'name': 'John',
            'age': 25
        },
        {
            'color': 'blue'
        },
        {
            'height': 175,
            'weight': 55
        }
    ]
```

```
    for dictionary in arr_of_dicts:
        if key_exists(dictionary):
            print(f"name" key exists in {dictionary})
else:

    print(f"name" key does not exist in {dictionary})
```

Solution

```
def key_exists(dictionary: dict) -> bool:  
    return 'name' in dictionary
```

```
arr_of_dicts = [  
    {  
        'name': 'John',  
        'age': 25  
    },  
    {  
        'color': 'blue'  
    },  
    {  
        'height': 175,  
        'weight': 55  
    }  
]
```

```
for dictionary in arr_of_dicts:
    if key_exists(dictionary):
        print(f"name" key exists in {dictionary}')
else:
    print(f"name" key does not exist in {dictionary}')
```

154. Remove the name key from the dictionary and save its value-to-value variable.

Expected result:

'Anna'

dictionary = {'name': 'Anna', 'age': 20, 'weight': 55, 'height':

170}

value =

Solution

```
dictionary = {'name': 'Anna', 'age': 20, 'weight': 55, 'height':  
170}
```

```
value = dictionary.pop('name')
```


155. Create the new_dict dictionary with all elements from the dictionary except for the age key.

Use dict comprehension.

Expected result:

```
{'name': 'Anna', 'weight': 55, 'height': 170}
```

```
dictionary = {'name': 'Anna', 'age': 20, 'weight': 55, 'height': 170}
```

```
new_dict =
```

Solutions

```
dictionary = {'name': 'Anna', 'age': 20, 'weight': 55, 'height': 170}
```

```
new_dict = {key: value for key, value in dictionary.items() if key  
!= 'age'}
```

156. Get the key with maximum value in the dictionary. Save the result in the key variable.

Expected result:

'Anna'

```
dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}
```

```
key =
```

Solution

```
dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}
```

```
key = max(dictionary, key=dictionary.get)
```

```
# or
```

```
    # key = max(dictionary, key=lambda item:  
dictionary[item])
```

157. Get the smallest value from the dictionary. Save the result in the value variable.

Expected result:

10

dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}

value =

Solution

```
dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}
```

```
value = min(dictionary.values())
```

158. Get the name with the smallest value in the age key. Save the result in the value variable.

Expected result:

'Alice'

```
arr_of_dicts = [  
    {  
        'name': 'John',  
        'age': 26  
    },  
    {  
        'name': 'Anna',  
        'age': 28  
    },  
    {  
        'name': 'Alice',  
        'age': 25  
    }  
]  
  
value =
```

Solution:

```
arr_of_dicts = [  
    {  
        'name': 'John',  
        'age': 26  
    },  
    {  
        'name': 'Anna',  
        'age': 28  
    },  
    {  
        'name': 'Alice',  
        'age': 25  
    }  
]  
  
value = min(arr_of_dicts, key=lambda item: item['age'])  
['name']
```


159. Create a copy of the dictionary and save it in the `new_dictionary` variable (shallow copy).

```
dictionary = {'name': 'Anna', 'age': 20}
```

```
new_dictionary =
```

Solution

```
dictionary = {'name': 'Anna', 'age': 20}
```

```
new_dictionary = dictionary.copy()
```

```
# or
```

```
# new_dictionary = dict(dictionary)
```

160. Create a copy of the dictionary and save it in the new_dictionary variable (deep copy).

```
dictionary = {'name': 'Anna', 'age': 20, 'skills': {'english': 'perfect',  
'python': 'very good'}}
```

```
new_dictionary =
```

Solution

```
import copy
```

```
dictionary = {'name': 'Anna', 'age': 20, 'skills': {'english': 'perfect',  
'python': 'very good'}}
```

```
new_dictionary = copy.deepcopy(dictionary)
```

161. Get dictionary values as a list. Save the result in the
names
variable.

Expected result:

['Anna', 'Mark', 'Clinton', 'Alice']

dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}

names =

Solution

```
dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}
```

```
names = [*dictionary]
```

```
# or
```

```
    # names = list(dictionary)
```

162. Invert the dictionary mapping.

Expected result:

{20: 'Anna', 15: 'Mark', 18: 'Clinton', 10: 'Alice'}

dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}

dictionary =

Solution

```
dictionary = {'Anna': 20, 'Mark': 15, 'Clinton': 18, 'Alice': 10}
```

```
dictionary = {value: key for key, value in dictionary.items()}
```


163. Count how many times a word occurs in the sentence variable. Save the result in the words dictionary. Every word should be stored as a key of the dictionary and its value should be the number of occurrences of a given word in the variable sentence. Exclude spaces, periods, and exclamation points.

Expected result:

```
{'I': 2, 'am': 1, 'learning': 1, 'Python': 3, 'love': 1, 'is': 1, 'the': 1, 'future': 1}
```

```
sentence = 'I am learning Python. I love Python. Python is the future!'
```

```
words =
```

Solution

```
sentence = 'I am learning Python. I love Python.  
Python is the future!'
```

```
words = {}  
sentence = sentence.replace('.', '').replace('!', '')  
for word in sentence.split():  
    if word not in words:  
        words[word] = 0  
    words[word] += 1
```

or

from collections import Counter

sentence = sentence.replace('.', '').replace('!', '')

words = Counter(sentence.split())

164. Merge two dictionaries dict_1 and dict_2. Add values for keys that appear in both dictionaries. Save the result in the combined_dict dictionary.

Expected result:

{'a': 6, 'b': 5, 'c': 4}

dict_1 = {'a': 1, 'b': 2}

dict_2 = {'a': 5, 'b': 3, 'c': 4}

combined_dict =

Solution

```
from collections import Counter
```

```
dict_1 = {'a': 1, 'b': 2}
```

```
dict_2 = {'a': 5, 'b': 3, 'c': 4}
```

```
counter_1 = Counter(dict_1)
```

```
counter_2 = Counter(dict_2)
```

```
combined_dict = counter_1 + counter_2
```

165. Create the list of dictionaries `arr_of_dicts` where every dictionary will have two keys: `id` and `name`.

The value for the `id` key should start with 1 and then it should be incremented in every

dictionary.

The value for the `name` key should match with elements in the list `names` .

Expected result:

```
[{'id': 1, 'name': 'Alice'}, {'id': 2, 'name':
```

```
{'id': 4, 'name': 'Sara'}]
```

```
'Tom'}, {'id': 3, 'name': 'john'}].
```

```
names = ['Alice', 'Tom', 'John', 'Sara']
```

```
arr_of_dicts =
```

Solution

```
names = ['Alice', 'Tom', 'John', 'Sara']
arr_of_dicts = [
    {
        'id': num,
        'name': name
    } for num, name in enumerate(names, start=1)
]
```

166. Create the dictionary where elements of the their value will be the string Be happy! .

Arr will be stored as keys of dictionary and

Expected result:

{50: 'Be happy!', 51: 'Be happy!', 52: 'Be happy!', 53: 'Be happy!', 54: 'Be happy!'}

```
arr = range(50, 55)
```

```
dictionary =
```


Solution

```
arr = range(50, 55)
```

```
dictionary = dict.fromkeys(arr, "Be happy!")
```

Module 12 Lists

167. Modify the value of the last element in the arr list to 0.

Expected result:

[1, 2, 0]

```
arr = [1, 2, 3]
```

```
arr =
```

Solution

arr = [1, 2, 3]

arr[-1] = 0

168. Create the new arr list which contains only last two elements of the arr list.

Expected result:

[8, 9]

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
new_arr =
```

Solution

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
new_arr = arr[-2:]
```

169. Create the new_arr list which contains only first five elements of the arr list.

Expected result:

[1, 2, 3, 4, 5]

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
new_arr =
```

Solution

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
new_arr = arr[:5]
```


170. Create the new_arr list which contains only the fourth, fifth and sixth element of the arr list.

Expected result:

[4, 5, 6]

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

new_arr =

Solution

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
new_arr = arr[3:6]
```

171. Create the new arr list which contains only odd elements of the arr list.

Expected result: [1, 3, 5, 7, 9]

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

new_arr =

Solution

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
new_arr = arr[::2]
```

```
# or
```

```
# new_arr = [item for item in arr if item % 2 != 0]
```

172. Reverse the order of elements in the arr list. Save the result in the new_arr variable.

Expected result:

[2, 7, 7, 6, 5, 7, 3, 8, 1]

arr = [1, 8, 3, 7, 5, 6, 7, 7, 2]

new_arr =

Solution

```
arr = [1, 8, 3, 7, 5, 6, 7, 7, 2]
```

```
new_arr = arr[::-1]
```

```
# or
```

```
# new_arr = list(reversed(arr))
```

173. Create the function `is_list_empty` which checks if a provided to it list is empty. If a list is empty the function should return `True` otherwise `False`.

```
def is_list_empty(arr: list) -> bool:
    # TODO

arrays = [[1, 2, 3], ['item'], []]
for arr in arrays:
    if is_list_empty(arr):
        print(f'{arr} is a empty list')
    else:
        print(f'{arr} contains elements')
```

Solution:

```
def is_list_empty(arr: list) -> bool:  
    return not bool(arr)  
  
arrays = [[1, 2, 3], ['item'], []]  
  
for arr in arrays:  
    if is_list_empty(arr):  
        print(f'{arr} is a empty list')  
    else:  
        print(f'{arr} contains elements')
```


174. Create the new_arr list which contains elements from two lists. First there should be elements from the arr 1 list then elements from the arr 2 list.

Expected result:

[1, 2, 3, 4, 5]

arr_1 = [1, 2]

arr_2 = [3, 4, 5]

new_arr =

Solution

```
arr_1 = [1, 2]
```

```
arr_2 = [3, 4, 5]
```

```
new_arr = arr_1 + arr_2
```

175. Add all elements from the arr_i list to the arr_2 list.

Expected result:

[1, 2, 3, 4]

arr_1 = [3, 4]

arr_2 = [1, 2]

Solution

```
arr_1 = [3, 4]
```

```
arr_2 = [1, 2]
```

```
arr_2.extend(arr_1)
```

176. Create a copy of the arr list and save it in the new_arr variable (shallow copy).

```
arr = [1, 2]
```

```
new_arr =
```

Solution

```
arr = [1, 2]
```

```
new_arr = arr.copy()
```

```
# or
```

```
# new_arr = list(arr)
```

177. Create a copy of the arr list and save it in the new_arr variable (deep copy).

```
arr = [1, [2]]
```

```
new_arr =
```

Solution

```
import copy
```

```
arr = [1, [2]]
```

```
new_arr = copy.deepcopy(arr)
```


178. Get the number of elements in the arr list. Save the result in the arr_length variable.

Expected result:

4

```
arr = [1, 2, 3, 4]
```

```
arr_length =
```

Solution

```
arr = [1, 2, 3, 4]
```

```
arr_length = len(arr)
```

179. Count the occurrences of the digit 2 in the arr list. Save the result in the count variable.

Expected result:

3

```
arr = [1, 2, 3, 4, 2, 5, 5, 2]
```

```
count =
```

Solution

```
arr = [1, 2, 3, 4, 2, 5, 5, 2]
```

```
count = arr.count(2)
```

180. Remove the last element from the arr list and save its result to the last item variable.

```
arr = [1, 2, 3]
```

```
last_item =
```

Solution

```
arr = [1, 2, 3]
```

```
last_item = arr.pop()
```

list. 181. Remove the element with the value 2 from the arr

Expected result:

[1, 3, 4]

arr = [1, 2, 3, 4]

Solution

```
arr = [1, 2, 3, 4]
```

```
arr.pop(1)
```


182. Sort the elements of the arr list in descending order.

Expected result:

[9, 6, 4, 2]

arr = [2, 6, 4, 9]

Solution

```
arr = [2, 6, 4, 9]
```

```
arr.sort(reverse=True)
```

183. Create the new_arr list which contains only unique elements from the arr list.

Expected result:

[1, 2, 3, 4]

```
arr = [4, 2, 1, 1, 3, 2, 1, 3, 4, 4, 4, 3]
```

```
new_arr =
```

Solution

```
arr = [4, 2, 1, 1, 3, 2, 1, 3, 4, 4, 4, 3]
```

```
new_arr = list(set(arr))
```

184. Remove the elements with the value i and 3 from the arr list.

Expected result:

[4, 2, 2, 4, 4, 4]

arr = [4, 2, 1, 1, 3, 2, 1, 3, 4, 4, 4, 3]

arr =

Solution

arr = [4, 2, 1, 1, 3, 2, 1, 3, 4, 4, 4, 3]

arr = [item for item in arr if item != 1 and item != 3]

185. Create the difference list with the elements from the am_i list which are not present in the arr 2 list.

Expected result:

[1, 2]

arr_1 = [1, 2, 3, 4]

arr_2 = [3, 4, 6]

difference =

Solution

```
arr_1 = [1, 2, 3, 4]
```

```
arr_2 = [3, 4, 6]
```

```
difference = list(set(arr_1) - set(arr_2))
```

```
# or
```

```
# difference = [item for item in arr_1 if item not in  
arr_2]
```


186. Convert the an list to a string. Elements from the list should be separated by comma , and space . Save the result in the names variable.

Expected result:

'Alice, Tom, John'

```
arr = ['Alice', 'Tom', 'John']
```

```
names =
```

Solution

```
arr = ['Alice', 'Tom', 'John']
```

```
names = ', '.join(arr)
```

187. Remove empty strings from the arr list.

Expected result:

['Alice', 'Tom', 'John']

```
arr = ['Alice', '', 'Tom', 'John', '']
```

```
arr =
```

Solution

```
arr = ['Alice', '', 'Tom', 'John', '']
```

```
arr = [item for item in arr if item]
```

```
# or
```

```
# list(filter(bool, arr))
```

188. Find the smallest number in the array list. Save the result in the smallest number variable.

Expected result:

-60

```
arr = [104, -60, 0, 399, -30, -5]
```

```
smallest_number =
```

Solution

```
arr = [104, -60, 0, 399, -30, -5]
```

```
smallest_number = min(arr)
```

189. Create the `are_elements_true` function which checks if all elements of the given list are True . The function should return True or False.

```
def are_elements_true(arr: list) -> bool:
# TODO
arrays = [[0, True, 1], [22, True], ['a', 'b', 4]]
for arr in arrays:
    if are_elements_true(arr):
        print(fAll items in {arr} are "True")
    else:
        print(fSome items in {arr} are "False")
```

Solution:

```
def are_elements_true(arr: list) -> bool:
```

```
    return all(arr)
```

```
arrays = [[0, True, 1], [22, True], ['a', 'b', 4]]
```

```
for arr in arrays:
```

```
if are_elements_true(arr):
```

```
    print(fAll items in {arr} are "True")
```

```
else:
```

```
    print(fSome items in {arr} are "False")
```


190. In the numbers list find all numbers which are greater than 25 . Save the result in the

new numbers Variable.

Expected result:

[88, 45, 98, 34, 55, 48]

```
numbers = [88, 45, 4, 7, 98, 34, 21, 18, 14, 55, 48, 1]
```

```
new_numbers =
```

Solution

```
numbers = [88, 45, 4, 7, 98, 34, 21, 18, 14, 55, 48, 1]
```

```
new_numbers = [num for num in numbers if num > 25]
```

```
# or
```

```
# new_numbers = list(filter(lambda num: num > 25,  
new_numbers))
```

```
# or
```

```
# def is_greater_than_25(num: int) -> bool:
```

```
#     return num > 25
```

```
    # new_numbers = list(filter(is_greater_than_25,  
new_numbers))
```

191. Find the average of the am list and round it to two decimal places. Save the result in the mean variable.

Expected result:

28.11

```
arr = [6, 4, 56, 34, 23, 78, 1, 34, 17]
```

```
mean =
```

Solution

```
import statistics
```

```
arr = [6, 4, 56, 34, 23, 78, 1, 34, 17]
```

```
mean = round(statistics.mean(arr), 2)
```

192. Find the duplicates in the arr list and create the duplicates list with them.

Expected result:

[' a ', 11, 4, 8]

arr = ['a', 11, 14, 11, 'a', 'a', 'f', 4, 4, 'a', 6, 8, 8, 8]

duplicates =

Solution

```
from collections import Counter
```

```
arr = ['a', 11, 14, 11, 'a', 'a', 'f', 4, 4, 'a', 6, 8, 8, 8]
```

```
duplicates = [key for key, value in Counter(arr).items() if  
value > 1]
```

193. Get all indices from the arr list for the elements which have the value greater than 2. Save the result in the indexes list.

Expected result:

[2, 5, 6, 9]

arr = [5, 4, 2, 7, 6, 2, 2, 1, 1, 2]

indexes =

Solution

```
arr = [5, 4, 2, 7, 6, 2, 2, 1, 1, 2]
```

```
indexes = [index for index, item in enumerate(arr) if item  
           == 2]
```


194. Create the numbers list which contains only even numbers in the range 16-76.

Expected result:

[16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76]

numbers =

Solution:

```
numbers = list(range(16, 77, 2))
```

195. Get list of values from the **names_dict** dictionary and save it in the

Names variable.

Expected result:

['Anna', 'Tom', 'Alice', 'Mark']

```
names_dict = {  
    1: 'Anna',  
    2: 'Tom',  
    3: 'Alice',  
    4: 'Mark'  
}  
names =
```

Solution

```
names_dict = {  
    1: 'Anna',  
    2: 'Tom',  
    3: 'Alice',  
    4: 'Mark'  
}  
  
names = list(names_dict.values())
```

196. Add the value 10 to every element in the numbers list.

Expected result:

[15, 14, 17, 18, 32, 577]

```
numbers = [5, 4, 7, 8, 22, 567]
```

```
numbers =
```

Solution:

```
numbers = [5, 4, 7, 8, 22, 567]
```

```
numbers = [num + 10 for num in numbers]
```

```
# or
```

```
# numbers = list(map(lambda num: num + 10, numbers))
```

197. Create the new_arr list with only unique elements from the new arr list must be the same as in the arr list. The order of elements in

Expected result:

[4, 7, 15, 56, 6, 89, 111]

Solution

```
arr = [4, 7, 15, 4, 4, 56, 6, 6, 89, 15, 111]
```

```
new_arr = []
```

```
for item in arr:
```

```
    if item not in new_arr:
```

```
        new_arr.append(item)
```


198. Make a flat list out of a list of lists. Save the result in the merged_arr variable.

Expected result:

[1, 2, 3, 4, 5, 6, 7, 8, 9]

```
arr = [[1, 2], [3], [4, 5, 6, 7], [8, 9]]
```

```
merged_arr =
```

Solution

```
import itertools
```

```
arr = [[1, 2], [3], [4, 5, 6, 7], [8, 9]]
```

```
merged_arr = list(itertools.chain.from_iterable(arr))
```

199. Sort the am list based on an age attribute of the objects. The list should be sorted in descending order.

Expected result:

[Dohn - 37, Tom - 34, Alice - 28]

```
class Person:
    def __init__(self, name: str, age: int):
        self.name = name
        self.age = age
    def __repr__(self):
        return f'{self.name} - {self.age}'

p1 = Person('Tom', 34)
p2 = Person('Alice', 28)
p3 = Person('John', 37)
arr = [p1, p2, p3]
arr =
```

Solution

class Person:

def __init__(self, name: str, age: int):

self.name = name

self.age = age

def __repr__(self):

return f'{self.name} - {self.age}'

p1 = Person('Tom', 34)

p2 = Person('Alice', 28)

p3 = Person('John', 37)

arr = [p1, p2, p3]

arr.sort(key=lambda x: x.age, reverse=True)

200. Given 2D array calculate the sum of diagonal elements.

Ex: $[[1,8,9],[1,5,6],[7,6,9]] \Rightarrow \text{sum of } 1 + 5 + 9 \Rightarrow 15$

```
#Edcorner Learning Python Exercises
```

```
a_2d = [[1,2,3],[4,5,6],[11,12,14]]
```

```
b_2d = [[1,0,1],[1,1,0],[1,1,1]]
```

```
c_2d = [[2,0],[0,2]]
```

```
def sum_of_diagonal(a):
```

```
    sum = 0
```

```
    for i in range(len(a)):
```

```
        sum += a[i][i]
```

```
    return sum
```

```
print("sum a_2d: ",sum_of_diagonal(a_2d))
```

```
print("sum b_2d: ",sum_of_diagonal(b_2d))
```

```
print("sum c_2d: ",sum_of_diagonal(c_2d))
```

```
sum a_2d: 20
```

```
sum b_2d: 3
```

```
sum c_2d: 4
```

201. Exercise 3: Given a Python list of numbers. Turn every item of a list into its square

Given:

```
aList = [1, 14, 23, 4, 15, 6, 27]
```

Solution:

```
aList = [1, 14, 23, 4, 15, 6, 27]  
aList = [x * x for x in aList]  
print(aList)
```


ABOUT THE AUTHOR

“Edcorner Learning” and have a significant number of students on **Udemy** with more than **90000 + Student and Rating of 4.1 or above.**

Edcorner Learning is Part of Edcredibly.

Edcredibly is an online eLearning platform provides Courses on all trending technologies that maximizes learning outcomes and career opportunity for professionals and as well as students. Edcredibly have a significant number of 100000+ students on their own platform and have a **Rating of 4.9 on Google Play Store – Edcredibly App.**

Feel Free to check or join our courses on:

Edcredibly Website - <https://www.edcredibly.com/>

**Edcredibly App –
[https://play.google.com/store/apps/details?
id=com.edcredibly.courses](https://play.google.com/store/apps/details?id=com.edcredibly.courses)**

**Edcorner Learning Udemy - [https://www.udemy.com/user/
edcorner/](https://www.udemy.com/user/edcorner/)**

Do check our other eBooks available on Kindle Store.