Python programming for beginners

Stefan Zhauryd
Instructor

Module 3

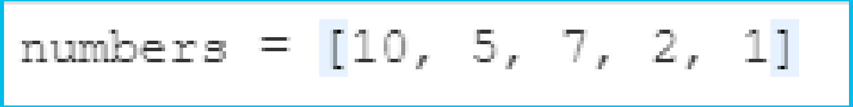
Boolean Values, Conditional Execution, Loops, Lists and List Processing, Logical and Bitwise Operations

In this module, you will learn about:

- lists in Python (constructing, indexing, and slicing; content manipulation)
- how to sort a list using bubble-sort algorithms;
- multidimensional lists and their applications.

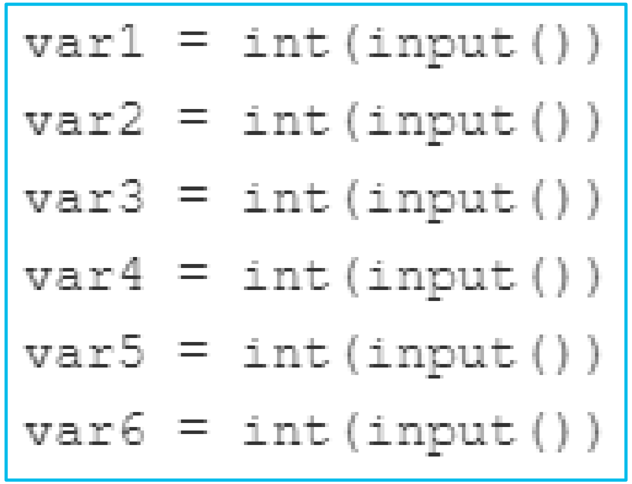
Why do we need lists?

[ ] through ,(comma)

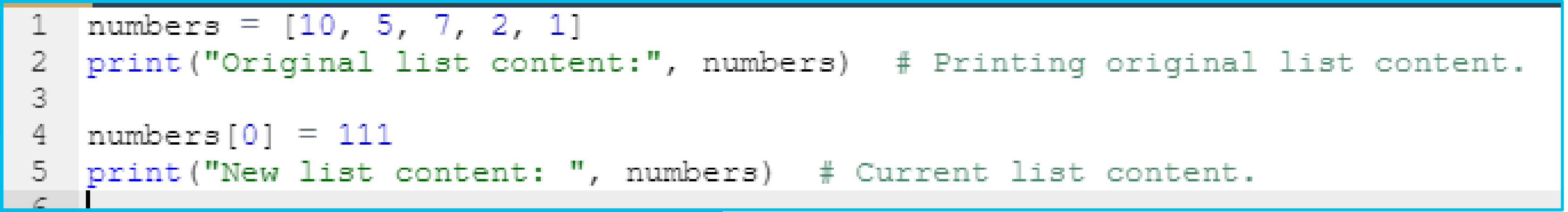


```
numbers = [10, 5, 7, 2, 1]
```

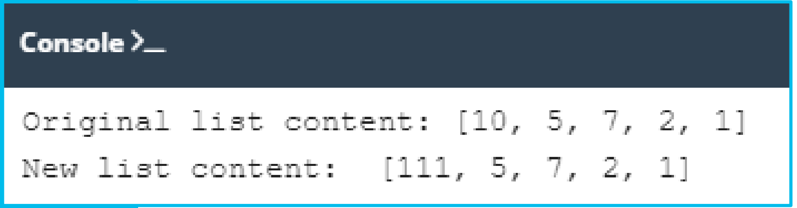
- the elements in a list are always numbered starting from zero
- may have different types
- list starts with an open square bracket and ends with a closed square bracket



```
var1 = int(input())
var2 = int(input())
var3 = int(input())
var4 = int(input())
var5 = int(input())
var6 = int(input())
```

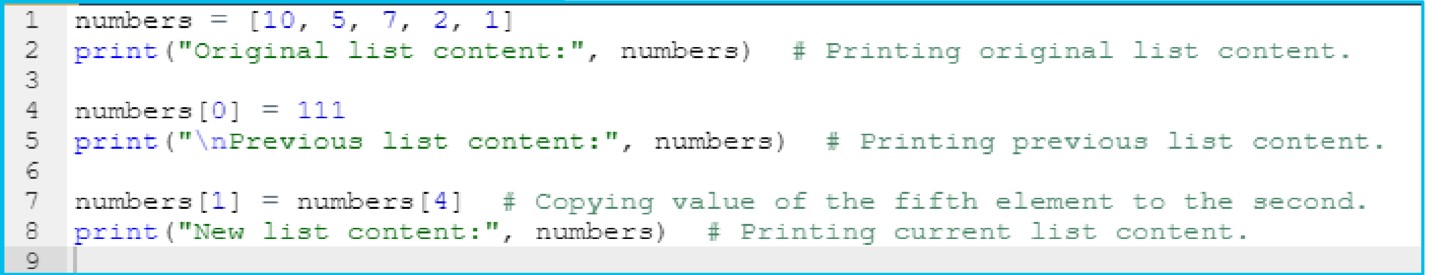


```python
numbers = [10, 5, 7, 2, 1]
print("Original list content:", numbers)  # Printing original list content.

numbers[0] = 111
print("New list content: ", numbers)  # Current list content.
```

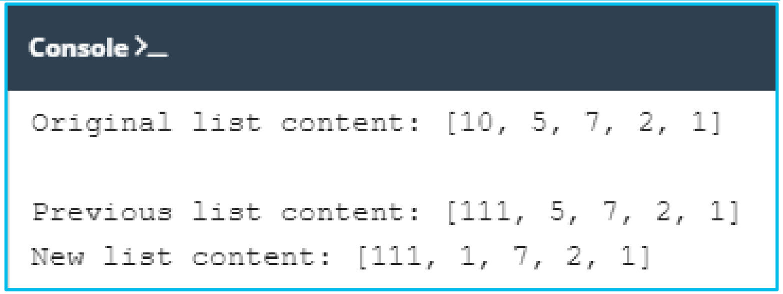


```
Original list content: [10, 5, 7, 2, 1]
New list content:  [111, 5, 7, 2, 1]
```

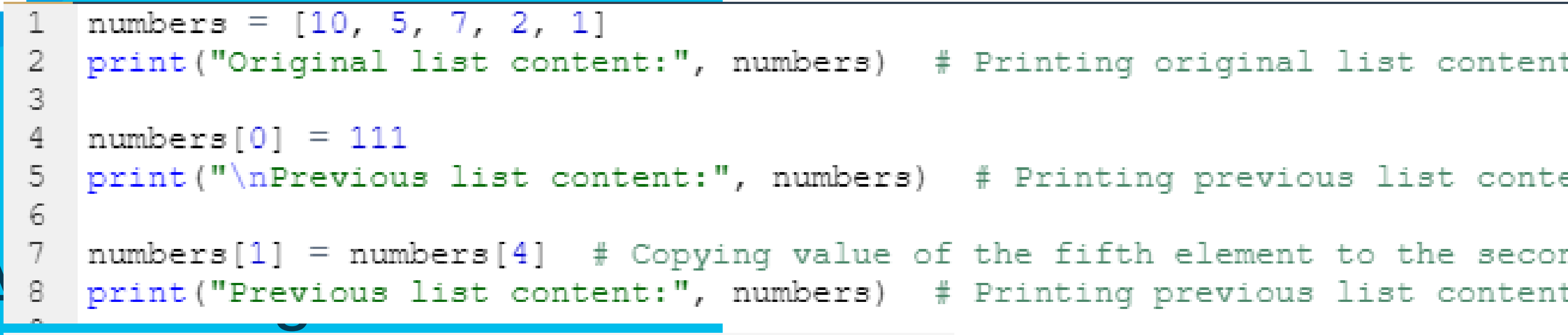
Indexing lists



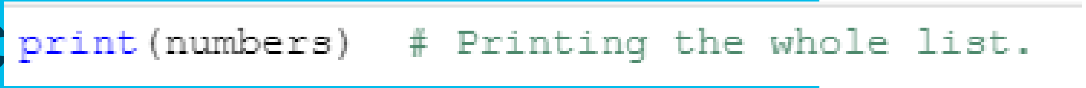
```python
numbers = [10, 5, 7, 2, 1]
print("Original list content:", numbers)  # Printing original list content.

numbers[0] = 111
print("\nPrevious list content:", numbers)  # Printing previous list content.

numbers[1] = numbers[4]  # Copying value of the fifth element to the second.
print("New list content:", numbers)  # Printing current list content.
```



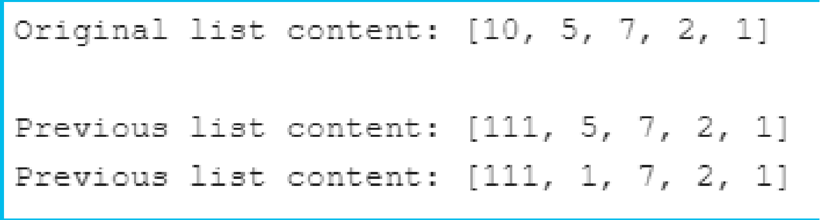
```
Original list content: [10, 5, 7, 2, 1]

Previous list content: [111, 5, 7, 2, 1]
New list content: [111, 1, 7, 2, 1]
```



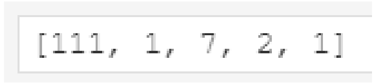
```python
numbers = [10, 5, 7, 2, 1]
print("Original list content:", numbers)  # Printing original list content
 
numbers[0] = 111
print("\nPrevious list content:", numbers)  # Printing previous list conte
 
numbers[1] = numbers[4]  # Copying value of the fifth element to the secon
print("Previous list content:", numbers)  # Printing previous list content
```



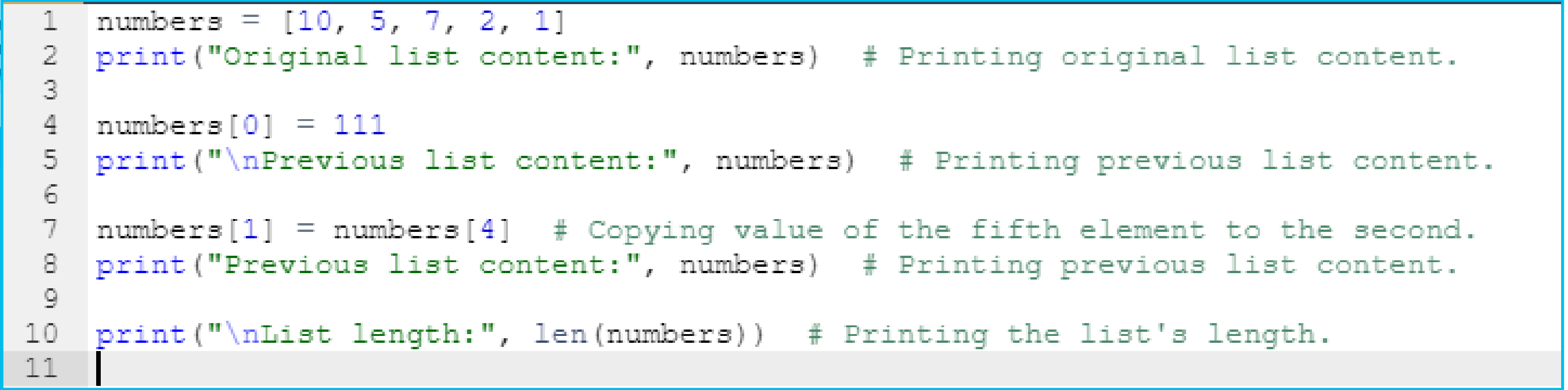
```python
print(numbers)  # Printing the whole list.
```



```
Original list content: [10, 5, 7, 2, 1]


Previous list content: [111, 5, 7, 2, 1]
Previous list content: [111, 1, 7, 2, 1]
```

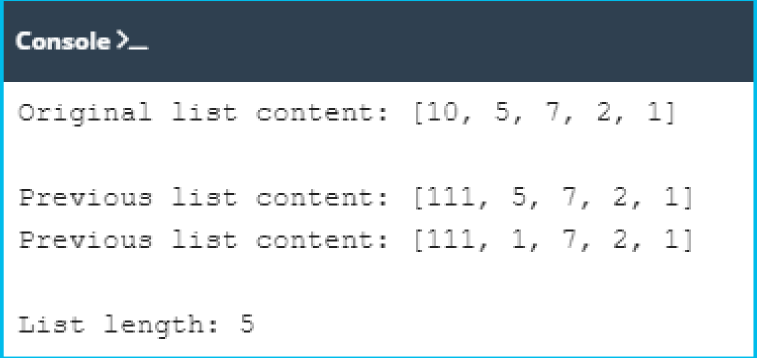


```
[111, 1, 7, 2, 1]
```



```python
numbers = [10, 5, 7, 2, 1]
print("Original list content:", numbers)  # Printing original list content.

numbers[0] = 111
print("\nPrevious list content:", numbers)  # Printing previous list content.

numbers[1] = numbers[4]  # Copying value of the fifth element to the second.
print("Previous list content:", numbers)  # Printing previous list content.

print("\nList length:", len(numbers))  # Printing the list's length.

```

The **len()** function



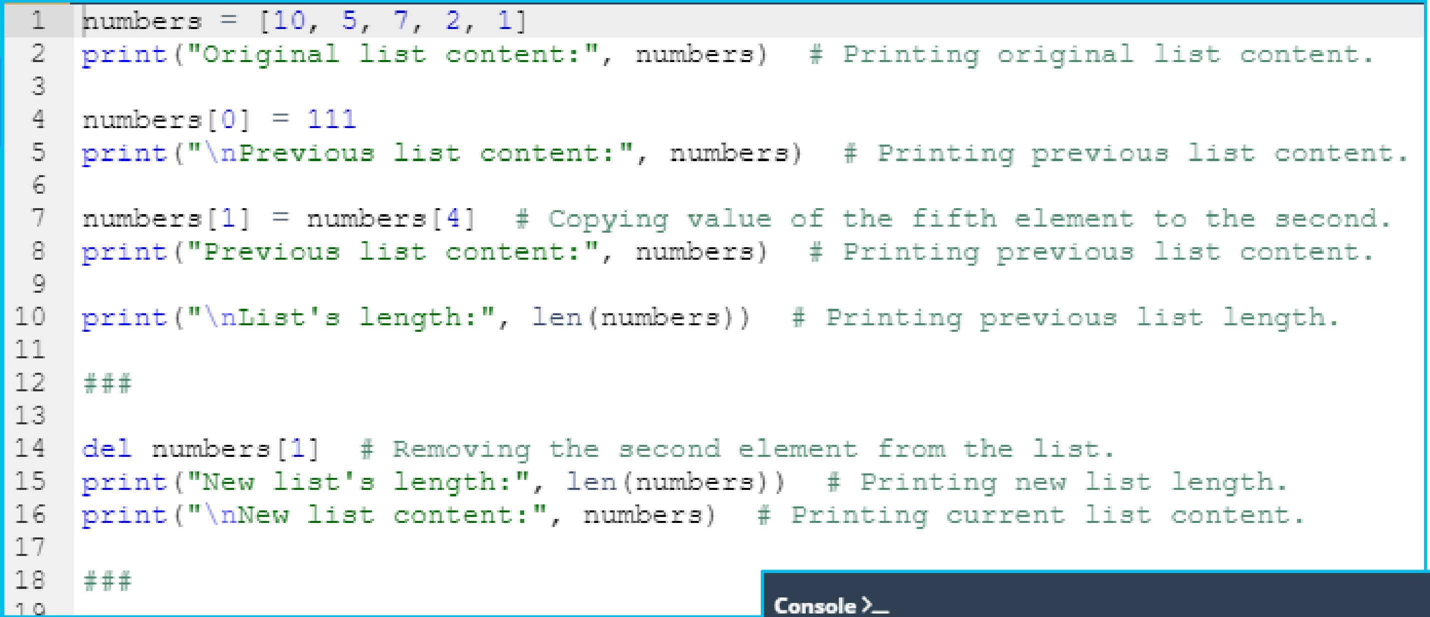
```
Original list content: [10, 5, 7, 2, 1]

Previous list content: [111, 5, 7, 2, 1]
Previous list content: [111, 1, 7, 2, 1]

List length: 5
```

len(numbers) == 5

0 1 2 3 4

numbers[**len**(numbers) - 1]

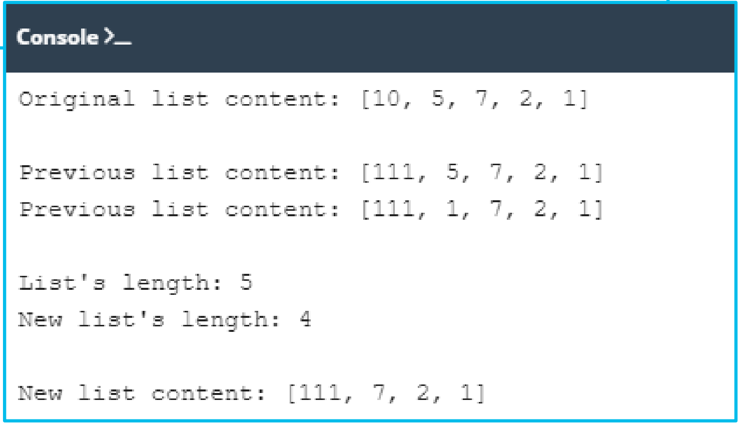
4



```python
numbers = [10, 5, 7, 2, 1]
print("Original list content:", numbers)  # Printing original list content.

numbers[0] = 111
print("\nPrevious list content:", numbers)  # Printing previous list content.

numbers[1] = numbers[4]  # Copying value of the fifth element to the second.
print("Previous list content:", numbers)  # Printing previous list content.

print("\nList's length:", len(numbers))  # Printing previous list length.

###

del numbers[1]  # Removing the second element from the list.
print("New list's length:", len(numbers))  # Printing new list length.
print("\nNew list content:", numbers)  # Printing current list content.

###
```

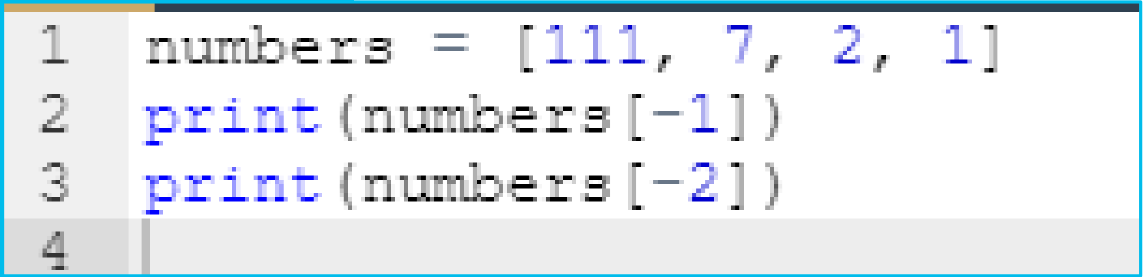
Removing elements from a list

```
del numbers[1]
```

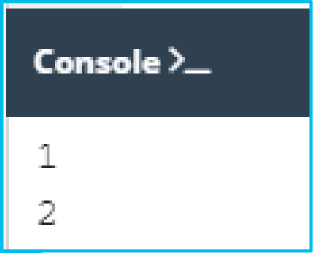


Console >_

```
Original list content: [10, 5, 7, 2, 1]

Previous list content: [111, 5, 7, 2, 1]
Previous list content: [111, 1, 7, 2, 1]

List's length: 5
New list's length: 4

New list content: [111, 7, 2, 1]
```



```python
numbers = [111, 7, 2, 1]
print(numbers[-1])
print(numbers[-2])
```



```
1
2
```

Negative indices are legal

0 1 2 3
-1 -2 -3 -4

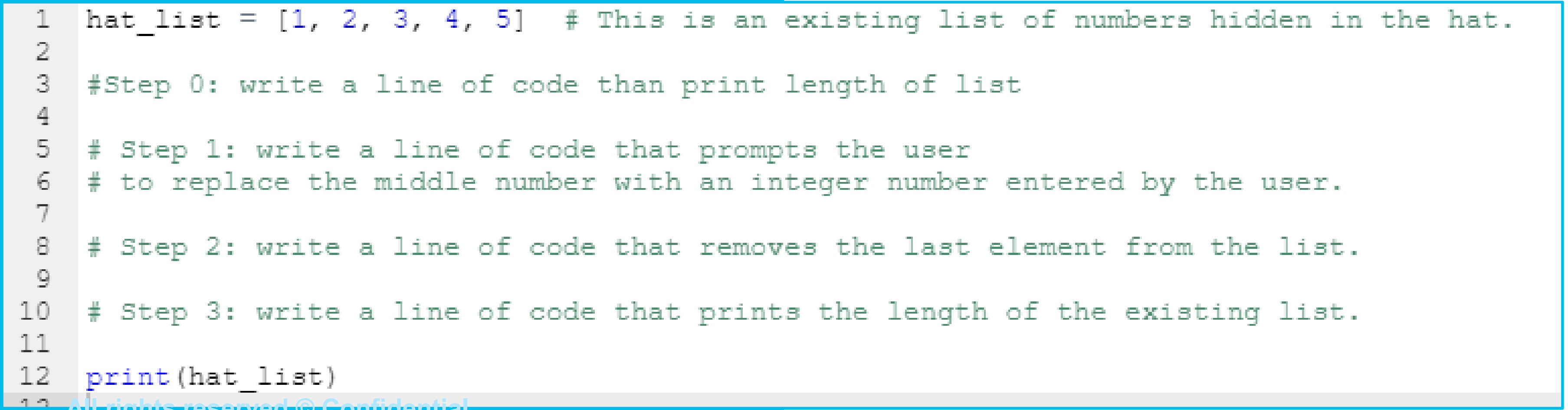
numbers[len(numbers) - 1]

numbers[-1]

Home work 3.1

Your task is to:

- write a line of code that prompts the user to replace the middle number in the list with an integer number entered by the user (Step 1)
- write a line of code that removes the last element from the list (Step 2)
- write a line of code that prints the length of the existing list (Step 3).



```
hat_list = [1, 2, 3, 4, 5]  # This is an existing list of numbers hidden in the hat.

#Step 0: write a line of code than print length of list

# Step 1: write a line of code that prompts the user
# to replace the middle number with an integer number entered by the user.

# Step 2: write a line of code that removes the last element from the list.

# Step 3: write a line of code that prints the length of the existing list.

print(hat_list)
```

Functions vs. methods

```
print("  ", 1, 2, 3, 3)
 len(numbers)

str1 = " ffdsfs   dfs  "
str1.rstrip()
```

A method is a specific kind of function - it behaves like a function and looks like a function, but differs in the way in which it acts, and in its invocation style.

A function doesn't belong to any data - it gets data, it may create new data and it (generally) produces a result.

A method does all these things, but is also able to change the state of a selected entity.

A method is owned by the data it works for, while a function is owned by the whole code.

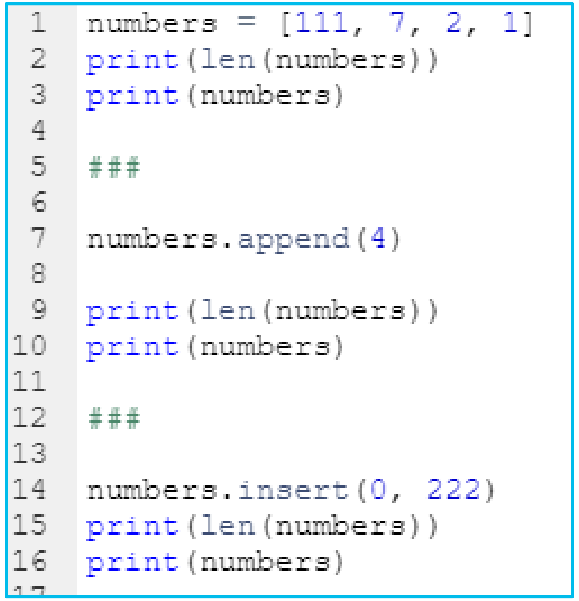
Note: the name of the method is preceded by the name of the data which owns the method. Next, you add a dot, followed by the method name, and a pair of parenthesis enclosing the arguments.

The method will behave like a function, but can do something more - it can change the internal state of the data from which it has been invoked.

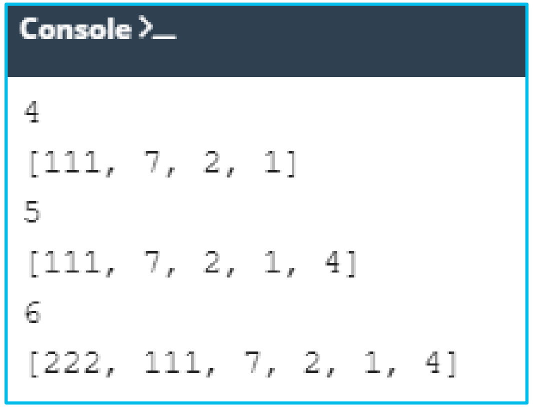
Adding elements to a list: append() and insert()

```
list.append(value)
```

```
list.insert(location, value)
```

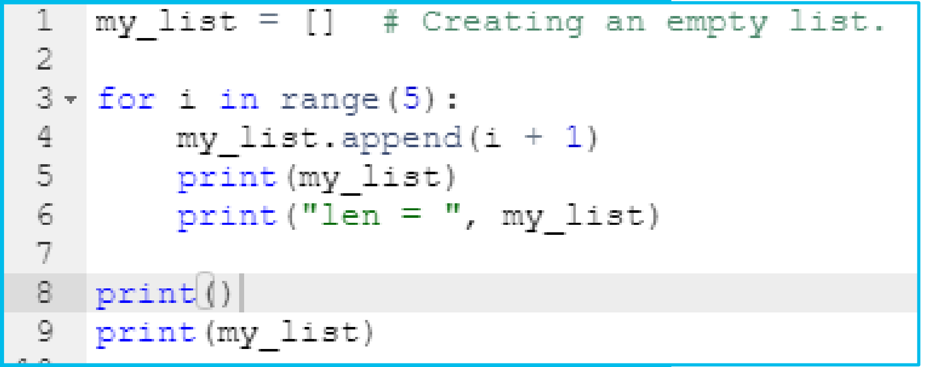


```
numbers = [111, 7, 2, 1]
print(len(numbers))
print(numbers)

###

numbers.append(4)

print(len(numbers))
print(numbers)

###

numbers.insert(0, 222)
print(len(numbers))
print(numbers)
```

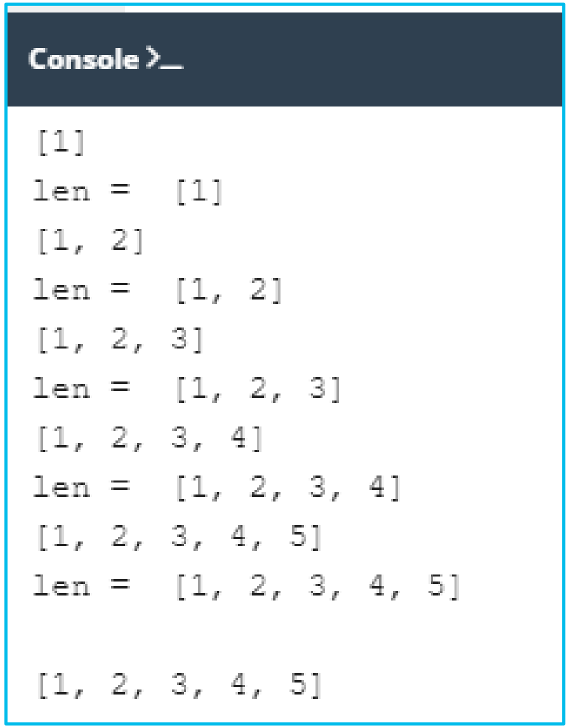


Console >_

```
4
[111, 7, 2, 1]
5
[111, 7, 2, 1, 4]
6
[222, 111, 7, 2, 1, 4]
```



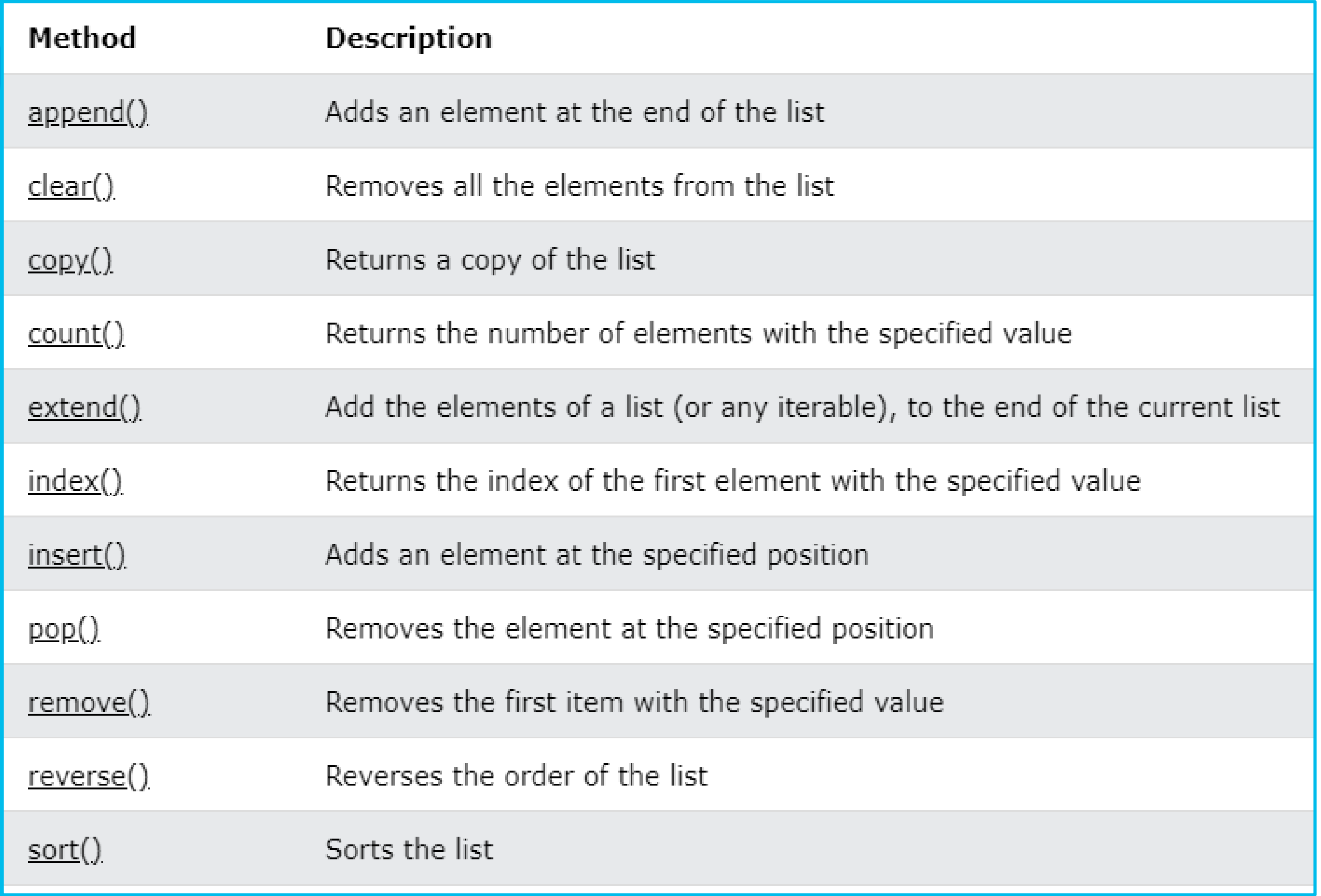
```python
my_list = []   # Creating an empty list.

for i in range(5):
    my_list.append(i + 1)
    print(my_list)
    print("len = ", my_list)

print()
print(my_list)
```



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

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

Adding elements to a list: continued

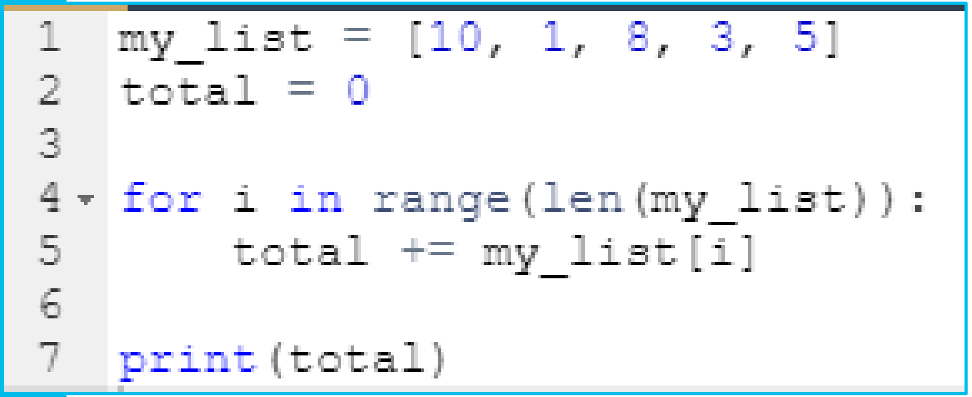
.insert(loc, val)



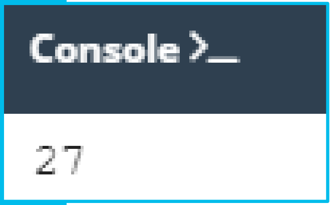
Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

https://www.w3schools.com/python/python_lists_methods.asp

Making use of lists
range(5)
0-4



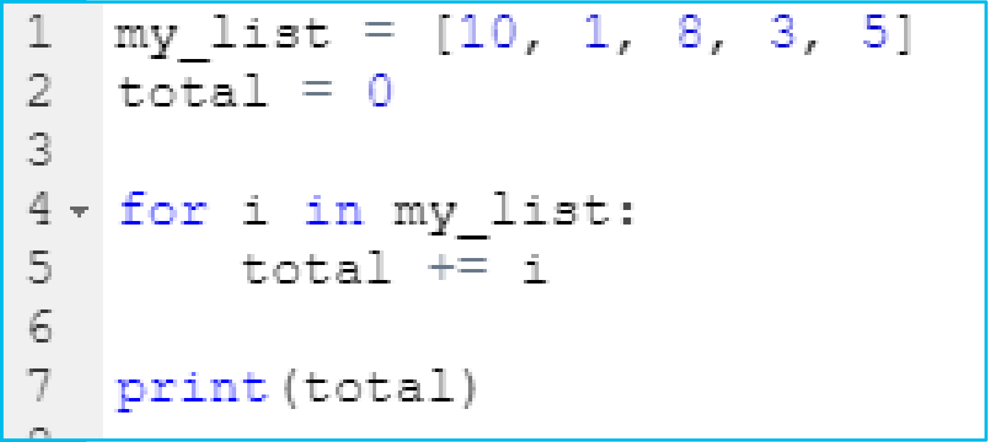
```python
my_list = [10, 1, 8, 3, 5]
total = 0

for i in range(len(my_list)):
    total += my_list[i]

print(total)
```



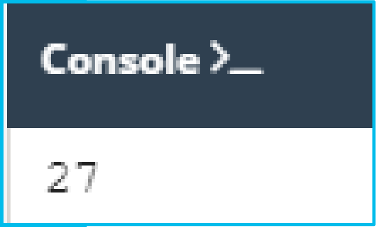
```
27
```

The second face of the for loop

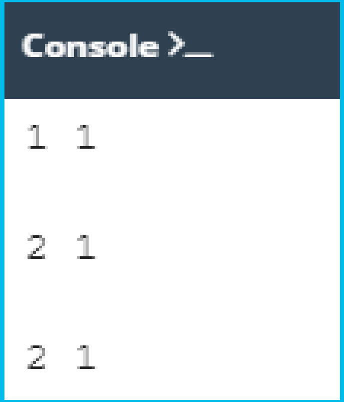
a = [1, 2]
b = a
id(a[1])
id(b[1])



```
my_list = [10, 1, 8, 3, 5]
total = 0

for i in my_list:
    total += i

print(total)
```

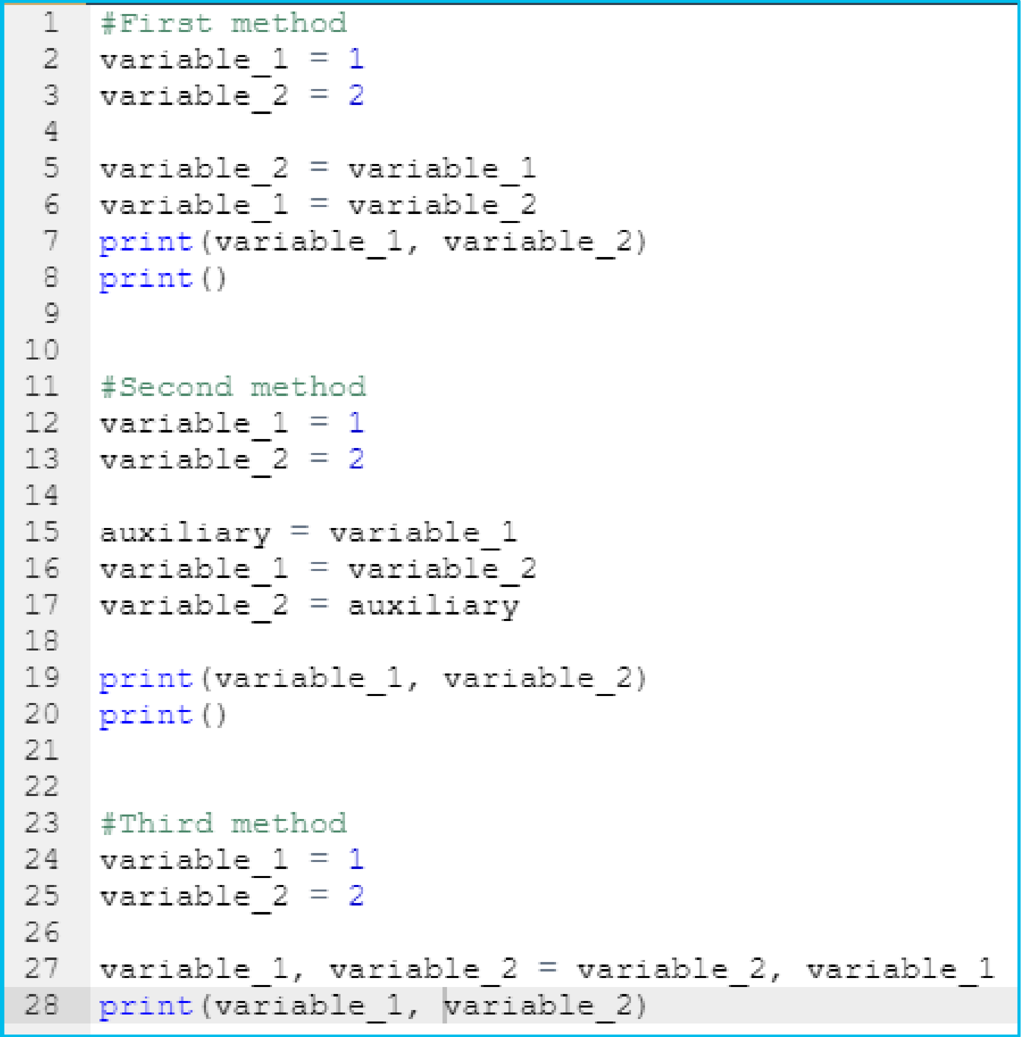


```
27
```

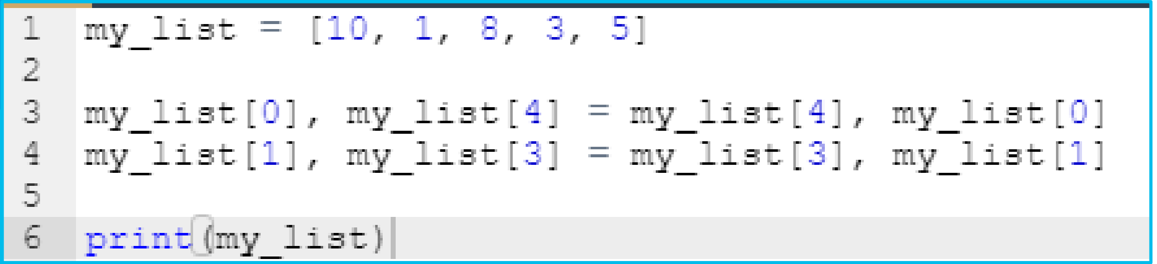


```
1 1

2 1

2 1
```

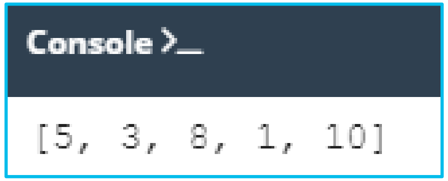
Lists in action
1 2 3



```
#First method
variable_1 = 1
variable_2 = 2

variable_2 = variable_1
variable_1 = variable_2
print(variable_1, variable_2)
print()


#Second method
variable_1 = 1
variable_2 = 2

auxiliary = variable_1
variable_1 = variable_2
variable_2 = auxiliary

print(variable_1, variable_2)
print()


#Third method
variable_1 = 1
variable_2 = 2

variable_1, variable_2 = variable_2, variable_1
print(variable_1, variable_2)
```

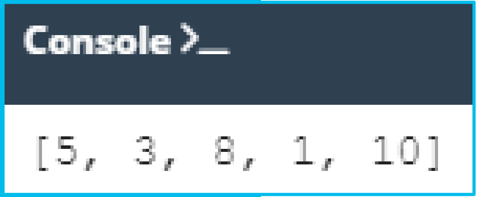


```
my_list = [10, 1, 8, 3, 5]

my_list[0], my_list[4] = my_list[4], my_list[0]
my_list[1], my_list[3] = my_list[3], my_list[1]

print(my_list)
```

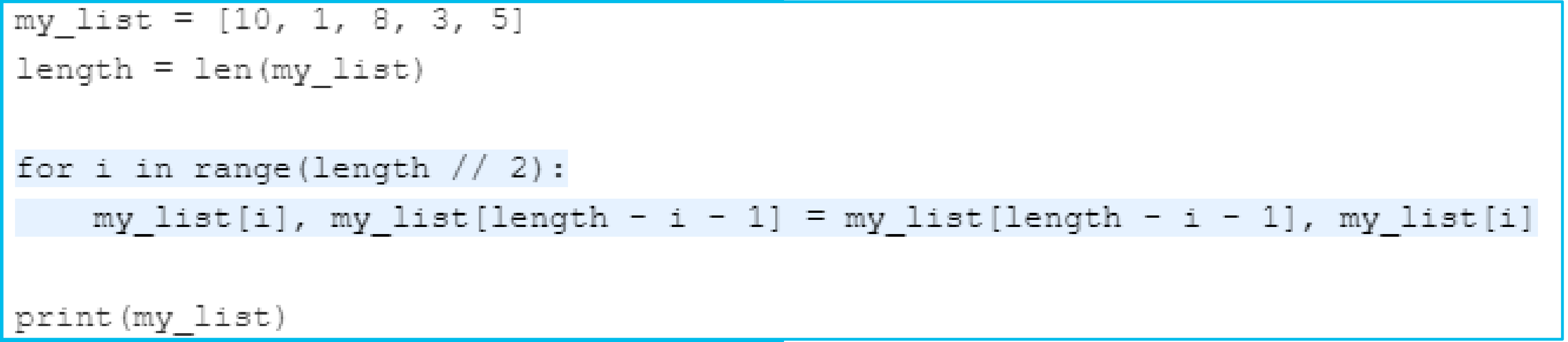


```
[5, 3, 8, 1, 10]
```

!Lists in action
***

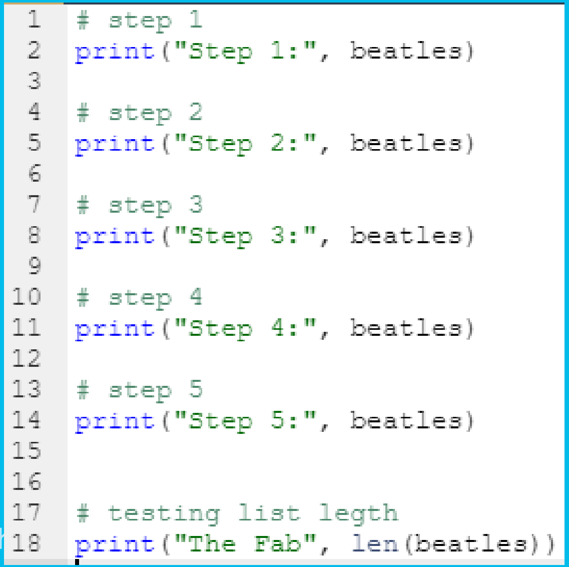


```
[5, 3, 8, 1, 10]
```



```
my_list = [10, 1, 8, 3, 5]
length = len(my_list)


for i in range(length // 2):
    my_list[i], my_list[length - i - 1] = my_list[length - i - 1], my_list[i]


print(my_list)
```

Home work 3.2



```
# step 1
print("Step 1:", beatles)

# step 2
print("Step 2:", beatles)

# step 3
print("Step 3:", beatles)

# step 4
print("Step 4:", beatles)

# step 5
print("Step 5:", beatles)


# testing list legth
print("The Fab", len(beatles))
```

The Beatles were one of the most popular music group of the 1960s, and the best-selling band in history. Some people consider them to be the most influential act of the rock era. Indeed, they were included in Time magazine's compilation of the 20th Century's 100 most influential people.

The band underwent many line-up changes, culminating in 1962 with the line-up of John Lennon, Paul McCartney, George Harrison, and Richard Starkey (better known as Ringo Starr).

Write a program that reflects these changes and lets you practice with the concept of lists. Your task is to:

step 1: **create an empty list named beatles**;

step 2: use the **append()** method to add the following members of the band to the list: **John Lennon**, **Paul McCartney**, and **George Harrison**;

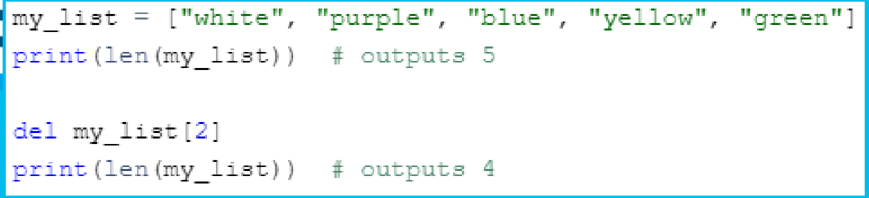
step 3: use the **for** loop and the **append()** method **to prompt the user to add** the following members of the band to the list: **Stu Sutcliffe**, and **Pete Best**; input()

step 4: use the **del** instruction to remove **Stu Sutcliffe** and **Pete Best** from the list;

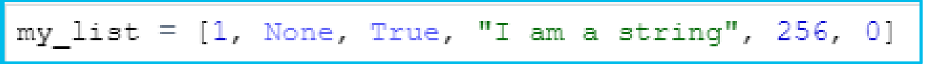
step 5: use the **insert()** method to **add Ringo Starr** to the **beginning** of the list.

By the way, are you a Beatles fan? (The Beatles is one of Greg's favorite bands. But wait...who's Greg...?)

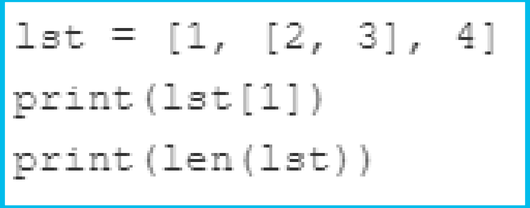
Examples



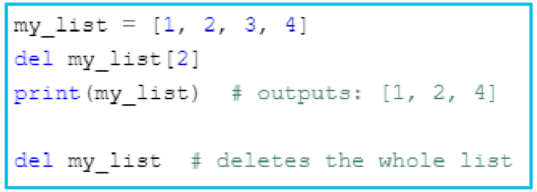
```
my_list = ["white", "purple", "blue", "yellow", "green"]
print(len(my_list))  # outputs 5

del my_list[2]
print(len(my_list))  # outputs 4
```



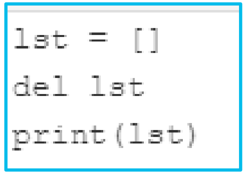
```
my_list = [1, None, True, "I am a string", 256, 0]
```



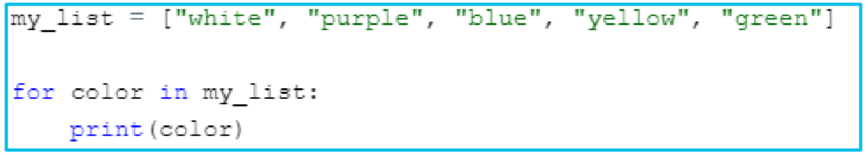
```
lst = [1, [2, 3], 4]
print(lst[1])
print(len(lst))
```



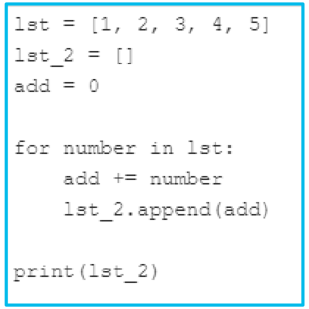
```
my_list = [1, 2, 3, 4]
del my_list[2]
print(my_list)  # outputs: [1, 2, 4]

del my_list  # deletes the whole list
```



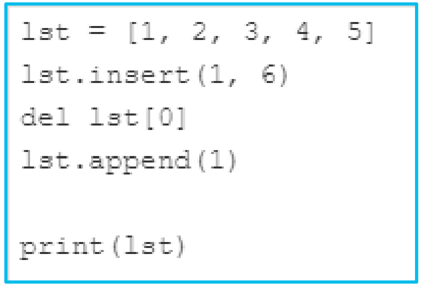
```
lst = []
del lst
print(lst)
```



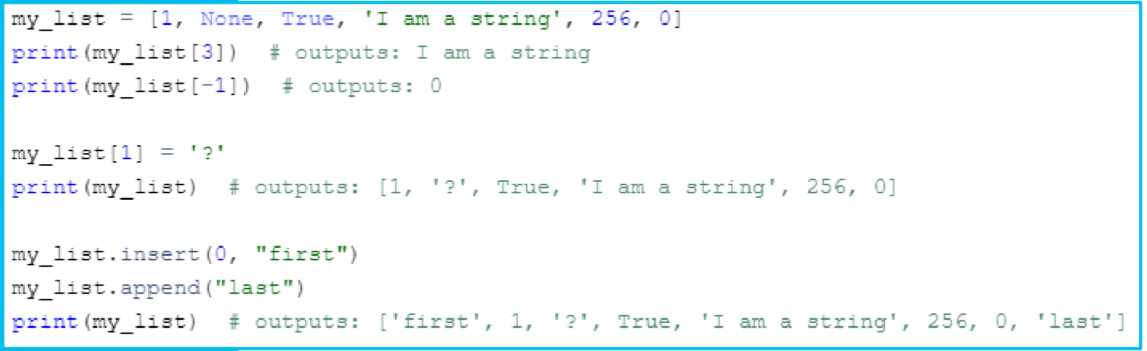
```
my_list = ["white", "purple", "blue", "yellow", "green"]

for color in my_list:
    print(color)
```



```
lst = [1, 2, 3, 4, 5]
lst_2 = []
add = 0

for number in lst:
    add += number
    lst_2.append(add)

print(lst_2)
```



```
lst = [1, 2, 3, 4, 5]
lst.insert(1, 6)
del lst[0]
lst.append(1)

print(lst)
```

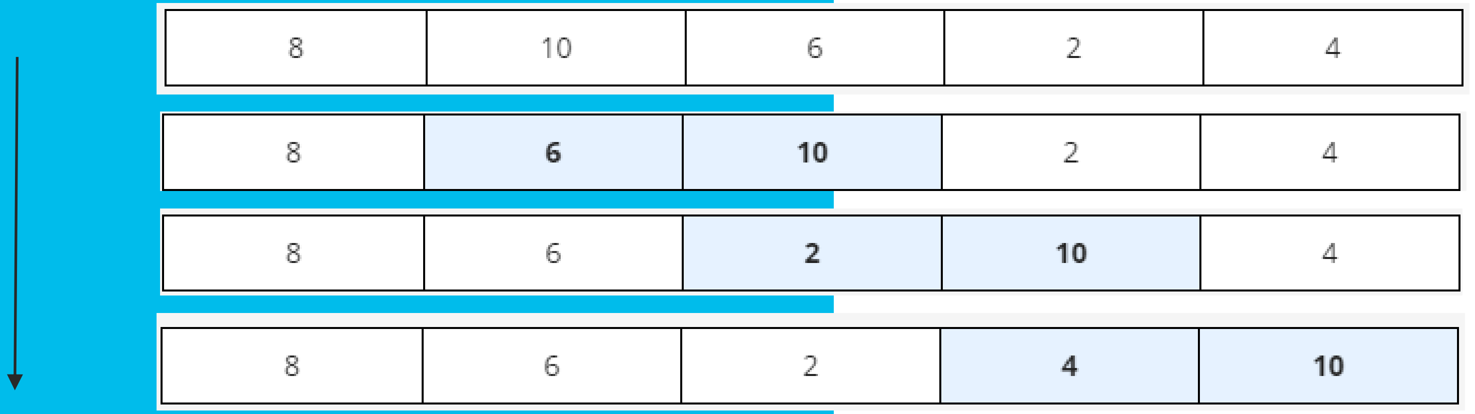


```
my_list = [1, None, True, 'I am a string', 256, 0]
print(my_list[3])  # outputs: I am a string
print(my_list[-1])  # outputs: 0

my_list[1] = '?'
print(my_list)  # outputs: [1, '?', True, 'I am a string', 256, 0]

my_list.insert(0, "first")
my_list.append("last")
print(my_list)  # outputs: ['first', 1, '?', True, 'I am a string', 256, 0, 'last']
```

Key takeaways

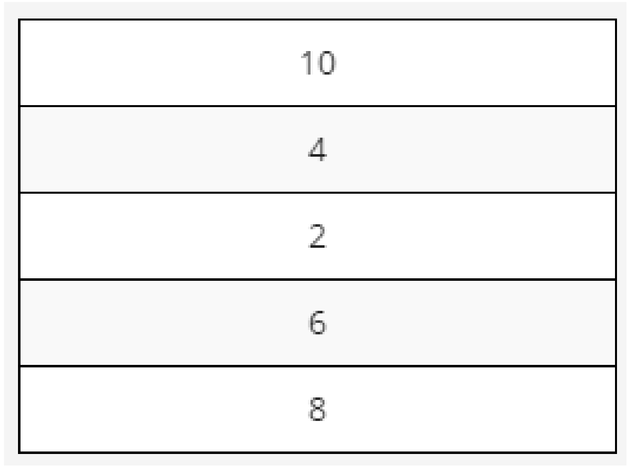
```
li = [1, 2, 3, 4]
li.append(888)
li.append("hfhf")
li.append('0')
[1, 2, ??, 4, 888, ??, '0']
len(li) # fact: ?  indexes: ? - ?

li[-1] is ???
li[0] is ???
li[len(li) - 2]  is ?

del li[2]
[1, 2, 4, 888, "hfhf", '0']
li.insert(position, val)
li.insert(3, 77777)
li[0], li[3] = li[3], li[0]

bar = [1, 2, 3, [3, 4, 5, True], 0]

del li
```

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

for i in li:
    print(i)

l = len(li)
for i in range(l):
    print(li[i])
```



8	10	6	2	4
8	**6**	**10**	2	4
8	6	**2**	**10**	4
8	6	2	**4**	**10**

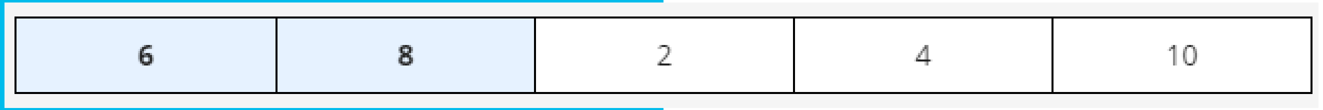
The bubble sort



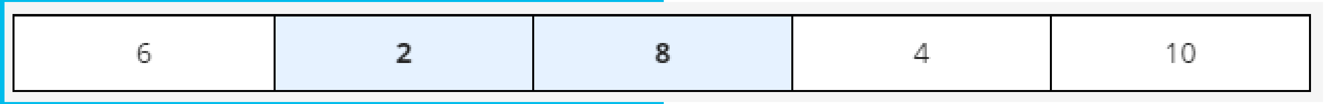
10
4
2
6
8

https://ru.wikipedia.org/wiki/Сортировка_пузырьком

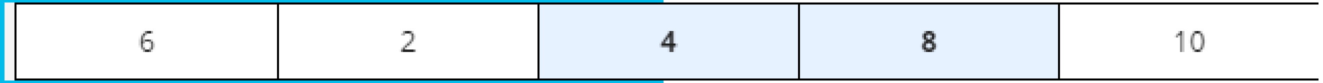
https://en.wikipedia.org/wiki/Bubble_sort



6	**8**	2	4	10

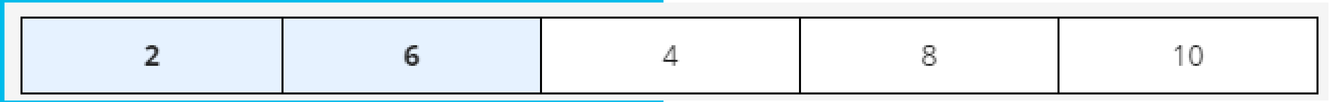


6	**2**	**8**	4	10

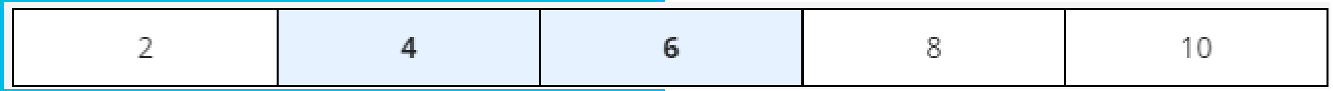


6	2	**4**	**8**	10

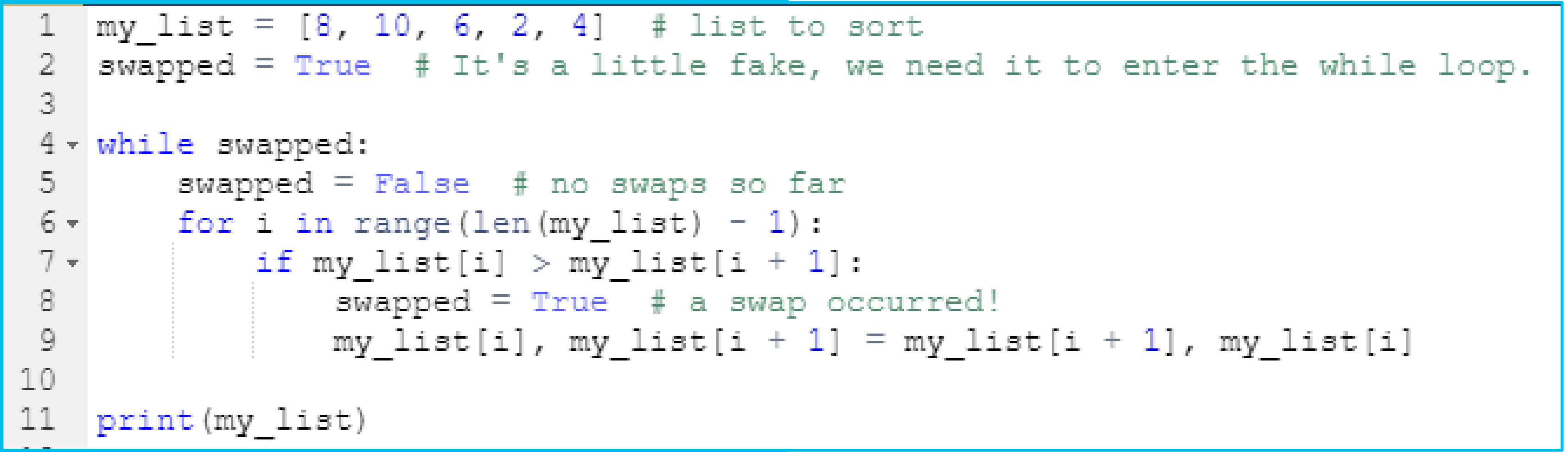
The bubble sort



2	**6**	4	8	10



2	**4**	**6**	8	10



```python
my_list = [8, 10, 6, 2, 4]   # list to sort
swapped = True   # It's a little fake, we need it to enter the while loop.

while swapped:
    swapped = False   # no swaps so far
    for i in range(len(my_list) - 1):
        if my_list[i] > my_list[i + 1]:
            swapped = True   # a swap occurred!
            my_list[i], my_list[i + 1] = my_list[i + 1], my_list[i]

print(my_list)
```

Sorting a list

len = 5   0-4

4   0-3

8 10 6 2 4

1 = 8 6 2 4 10 True

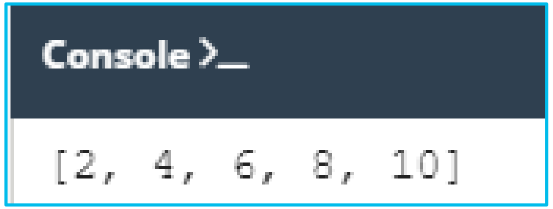
2 = 6 2 4 8 10 True

3 = 2 4 6 8 10 True

4 = False

......

[2 4 6 8 10]



Console >_

```
[2, 4, 6, 8, 10]
```

https://docs.microsoft.com/ru-ru/visualstudio/debugger/debugger-feature-tour?view=vs-2022

Home work 3.3 The bubble sort - interactive version Rev_li = li[::-1]

- as a base use previous slide
- input() int() float() print() and so on
- ***add ability to choose type of sort (**increased**, **reversed**)

n = how many numbers do u want: (n<0)

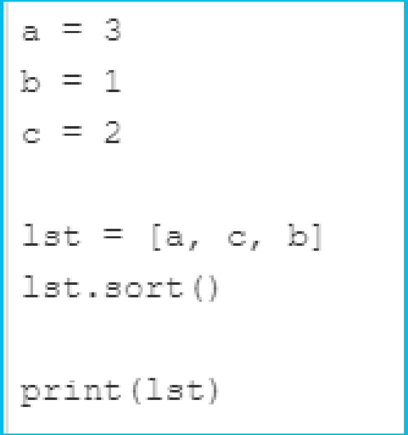
- 5 10 20 35

for i in range(n):

  - input
  - list.append

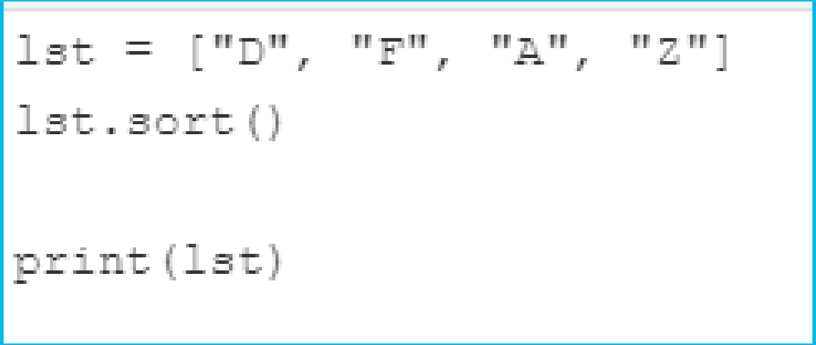
***Also try to use:

- li.sort()
- li.reverse()

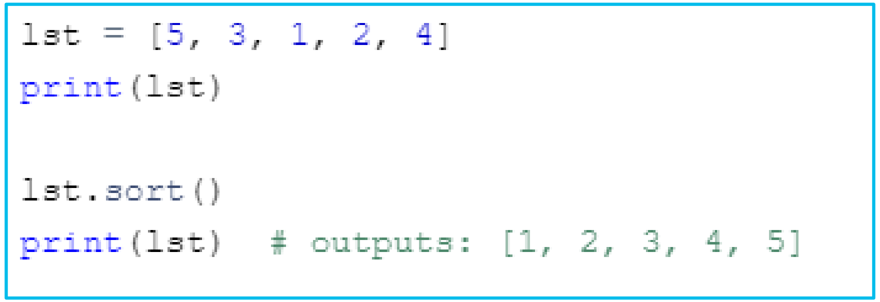


```
a = 3
b = 1
c = 2

lst = [a, c, b]
lst.sort()

print(lst)
```

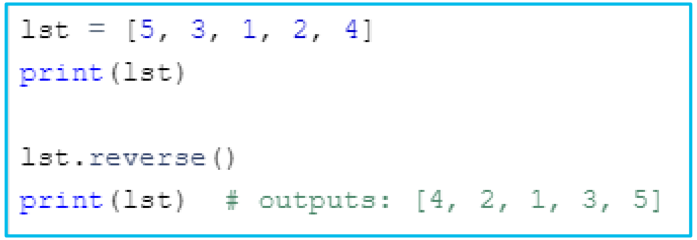
Examples



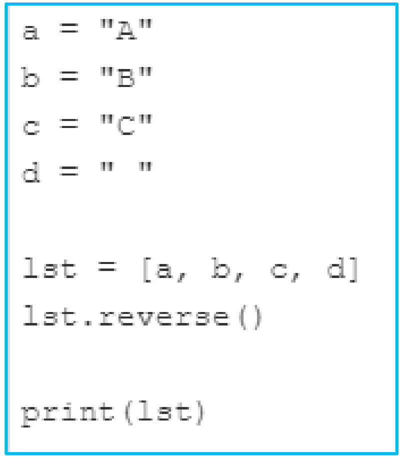
```
lst = ["D", "F", "A", "Z"]
lst.sort()

print(lst)
```



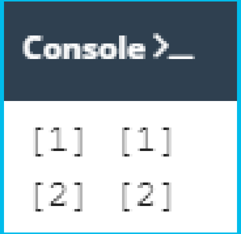
```
lst = [5, 3, 1, 2, 4]
print(lst)

lst.sort()
print(lst)  # outputs: [1, 2, 3, 4, 5]
```



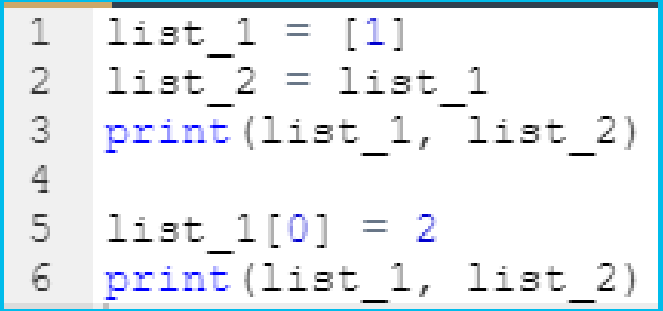
```
lst = [5, 3, 1, 2, 4]
print(lst)

lst.reverse()
print(lst)  # outputs: [4, 2, 1, 3, 5]
```



```
a = "A"
b = "B"
c = "C"
d = " "

lst = [a, b, c, d]
lst.reverse()

print(lst)
```



```
Console >_
[1] [1]
[2] [2]
```



```
list_1 = [1]
list_2 = list_1
print(list_1, list_2)

list_1[0] = 2
print(list_1, list_2)
```

The inner life of lists

a = 1
b = a
b = 666
a b # 1 666

The program:

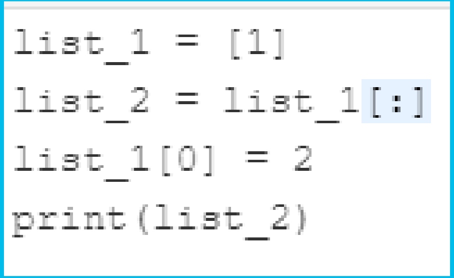
- creates a one-element list named list_1;
- assigns it to a new list named list_2;
- changes the only element of list_1;
- prints out list_2.

You could say that:

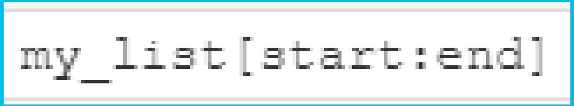
- the name of an ordinary variable is the name of its content;
- the name of a list is the name of a memory location where the list is stored.

The assignment: list_2 = list_1 copies the name of the array, not its contents. In effect, the two names (list_1 and list_2) identify the same location in the computer memory. Modifying one of them affects the other, and vice versa.

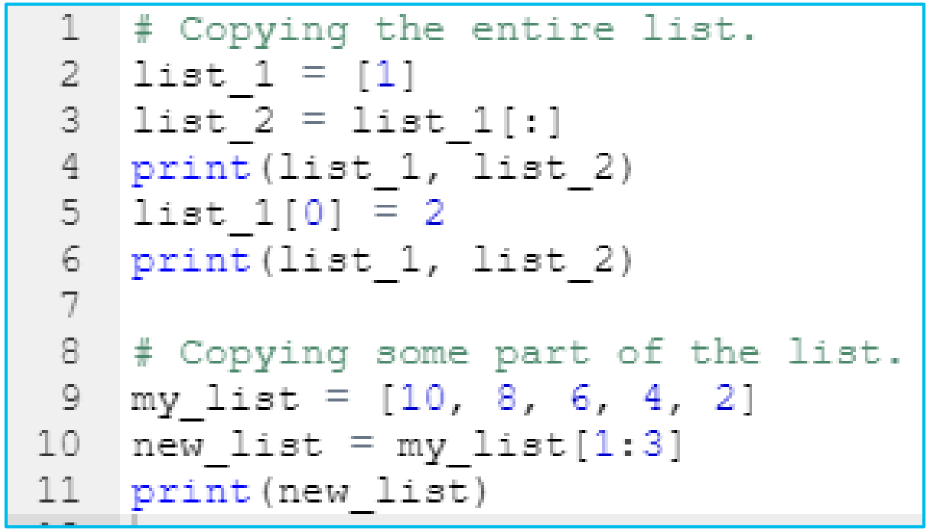
Powerful slices



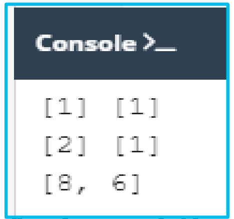
```
list_1 = [1]
list_2 = list_1[:]
list_1[0] = 2
print(list_2)
```



```
my_list[start:end]
```



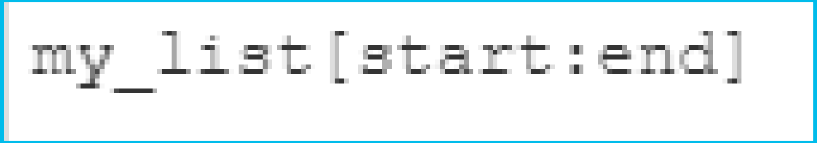
```
# Copying the entire list.
list_1 = [1]
list_2 = list_1[:]
print(list_1, list_2)
list_1[0] = 2
print(list_1, list_2)

# Copying some part of the list.
my_list = [10, 8, 6, 4, 2]
new_list = my_list[1:3]
print(new_list)
```



```
[1] [1]
[2] [1]
[8, 6]
```

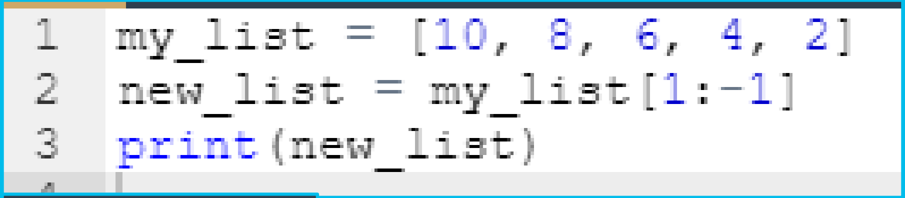
A slice of this form makes a new (target) list, taking elements from the source list - the elements of the indices from start to end - 1.

Note: not to end but to end - 1. An element with an index equal to end is the first element which does not take part in the slicing.

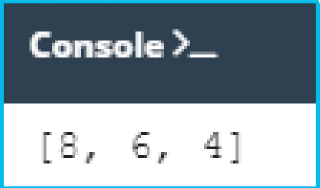


```
my_list[start:end]
```

Slices - negative indices



```
my_list = [10, 8, 6, 4, 2]
new_list = my_list[1:-1]
print(new_list)
```

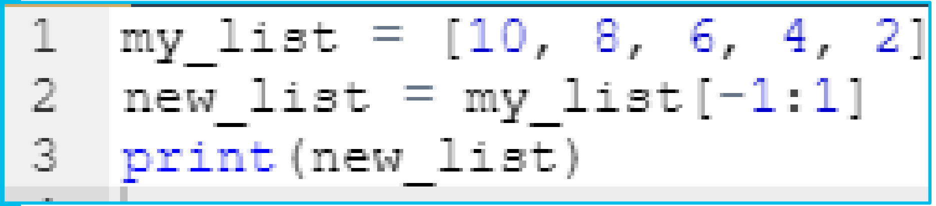


Console >_

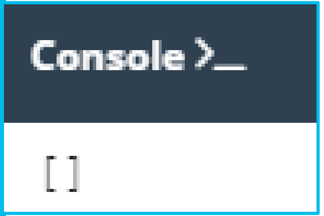
```
[8, 6, 4]
```

To repeat:

- start is the index of the first element included in the slice;
- end is the index of the first element not included in the slice.



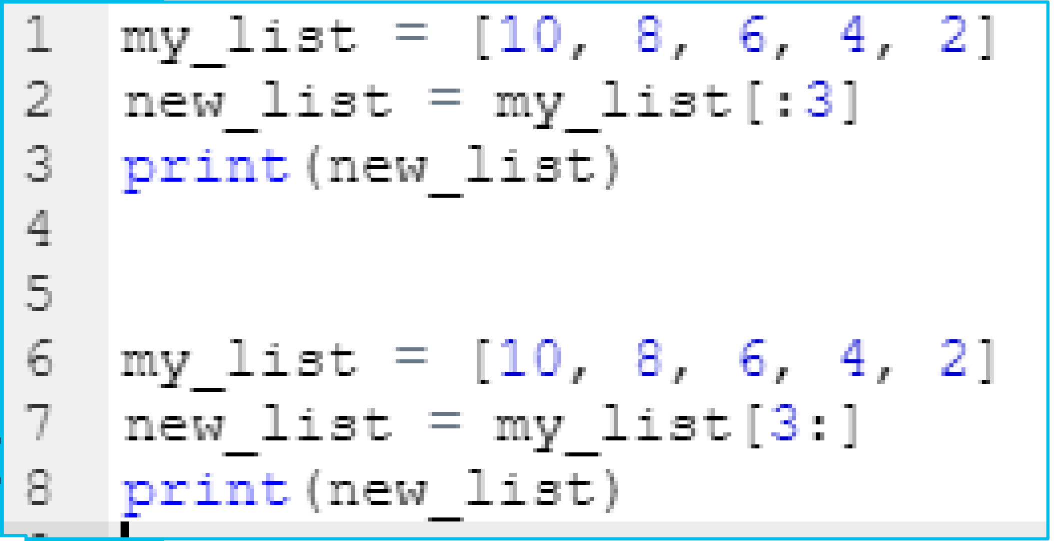
```
my_list = [10, 8, 6, 4, 2]
new_list = my_list[-1:1]
print(new_list)
```



Console >_

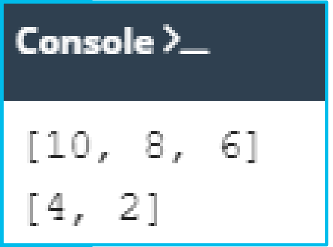
```
[]
```

Slices: continued



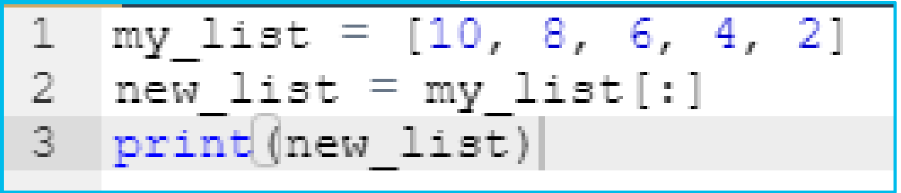
```
my_list = [10, 8, 6, 4, 2]
new_list = my_list[:3]
print(new_list)


my_list = [10, 8, 6, 4, 2]
new_list = my_list[3:]
print(new_list)
```

```
my_list[start:end]
```

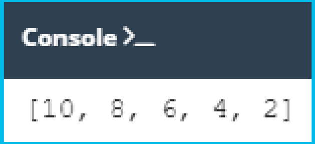


```
[10, 8, 6]
[4, 2]
```

Slices: continued

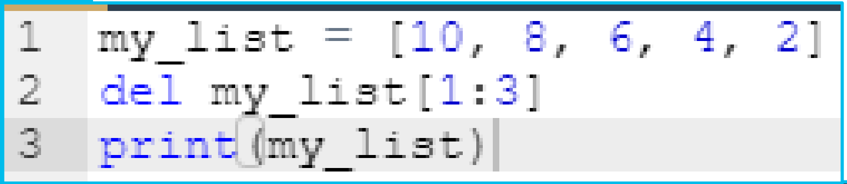


```python
my_list = [10, 8, 6, 4, 2]
new_list = my_list[:]
print(new_list)
```

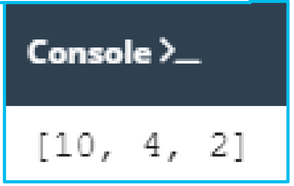


Console >_

```
[10, 8, 6, 4, 2]
```

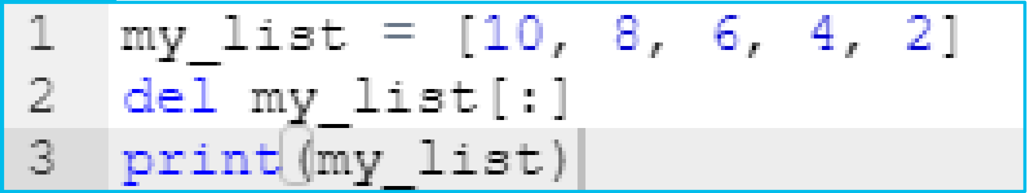


```python
my_list = [10, 8, 6, 4, 2]
del my_list[1:3]
print(my_list)
```

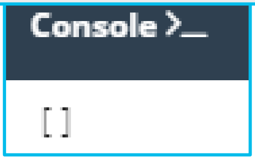


Console >_

```
[10, 4, 2]
```

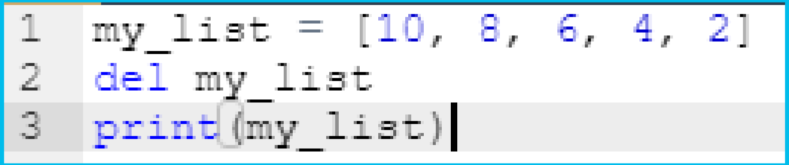


```python
my_list = [10, 8, 6, 4, 2]
del my_list[:]
print(my_list)
```

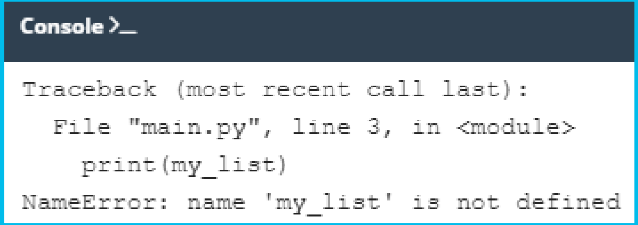


Console >_

```
[]
```



```python
my_list = [10, 8, 6, 4, 2]
del my_list
print(my_list)
```

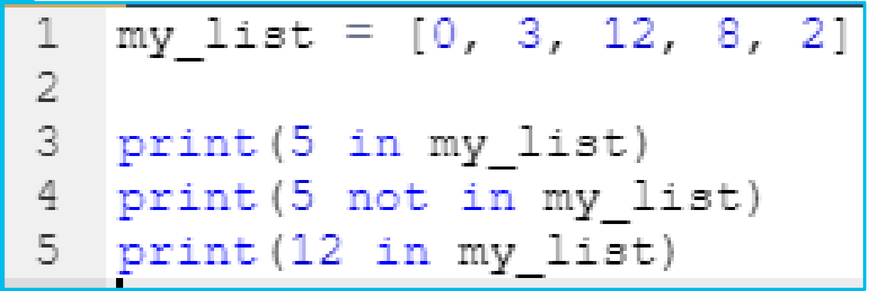


Console >_

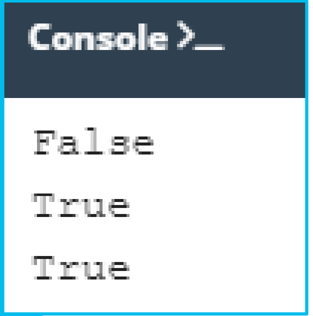
```
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    print(my_list)
NameError: name 'my_list' is not defined
```

The **in** and **not in** operators

Python offers two very powerful operators, able to look through the list in order to check whether a specific value is stored inside the list or not.

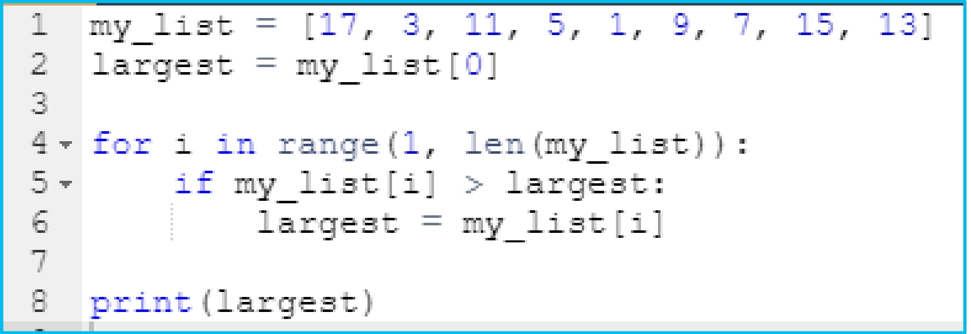


```python
my_list = [0, 3, 12, 8, 2]

print(5 in my_list)
print(5 not in my_list)
print(12 in my_list)
```

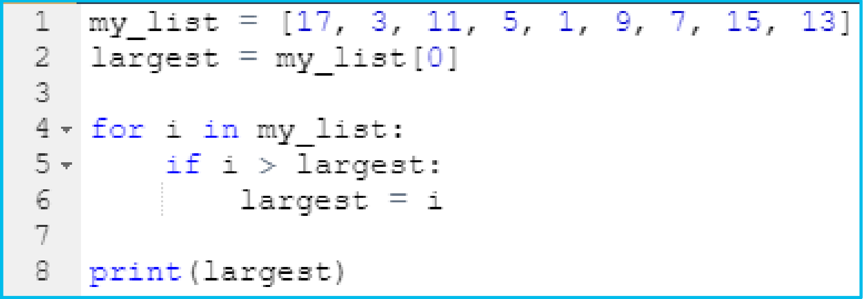


```
False
True
True
```

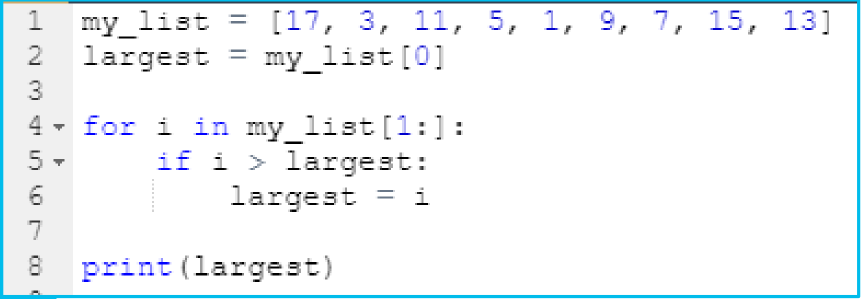
Lists - some simple programs



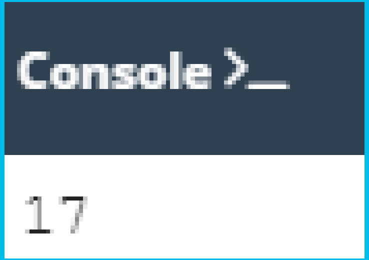
```
my_list = [17, 3, 11, 5, 1, 9, 7, 15, 13]
largest = my_list[0]

for i in range(1, len(my_list)):
    if my_list[i] > largest:
        largest = my_list[i]

print(largest)
```



```
my_list = [17, 3, 11, 5, 1, 9, 7, 15, 13]
largest = my_list[0]

for i in my_list:
    if i > largest:
        largest = i

print(largest)
```

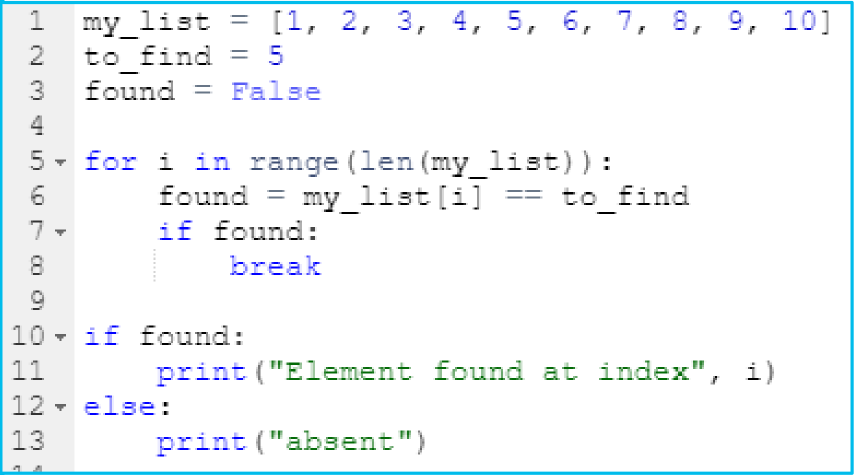


```
my_list = [17, 3, 11, 5, 1, 9, 7, 15, 13]
largest = my_list[0]

for i in my_list[1:]:
    if i > largest:
        largest = i

print(largest)
```

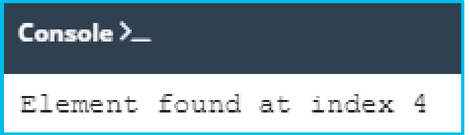


```
17
```

Lists - some simple programs

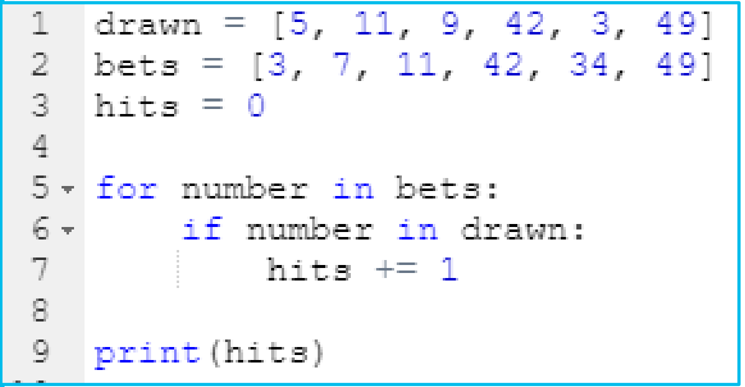


```python
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
to_find = 5
found = False

for i in range(len(my_list)):
    found = my_list[i] == to_find
    if found:
        break

if found:
    print("Element found at index", i)
else:
    print("absent")
```



Console >_

```
Element found at index 4
```



```python
drawn = [5, 11, 9, 42, 3, 49]
bets = [3, 7, 11, 42, 34, 49]
hits = 0

for number in bets:
    if number in drawn:
        hits += 1

print(hits)
```

Home work 3.4 Unique

in
not in

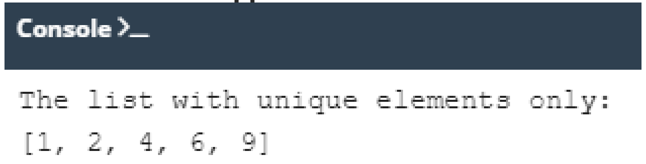
Imagine a list - not very long, not very complicated, just a simple list containing some integer numbers. Some of these numbers may be repeated, and this is the clue. We don't want any repetitions. We want them to be removed.

Your task is to write a program which removes all the number repetitions from the list. The goal is to have a list in which all the numbers appear not more than once.

Note: assume that the source list is hard-coded inside the code - you don't have to enter it from the keyboard. Of course, you can improve the code and add a part that can carry out a conversation with the user and obtain all the data from her/him.

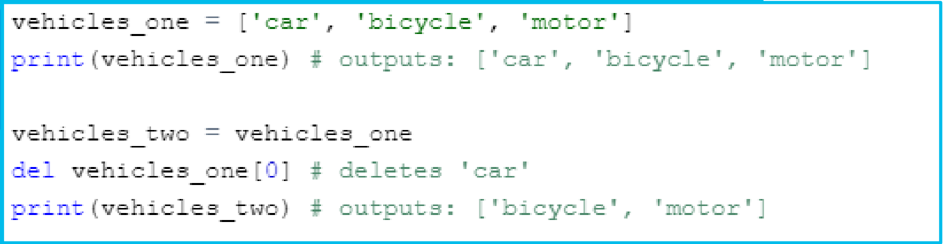
Hint: we encourage you to create a new list as a temporary work area - you don't need to update the list in situ.

```
my_list = [1, 2, 4, 4, 1, 4, 2, 6, 2, 9]
res_list = []
```

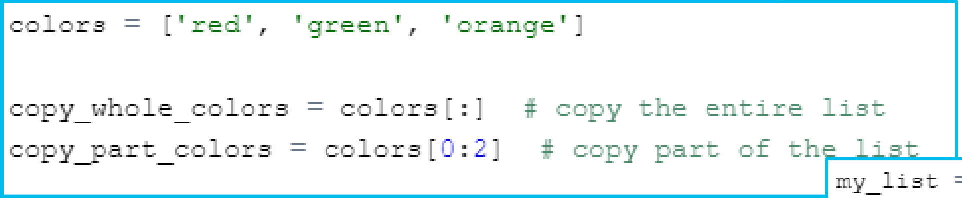


```
The list with unique elements only:
[1, 2, 4, 6, 9]
```

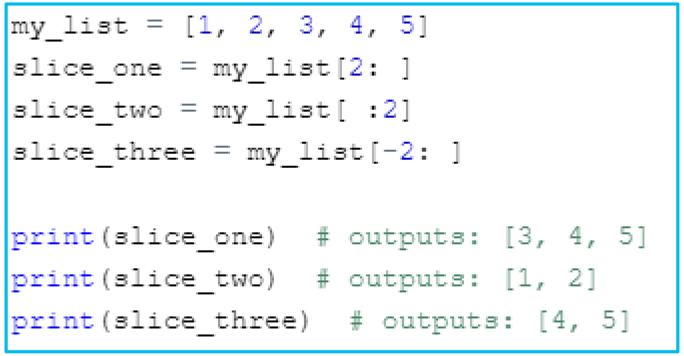
Examples



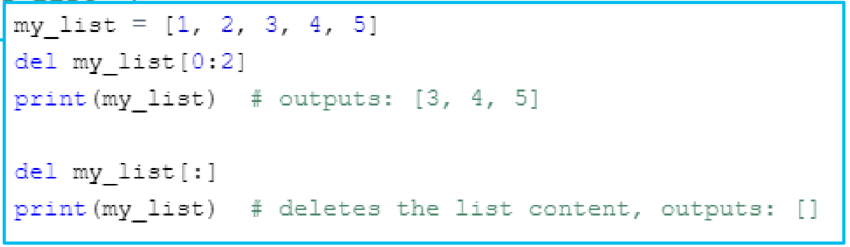
```python
vehicles_one = ['car', 'bicycle', 'motor']
print(vehicles_one) # outputs: ['car', 'bicycle', 'motor']

vehicles_two = vehicles_one
del vehicles_one[0] # deletes 'car'
print(vehicles_two) # outputs: ['bicycle', 'motor']
```



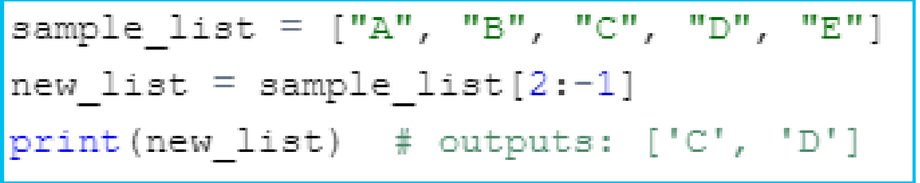
```python
colors = ['red', 'green', 'orange']

copy_whole_colors = colors[:]  # copy the entire list
copy_part_colors = colors[0:2]  # copy part of the list
```



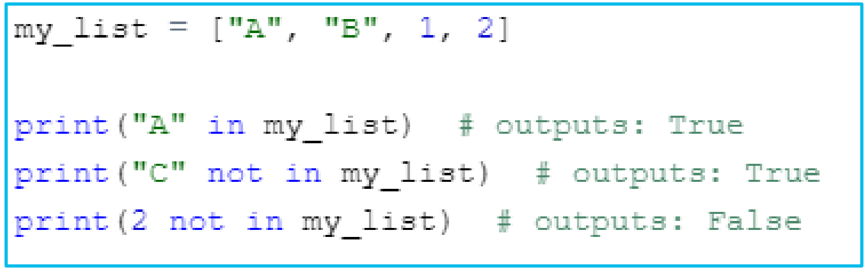
```python
my_list = [1, 2, 3, 4, 5]
slice_one = my_list[2: ]
slice_two = my_list[ :2]
slice_three = my_list[-2: ]

print(slice_one)  # outputs: [3, 4, 5]
print(slice_two)  # outputs: [1, 2]
print(slice_three)  # outputs: [4, 5]
```



```python
my_list = [1, 2, 3, 4, 5]
del my_list[0:2]
print(my_list)  # outputs: [3, 4, 5]

del my_list[:]
print(my_list)  # deletes the list content, outputs: []
```

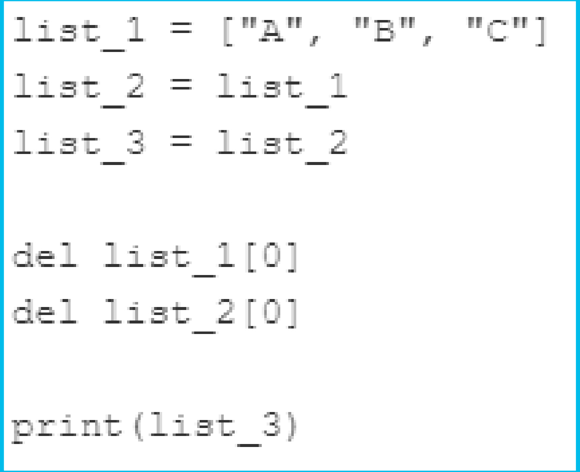


```python
sample_list = ["A", "B", "C", "D", "E"]
new_list = sample_list[2:-1]
print(new_list)  # outputs: ['C', 'D']
```

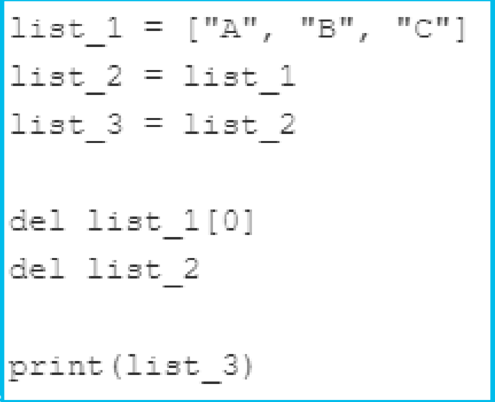


```python
my_list = ["A", "B", 1, 2]

print("A" in my_list)  # outputs: True
print("C" not in my_list)  # outputs: True
print(2 not in my_list)  # outputs: False
```

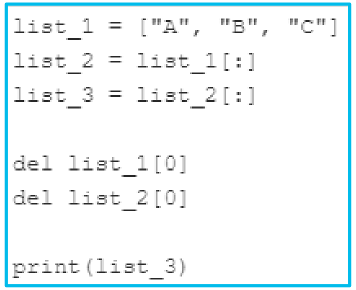
Examples



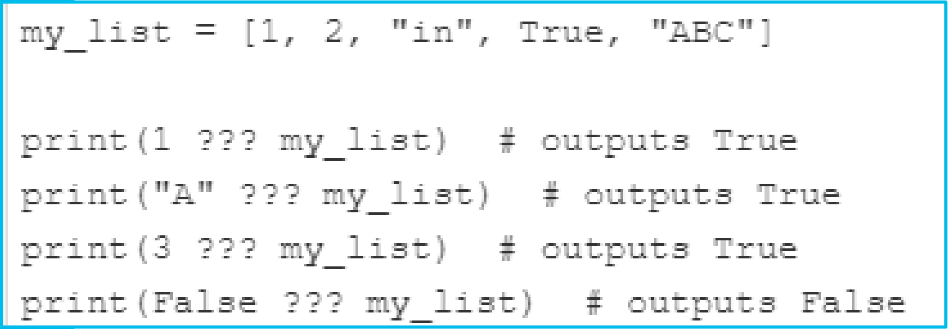
```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2

del list_1[0]
del list_2[0]

print(list_3)
```



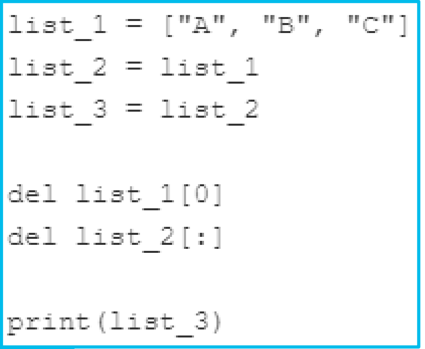
```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2

del list_1[0]
del list_2

print(list_3)
```



```
list_1 = ["A", "B", "C"]
list_2 = list_1[:]
list_3 = list_2[:]

del list_1[0]
del list_2[0]

print(list_3)
```



```
my_list = [1, 2, "in", True, "ABC"]

print(1 ??? my_list)  # outputs True
print("A" ??? my_list)  # outputs True
print(3 ??? my_list)  # outputs True
print(False ??? my_list)  # outputs False
```



```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2

del list_1[0]
del list_2[:]

print(list_3)
```

ЗАДАНИЯ

1) Прорешать всю классную работу
2) Выполнить все домашние задания

Почитать:

1) Byte of Python –
Прочитать страницы -
стр. 47-54
стр. 55-63   стр. 85-87

Крайний срок сдачи 30/09 в 21:00 (можно раньше, но не позже)

ЗАДАНИЯ

Название файлов, которые вы отправляете мне в telegram:

Vasia_Pupkin_class_work_L3_1.py

+все задания ОДНИМ ФАЙЛОМ - Vasia_Pupkin_L3_1.py

Формат сообщения которое вы присылаете мне

(после полного выполнения домашнего задания, только один раз) в Telegram:

Добрый день/вечер. Я Вася Пупкин, и это мои домашние задания к лекции 3 часть 1.

И отправляете файлы

Крайний срок сдачи 30/09 в 21:00 (можно раньше, но не позже)

https://docs.github.com/articles/using-pull-requests

Q&A

Create your possibilities. Bye bye.