Which of the following is false with respect to the following python code?

```
class Student:
    def __init__(random, id, age):
        random.id = id
        random.age = age

s1 = Student(1, 5000)
```

88 users have participated

A	`s1` is an instance of `Student`	13%
B	`id` and `age` are parameters	10%
C	`random` is a reference to `s1`	31%
D	It is mandatory to define an `__init__` dunder	47%

```
print(id(a1))
print(id(a2))
```

```
False
140392013646864
140392013645568
```

Quiz - 3

In [ ]:

```
class Random:
    def __init__(self, a = "Is it possible?"):
        self.a = a

    def display(self):
        print(self.a)

r = Random()
r.display()
```

In [ ]:

In [ ]:

In [ ]:

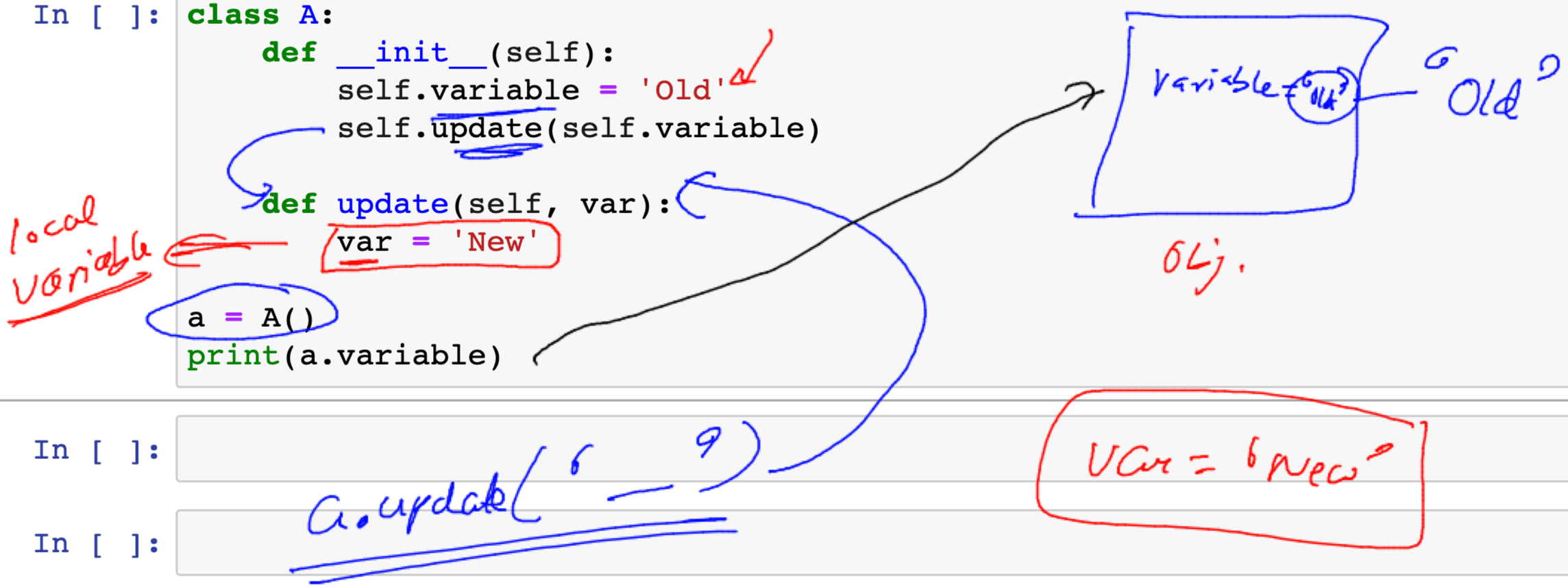
Is it possible?

In [11]:
```
r2 = Random("Hello world")
r2.display()
```

```
Hello world
```

Quiz - 4

In [ ]:
```
class A:
    def __init__(self):
        self.variable = 'Old'
        self.update(self.variable)

    def update(self, var):
        var = 'New'

a = A()
print(a.variable)
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

Python 3.6
(known limitations)

```
1  class A:
2      def __init__(self):
3          self.variable = 'Old'
4          self.update(self.variable)
5
6      def update(self, var):
7          var = 'New'
8
a = A()
print(a.variable)
```

Edit this code

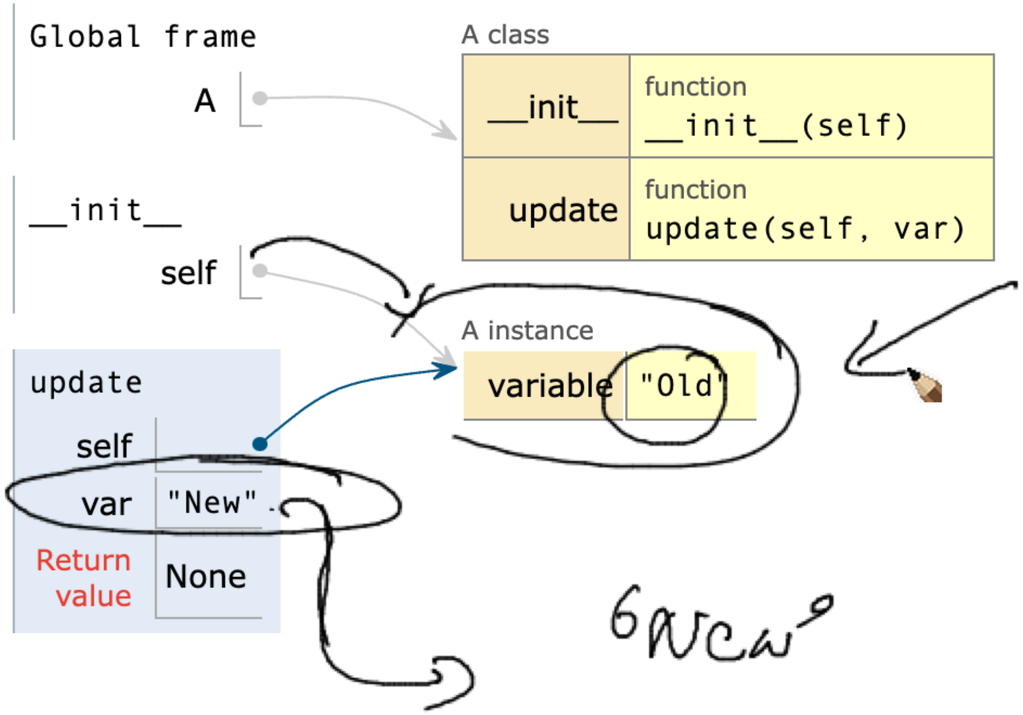
<< First | < Prev | Next > | Last >>

Step 8 of 10

Print output (drag lower right corner to resize)

Strings? immutable.

Frames Objects



Python 3.6
(known limitations)

```
1  class A:
2      def __init__(self):
3          self.variable = 'Old'
4          self.update(self.variable)
5
6      def update(self, var):
7          var = 'New'
8
   a = A()
   print(a.variable)
```

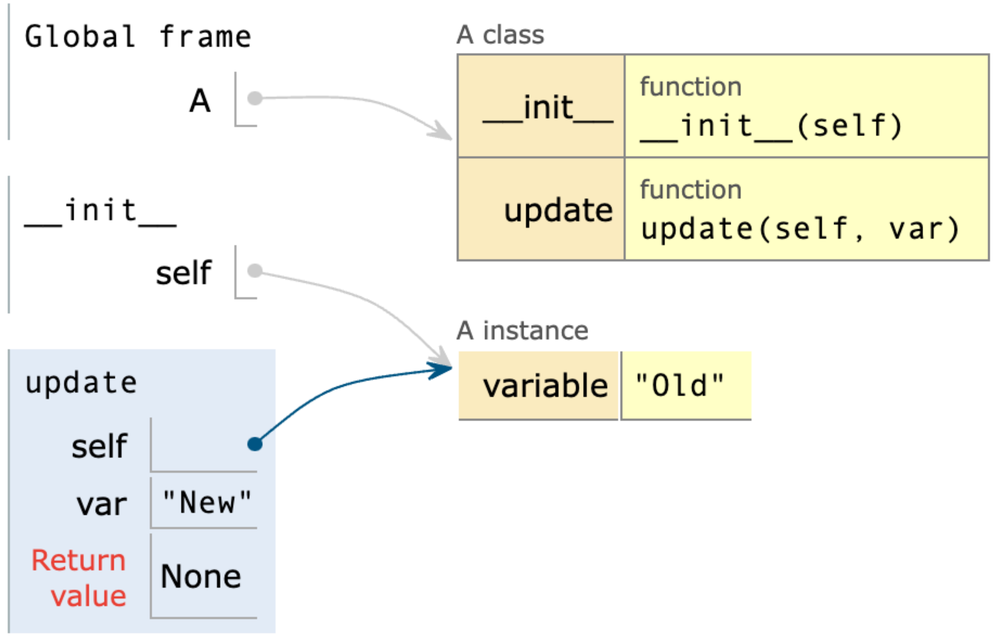
Edit this code

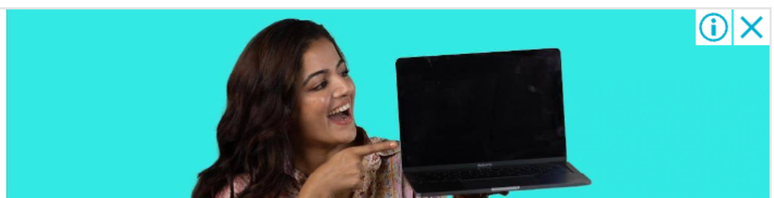
<< First | < Prev | Next > | Last >>

Step 8 of 10

Print output (drag lower right corner to resize)

Frames Objects





In [11]:
```
r2 = Random("Hello world")
r2.display()
```

```
Hello world
```

Quiz - 4

In [12]:
```
class A:
    def __init__(self):
        self.variable = 'Old'
        self.update(self.variable)

    def update(self, var):
        var = 'New'

a = A()
print(a.variable)
```

same ids

=> new obj

```
Old
```

In [13]:
```
print(var)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [13], in <cell line: 1>()
----> 1 print(var)
```

Quiz - 4

In [12]:
```
class A:
    def __init__(self):
        self.variable = 'Old'
        self.update(self.variable)

    def update(self, var):
        var = 'New'

a = A()
print(a.variable)
```

```
Old
```

In [15]:
```
a.update('New')
print(a.variable)
```

```
Old
```

In [13]:
```
print(var)
```

```
---------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Input In [13], in <cell line: 1>()
----> 1 print(var)

NameError: name 'var' is not defined
```

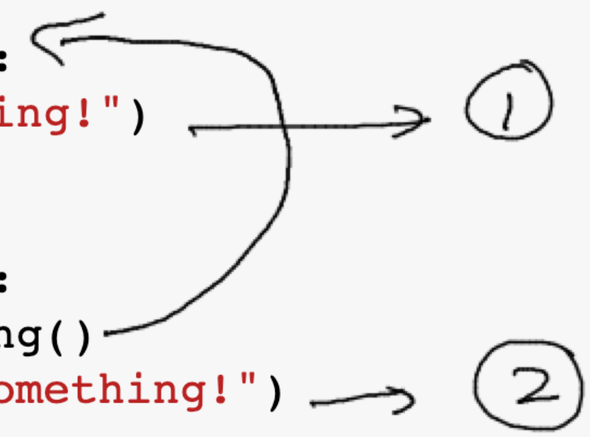
```
a = A()
print(a.variable)
```

New

Quiz - 5

In [ ]:

```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    def do_something(self):
        super().do_something()
        print("I also do something!")

b = B()
b.do_something()
```



In [ ]:

In [ ]:

```
a = A()
print(a.variable)
```

```
New
```

Quiz - 5

In [20]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    def do_something(self):
        super().do_something()
        print("I also do something!")

b = B()
b.do_something()
```

```
I do something!
I also do something!
```

In [ ]:

In [ ]:

```
a = A()
print(a.variable)
```

```
New
```

Quiz - 5

Polymorphism

In [21]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    pass

b = B()
b.do_something()
```

```
I do something!
```

In [ ]:

In [ ]:

Quiz - 5

In [21]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    pass

b = B()
b.do_something()
```

```
I do something!
```

In [22]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    def do_something(self):
        print("I do something else!")

b = B()
b.do_something()
```

method overriding

(Polymorphism)

```
I do something else!
```

In [ ]:

Quiz - 5

In [21]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    pass

b = B()
b.do_something()
```

```
I do something!
```

In [22]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    def do_something(self):
        print("I do something else!")

b = B()
b.do_something()
```

```
I do something else!
```

In [ ]:

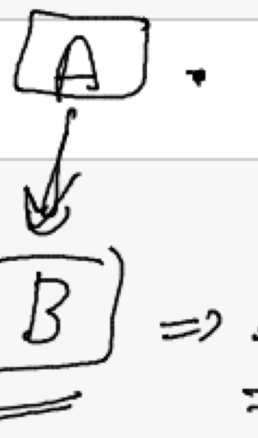
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [21]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    pass

b = B()
b.do_something()
```

Good to know

How to decide which method to call?

```
I do something!
```



Order to resolve methods

In [25]:
```
class A:
    def do_something(self):
        print("I do something!")

class B(A):
    def do_something(self):
        print("I do something else!")

b = B()
b.do_something()
```

Method Resolution Order (MRO)

```
I do something else!
```

In [26]:
```
B.__mro__
```

Out[26]: `(__main__.B, __main__.A, object)`

13 / 13

```
I do something!
```

In [27]:
```
class A:
    def do_something(self):
        print("I do something!")

    def do_nothing(self):
        print("I do nothing")

class B(A):
    def do_something(self):
        print("I do something else!")

b = B()
b.do_something()
```

```
I do something else!
```

In [26]:
```
B.__mro__
```

Out[26]:
```
(__main__.B, __main__.A, object)
```

In [28]:
```
b.do_nothing()
```

```
I do nothing
```

In [ ]:

```
    def do_something(self):
        print("I do something!")

    def do_nothing(self):
        print("I do nothing")

class B(A):
    def do_something(self):
        print("I do something else!")

b = B()
b.do_something()
```

```
I do something else!
```

In [26]:
```
B.__mro__
```

Out[26]:
```
(__main__.B, __main__.A, object)
```

In [28]:
```
b.do_nothing()
```

```
I do nothing
```

In [29]:
```
b.__str__
```

Out[29]:
```
<method-wrapper '__str__' of B object at 0x7faf90201ac0>
```

In [ ]:

In [ ]:

args (special argument)

In [35]:
```
# we want to pass variable number of arguments
# 0, 1, 2, 3, 4, etc. etc as many as you want

def sum_number(x, y, *args):
    result = x + y
    return result
```

variable, 0, 1, 2, 3 — — —

In [36]:
```
sum_number(3, 4)
```

Out[36]: 7

fixed

In [38]:
```
sum_number(1)
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [38], in <cell line: 1>()
----> 1 sum_number(1)

TypeError: sum_number() missing 1 required positional argument: 'y'
```

working now

In [51]:
```
# we want to pass variable number of arguments
# 0, 1, 2, 3, 4, etc. etc as many as you want

def sum_number(x, y, *args):
    print(args) # a tuple
    result = x + y
    result += sum(args)
    return result
```

In [52]:
```
sum_number(1, 2, 3, 4, 5)
# but not adding
```

```
(3, 4, 5)
```

Out[52]: 15

In [ ]:

In [ ]:

In [ ]:

Kwargs

In [82]:
```
def create_person(name, age, gender):
    person = {
        'name': name,
        'age': age,
        'gender': gender
    }
    return person
```

In [83]:
```
p1 = create_person("Anwesh", 25, 'M')
```

In [84]:
```
print(p1)
```

```
{'name': 'Anwesh', 'age': 25, 'gender': 'M'}
```

In [86]:
```
# keyword arguments
p1 = create_person(name = "Anwesh", age = 25, gender = 'M')
```

In [87]:
```
print(p1)
```

```
{'name': 'Anwesh', 'age': 25, 'gender': 'M'}
```

In [ ]:

In [ ]:

In [ ]:

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [ ]:

In [92]:
```
def create_person(name, age, gender, *args):
    person = {
        'name': name,
        'age': age,
        'gender': gender
    }
    return person
```

1) positional

2) keyword

**kwargs

In [93]:
```
p2 = create_person(name = "Anwesh", gender = 'M', age = 35, color = 'red',
                   hobby = "chess")
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [93], in <cell line: 1>()
----> 1 p2 = create_person(name = "Anwesh", gender = 'M', age = 35, color
= 'red',
      2                            hobby = "chess")

TypeError: create_person() got an unexpected keyword argument 'color'
```

19 / 19

In [ ]:

In [ ]:

```
Input In [93], in <cell line: 1>()
----> 1 p2 = create_person(name = "Anwesh", gender = 'M', age = 35, color
= 'red',
      2                            hobby = "chess")

TypeError: create_person() got an unexpected keyword argument 'color'
```

In [94]:
```
p2 = create_person(name = "Anwesh", gender = 'M', age = 35, "red", 95, 2.4
```

```
  Input In [94]
    p2 = create_person(name = "Anwesh", gender = 'M', age = 35, "red", 9
5, 2.44, 1+3j)

                                                  ^
SyntaxError: positional argument follows keyword argument
```

In [96]:
```
p3 = create_person("Anwesh", 25, 'M', 'red', 95, 3.45, 2+3j)
```

In [97]:
```
print(p3)
```

```
{'name': 'Anwesh', 'age': 25, 'gender': 'M'}
```

In [ ]:

In [97]:
```
print(p3)
```

```
{'name': 'Anwesh', 'age': 25, 'gender': 'M'}
```

**kwargs for variable keyworded arguments

In [98]:
```
def create_person(name, age, gender, **extra_info):
    print(extra_info)
    person = {
        'name': name,
        'age': age,
        'gender': gender
    }
    return person
```

In [100]:
```
p2 = create_person(name = "Anwesh", gender = 'M', age = 35,
                   color = 'red', hobby = "chess")
```

```
{'color': 'red', 'hobby': 'chess'}
```

In [ ]:

In [102]:
```
p2 = create_person(name = "Anwesh", gender = 'M', age = 35,
                   color = 'red', hobby = "chess")
```

```
{'color': 'red', 'hobby': 'chess'}
```

In [103]:
```
print(p2)
```

```
{'name': 'Anwesh', 'age': 35, 'gender': 'M', 'color': 'red', 'hobby': 'ch
ess'}
```

Quiz

In [105]:
```
def print_stats(name, **info):
    print(name, 'is:')
    for key, value in info.items():
        print(f'{key}: {value}')
```

In [106]:
```
print_stats(age=10, gender=m)
```

```
---------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Input In [106], in <cell line: 1>()
----> 1 print_stats(age=10, gender=m)

NameError: name 'm' is not defined
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Problems on FP with Quizzes

In [117]:
```
my_list = ['banana', 'mango', 'guava']
# sorted_list = ['banana','guava','mango']

def sort_strings(str_list):
#     sorted_list = ___A___

    return sorted_list
```

In [118]:
```
# sorted(str_list, key=lambda x: x.count('a'), reverse=True)
```

In [119]:
```
# sorted(str_list, key=x.count('a'), reverse=True)
```

In [120]:
```
# sorted(str_list, key=lambda x: for 'a' in x, reverse=True)
```

In [121]:
```
# None of the Above
```

In [ ]:
```

```

23 / 23

Problems on FP with Quizzes

In [117]:
```
my_list = ['banana', 'mango', 'guava']
# sorted_list = ['banana','guava','mango']

def sort_strings(str_list):
#     sorted_list = ___A___

    return sorted_list
```

In [118]:
```
# sorted(str_list, key=lambda x: x.count('a'), reverse=True)
```

In [119]:
```
# sorted(str_list, key=x.count('a'), reverse=True)
```

In [120]:
```
# sorted(str_list, key=lambda x: for 'a' in x, reverse=True)
```

In [121]:
```
# None of the Above
```

In [122]:
```
'banana'.count('a')
```

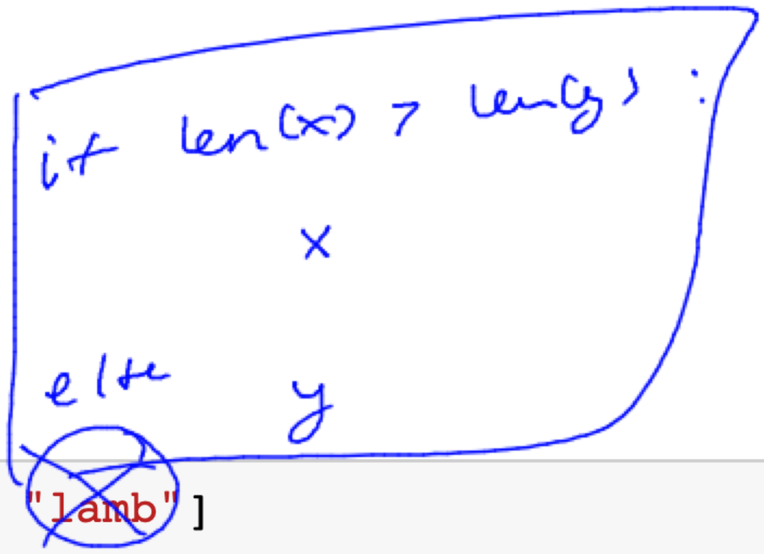
Out[122]: 3

In [ ]:

In [122]: banana.count('a')

Out[122]: 3

Type *Markdown* and LaTeX: α^2



Quiz - 2

In [135]:
```
words = ["mary", "had", "a", "little", "lamb"]
# biggestWord = _____A_____
# print(biggestWord)
```

In [ ]:
```
reduce(lambda x, y: x if len(x) > len(y) else y, words)
```

In [ ]:
```
reduce(lambda x, y: x if len(x) > len(y), words)
```

In [133]:
```
# map(lambda x, y: x if len(x) > len(y) else y, words)
```

In [ ]:

Out[122]: 3

Type *Markdown* and LaTeX: α^2

Quiz - 2

In [135]:
```
words = ["mary", "had", "a", "little", "lamb"]
# biggestWord = _____A__________
# print(biggestWord
```

In [138]:
```
from functools import reduce
reduce(lambda x, y: x if len(x) > len(y) else y, words)
```

Out[138]: 'little'

In [139]:
```
reduce(lambda x, y: x if len(x) > len(y), words)
```

```
  Input In [139]
    reduce(lambda x, y: x if len(x) > len(y), words)
                                              ^

SyntaxError: invalid syntax
```


In [133]:
```
# map(lambda x, y: x if len(x) > len(y) else y, words)
```

In [138]:
```
from functools import reduce
reduce(lambda x, y: x if len(x) > len(y) else y, words)
```

Out[138]: 'little'

In [139]:
```
reduce(lambda x, y: x if len(x) > len(y), words)
```

```
  Input In [139]
    reduce(lambda x, y: x if len(x) > len(y), words)
                                                    ^

SyntaxError: invalid syntax
```

In [133]:
```
# map(lambda x, y: x if len(x) > len(y) else y, words)
```

In [141]:
```
list(map(lambda x, y: x if len(x) > len(y) else y, words))
```

```
---------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [141], in <cell line: 1>()
----> 1 list(map(lambda x, y: x if len(x) > len(y) else y, words))

TypeError: <lambda>() missing 1 required positional argument: 'y'
```

In [ ]:

SyntaxError: invalid syntax

In [133]:
```
# map(lambda x, y: x if len(x) > len(y) else y, words)
```

In [141]:
```
list(map(lambda x, y: x if len(x) > len(y) else y, words))
```

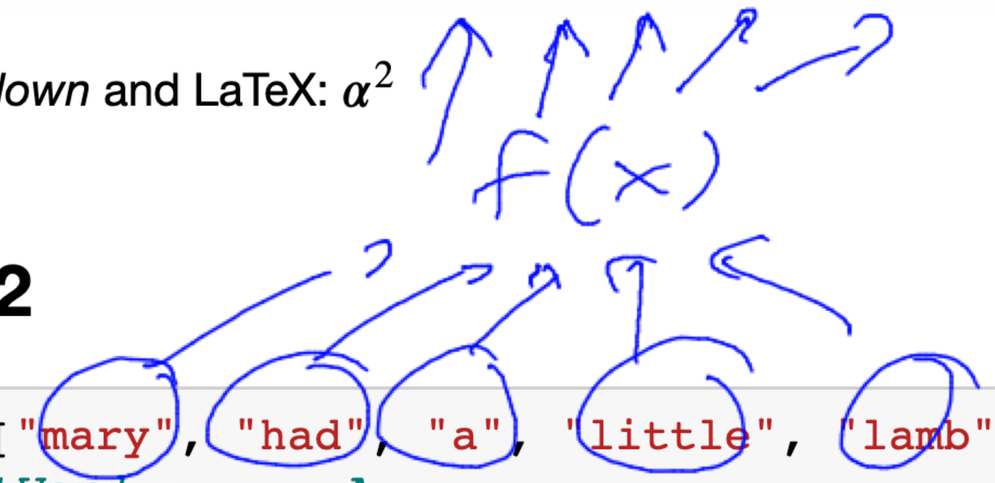
```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [141], in <cell line: 1>()
----> 1 list(map(lambda x, y: x if len(x) > len(y) else y, words))

TypeError: <lambda>() missing 1 required positional argument: 'y'
```

In [142]:
```
list(map(lambda x: len(x), words))
```

Out[142]:
```
[4, 3, 1, 6, 4]
```

In [ ]:

Type *Markdown* and LaTeX: α^2



Quiz - 2

In [135]:
```
words = ["mary", "had", "a", "little", 'lamb"]
# biggestWord = _____A__________
# print(biggestWord
```

In [138]:
```
from functools import reduce
reduce(lambda x, y: x if len(x) > len(y) else y, words)
```

Out[138]: 'little'

In [139]:
```
reduce(lambda x, y: x if len(x) > len(y), words)
```

```
  Input In [139]
    reduce(lambda x, y: x if len(x) > len(y), words)
                                              ^

SyntaxError: invalid syntax
```

In [133]:
```
# map(lambda x, y: x if len(x) > len(y) else y, words)
```

In [141]:
```
list(map(lambda x, y: x if len(x) > len(y) else y, words))
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [142]:
```
list(map(lambda x: len(x), words))
```

Out[142]: [4, 3, 1, 6, 4]

Additional Problems on FP

Q1.

In [146]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

# find the maximum number in this nested list
# in all the list, 1st element is a character/string, 2nd element is int

max(a)
```

Out[146]: ['e', 5]

In [145]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]

# find the maximum number in this nested list
```

In [ ]:

Out[142]:
```
[4, 3, 1, 6, 4]
```

Additional Problems on FP

Q1.

In [146]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

# find the maximum number in this nested list
# in all the list, 1st element is a character/string, 2nd element is int

max(a)
```

Out[146]:
```
['e', 5]
```



```
Docstring:
max(iterable, *[, default=obj, key=func]) -> value
max(arg1, arg2, *args, *[, key=func]) -> value

With a single iterable argument, return its biggest item. The
default keyword-only argument specifies an object to return if
the provided iterable is empty.
With two or more arguments, return the largest argument.
Type:      builtin_function_or_method
```

Out[142]: [4, 3, 1, 6, 4]

Additional Problems on FP

Q1.

In [146]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

# find the maximum number in this nested list
# in all the list, 1st element is a character/string, 2nd element is int

max(a)
```

Out[146]: ['e', 5]

```
string:
(iterable, *[, default=obj, key=func]) -> value
(arg1, arg2, *args, *[, key=func]) -> value

With a single iterable argument, return its biggest item. The
default keyword-only argument specifies an object to return if
the provided iterable is empty.
With two or more arguments, return the largest argument.
Type:      builtin_function_or_method
```

Q1.

In [146]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

# find the maximum number in this nested list
# in all the list, 1st element is a character/string, 2nd element is int

max(a)
```

Out[146]:
```
['e', 5]
```

In [147]:
```
max?
```

In [148]:
```
max(a, key = lambda x : x[0])
```

Out[148]:
```
['e', 5]
```

In [150]:
```
# max 2nd element in the nested lists
max(a, key = lambda x : x[1])
```

Out[150]:
```
['d', 9]
```

In [ ]:

In [ ]:

In [ ]:

Q1.

In [146]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

# find the maximum number in this nested list
# in all the list, 1st element is a character/string, 2nd element is int

max(a)
```

Out[146]:
```
['e', 5]
```

In [147]:
```
max?
```

In [148]:
```
max(a, key = lambda x : x[0])
```

Out[148]:
```
['e', 5]
```

In [150]:
```
# max 2nd element in the nested lists
max(a, key = lambda x : x[1])
```

Out[150]:
```
['d', 9]
```

In [153]:
```
res = max(a, key = lambda x : x[1])[1]
print(res)
```

```
9
```

Q1.

In [146]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

# find the maximum number in this nested list
# in all the list, 1st element is a character/string, 2nd element is int

max(a)
```

Out[146]:
```
['e', 5]
```

In [147]:
```
max?
```

In [148]:
```
max(a, key = lambda x : x[0])
```

Out[148]:
```
['e', 5]
```

In [150]:
```
# max 2nd element in the nested lists
max(a, key = lambda x : x[1])
```

Out[150]:
```
['d', 9]
```

In [153]:
```
res = max(a, key = lambda x : x[1])[1]
print(res)
```

```
9
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [147]:
```
max?
```

In [148]:
```
max(a, key = lambda x : x[0])
```

Out[148]:
```
['e', 5]
```

In [150]:
```
# max 2nd element in the nested lists
max(a, key = lambda x : x[1])
```

Out[150]:
```
['d', 9]
```

In [153]:
```
res = max(a, key = lambda x : x[1])[1]
print(res)
```

```
9
```

x y x[1] y[1]?

In [ ]:
```
from functools import reduce

a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

res = reduce(lambda x, y: )
```

In [ ]:

In [ ]:

In [ ]:

36 / 36

```
max(a, key = lambda x : x[1])
```

Out[150]:
```
['d', 9]
```

In [153]:
```
res = max(a, key = lambda x : x[1])[1]
print(res)
```

```
9
```

In [154]:
```
from functools import reduce

a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

res = reduce(lambda x, y: x if x[1] > y[1] else y, a)
```

In [155]:
```
print(res)
```

```
['d', 9]
```

In [156]:
```
res = res[-1]
print(res)
```

```
9
```

In [ ]:

In [145]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]
```

```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], ['e', 5]]

res = reduce(lambda x, y: x if x[1] > y[1] else y, a)
```

In [155]:
```
print(res)
```

```
['d', 9]
```

In [156]:
```
res = res[-1]
print(res)
```

```
9
```

In [ ]:

In [145]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]

# find the maximum number in this nested list
```

In [ ]:

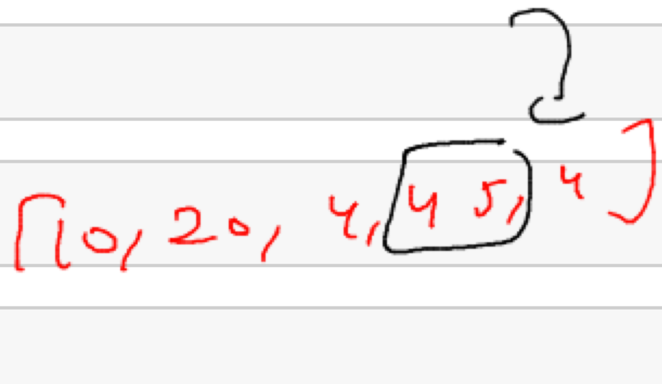
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [ ]:

In [145]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]

# find the maximum number in this nested list
```

In [ ]:

Q10



In [157]:
```
lst = [10, 20, 4, 45, 99]
# find the 2nd maximum element using functional programming
```

In [158]:
```
max1 = max(lst)
print(max1)
```

```
99
```

max
min

If we replace 1st max with min of the arr.

=> max of new arr => 2nd max.

In [ ]:

In [ ]:

In [ ]:

39 / 39

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [145]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]

# find the maximum number in this nested list
```

In [ ]:
```

```

Q10

In [157]:
```
lst = [10, 20, 4, 45, 99]
# find the 2nd maximum element using functional programming
```

In [158]:
```
max1 = max(lst)
print(max1)
```

```
99
```

In [159]:
```
mini = min(lst)
print(mini)
```

```
4
```

In [ ]:
```
max2 = max(lst, key = lambda x: mini if (x == max1) else x)
```

In [ ]:
```

```

40 / 40

Q10

In [157]:
```
lst = [10, 20, 4, 45, 99]
# find the 2nd maximum element using functional programming
```

In [158]:
```
max1 = max(lst)
print(max1)
```

```
99
```

In [159]:
```
mini = min(lst)
print(mini)
```

```
4
```

In [160]:
```
max2 = max(lst, key = lambda x: mini if (x == max1) else x)

print(max2)
```

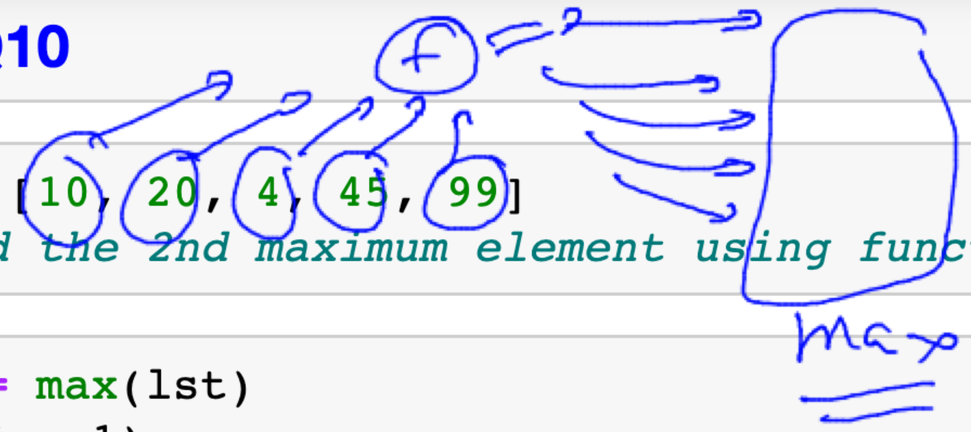
```
45
```

In [162]:
```
print(lst) # not at all changed
```

```
[10, 20, 4, 45, 99]
```

In [ ]:

Q10



In [157]:
```
lst = [10, 20, 4, 45, 99]
# find the 2nd maximum element using functional programming
```

In [158]:
```
max1 = max(lst)
print(max1)
```

```
99
```

In [159]:
```
mini = min(lst)
print(mini)
```

```
4
```

In [160]:
```
max2 = max(lst, key = lambda x: mini if (x == max1) else x)

print(max2)
```

```
45
```

In [162]:
```
print(lst) # not at all changed
```

```
[10, 20, 4, 45, 99]
```

In [ ]:

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

Python 3.6
(known limitations)

```
t = [10, 20, 4, 45, 99]
find the 2nd maximum element using functional programmin

x1 = max(lst)
int(max1)

ni = min(lst)
int(mini)

x2 = max(lst, key = lambda x: mini if (x == max1) else x)

int(max2)
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 21 of 22

Customize visualization

Print output (drag lower right corner to resize)

```
99
4
```

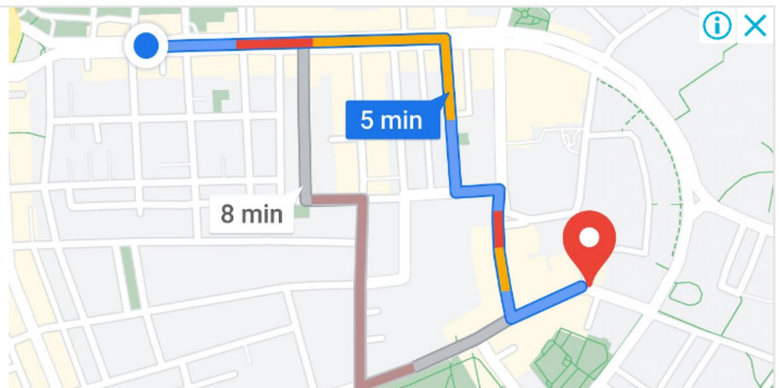
Frames | Objects

Global frame
- lst
- max1 99
- mini 4

λ
- x 99
- Return value 4

list

0	1	2	3	4
10	20	4	45	99



File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [160]:
```
max2 = max(lst, key = lambda x: mini if (x == max1) else x)

print(max2)
```

```
45
```

In [162]:
```
print(lst) # not at all changed
```

```
[10, 20, 4, 45, 99]
```

Doubts / QnA / Assignment HW Problems

In [ ]:
```
a = [["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]
max(a, key = lambda x: max(x[0], x[1]) if isinstance(x[0], int) and
                                          isinstance(x[1], int)
                       else
```

In [164]:
```
max('a', 6)
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [164], in <cell line: 1>()
----> 1 max('a', 6)
```

In [162]:
```
print(lst) # not at all changed
```

```
[10, 20, 4, 45, 99]
```

Doubts / QnA / Assignment HW Problems

In [166]:
```
[["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]


_list = max(a, key = lambda x: max(x[0], x[1]) if isinstance(x[0], int) and
                                                  isinstance(x[1], int)
                               else x[1])
```

In [167]:
```
print(res_list)
print(max(res_list))
```

```
[10, 5]
10
```

In [164]:
```
max('a', 6)
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [164], in <cell line: 1>()
----> 1 max('a', 6)
```

In [162]:
```
print(lst) # not at all changed
```

```
[10, 20, 4, 45, 99]
```

Doubts / QnA / Assignment HW Problems

In [166]:
```
[["a", 6], ["b", 1], ["c", 4], ["d", 9], [10, 5]]


_list = max(a, key = lambda x: max(x[0], x[1]) if isinstance(x[0], int) and
                                                  isinstance(x[1], int)
                          else x[1])
```

In [167]:
```
print(res_list)
print(max(res_list))
```

```
[10, 5]
10
```

In [164]:
```
max('a', 6)
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [164], in <cell line: 1>()
----> 1 max('a', 6)
```

```python
class Matrix:
    # Define properties here


    # Define constructor here
    def __init__(self, r, c):
        self.rows = r
        self.columns = c

    def input(self):
        # Complete the function
        self.data = []
        for _ in range(self.rows):
            cur_list = list(map(int, input().split()))
            self.data.append(cur_list)
        # [[1, 2, 3], [4, 5, 6]]

    def add(self, x):
        # Complete the function
        res = Matrix(self.rows, self.columns)
        res.data = []
        for i in range(self.rows):
            cur_list = []
            for j in range(self.columns):
                cur_list.append(x[i][j] + self.data[i][j])

    def subtract(self, x):
        # Complete the function


    def transpose(self):
        # Complete the function


    def print(self):
        # Complete the function
```

rows = 2

cols = 3

data =

[[1,2,3],

[4,5,6]]

x = [[0,0,0],

[1,1,1]]

```python
    # Define constructor here
    def __init__(self, r, c):
        self.rows = r
        self.columns = c

    def input(self):
        # Complete the function
        self.data = []
        for _ in range(self.rows):
            cur_list = list(map(int, input().split()))
            self.data.append(cur_list)
        # [[1, 2, 3], [4, 5, 6]]

    def add(self, x):
        # Complete the function
        res = Matrix(self.rows, self.columns)
        res.data = []
        for i in range(self.rows):
            cur_list = []
            for j in range(self.columns):
                cur_list.append(x.data[i][j] + self.data[i][j])
            res.data.append(cur_list)
        return res

    def subtract(self, x):
        # Complete the function
        pass


    def transpose(self):
        # Complete the function
        pass


    def print(self):
        # Complete the function
        for i in range(self.rows):
            for j in range(self.columns):
```

rows

Columns

data = [[--]]

x

x.data[i][j]

48 / 48

```
            pass


    def print(self):
        # Complete the function
        for i in range(self.rows):
            for j in range(self.columns):
                print(self.data[i][j])
            print()

a = Matrix(2, 3)
a.input()
b = Matrix(2, 3)
b.input()
c1 = a.add(b)
c1.print()
# Matrix c2 = a.subtract(b)
# Matrix c3 = a.transpose()
# a.print()
```

Edit this code

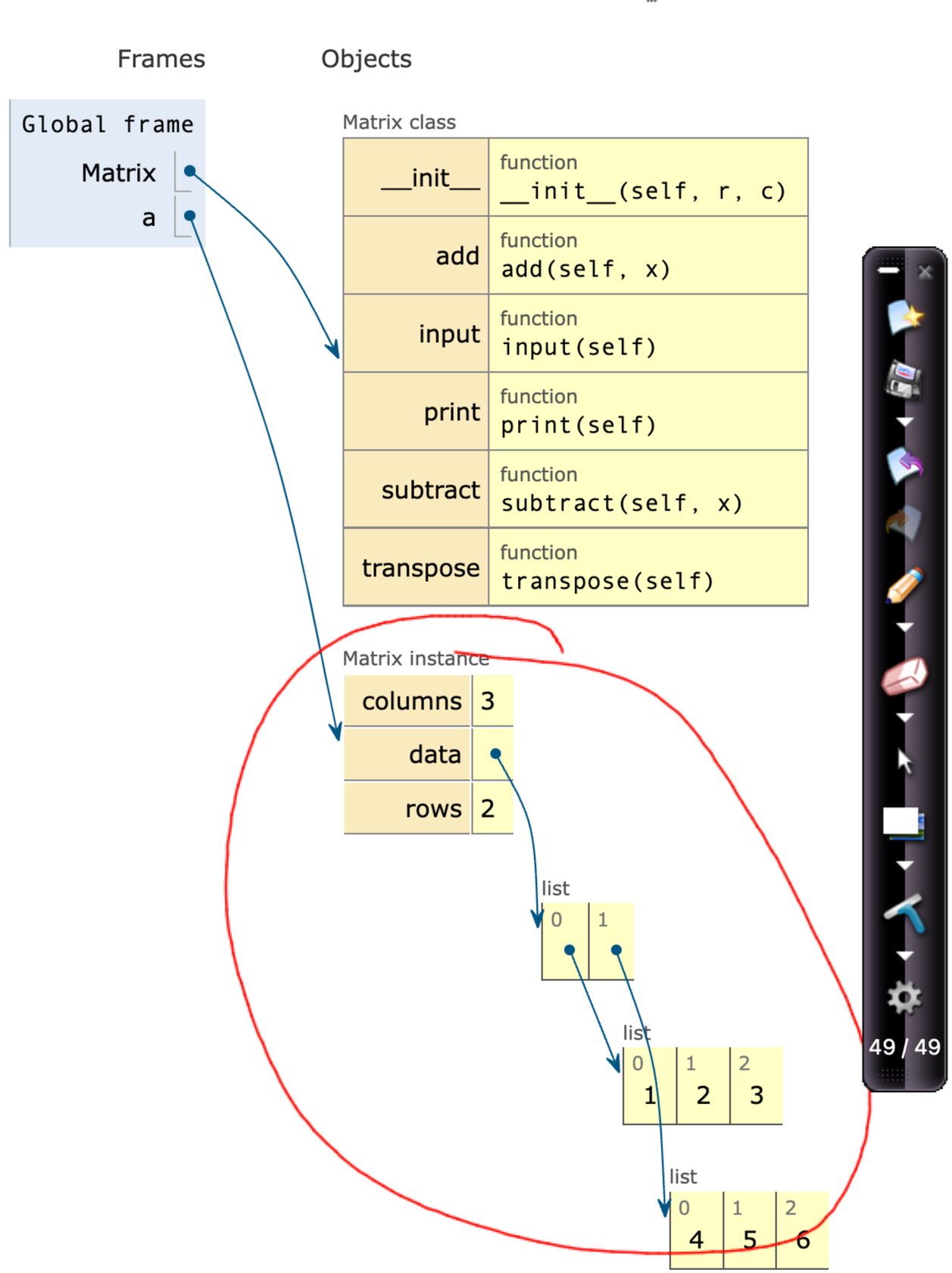
line that just executed

next line to execute

<< First | < Prev | Next > | Last >>

Step 18 of 26

Customize visualization



Edit this code

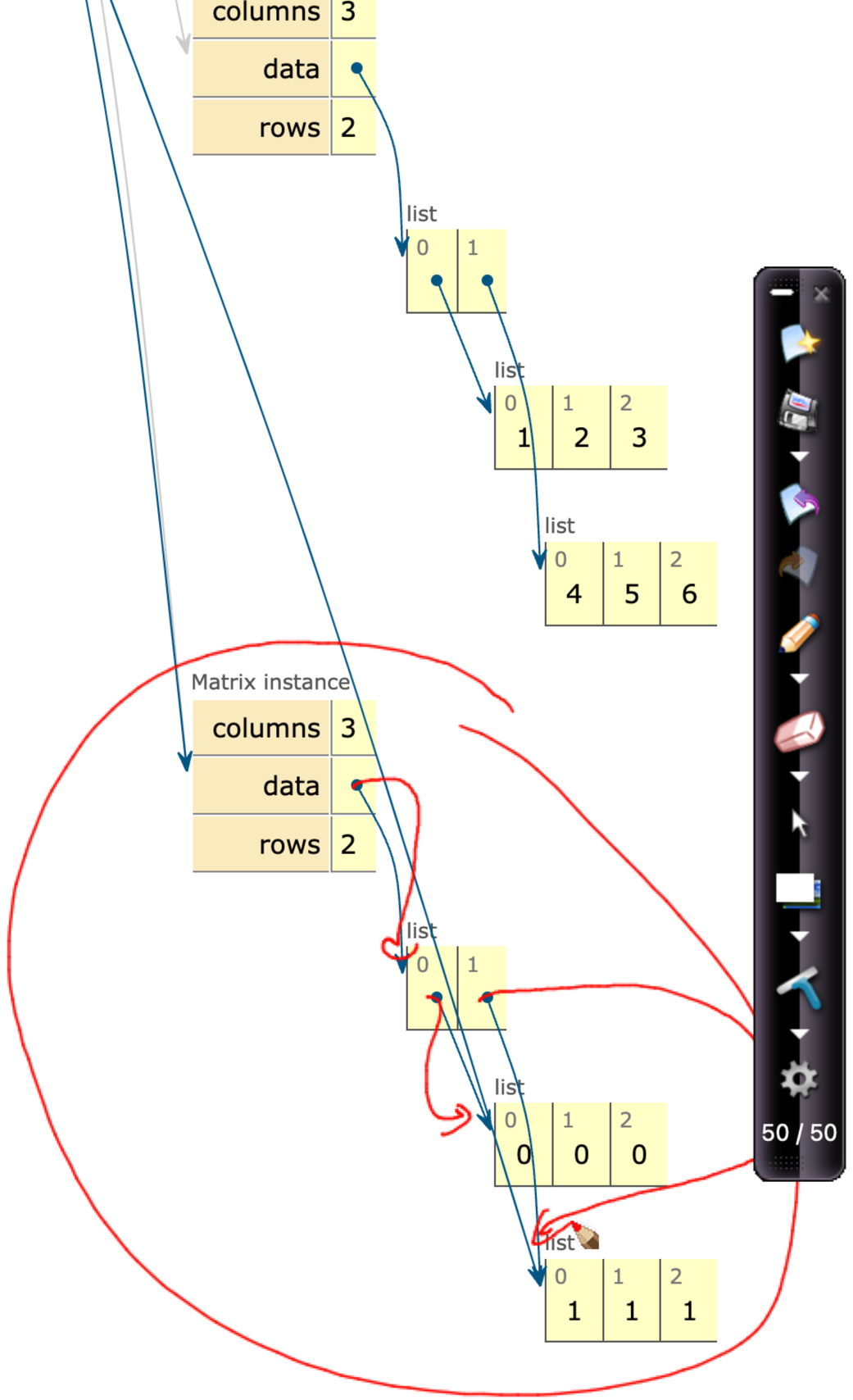
line that just executed

next line to execute

<< First | < Prev | Next > | Last >>

Step 32 of 86

Customize visualization



Edit this code

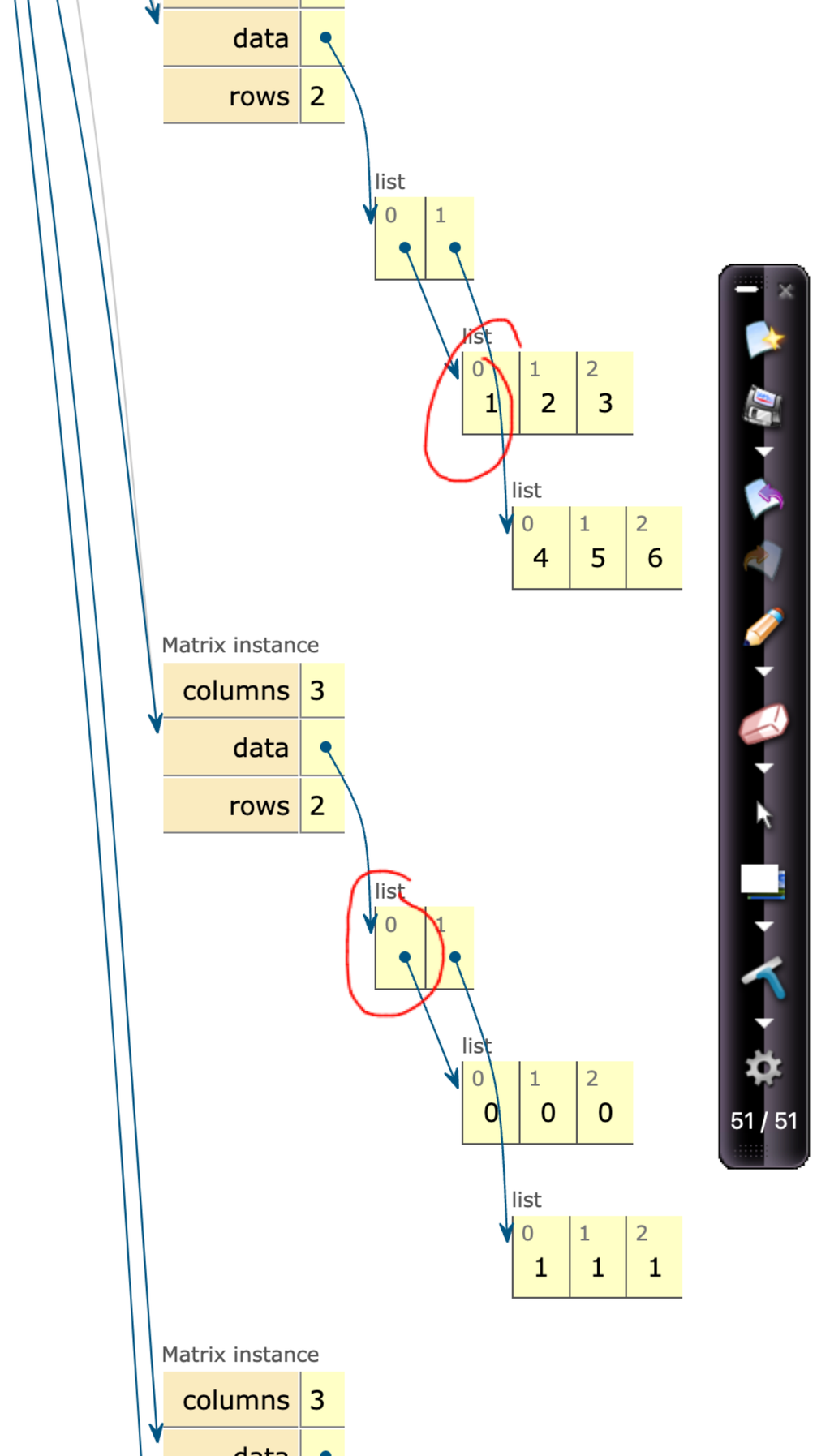
line that just executed

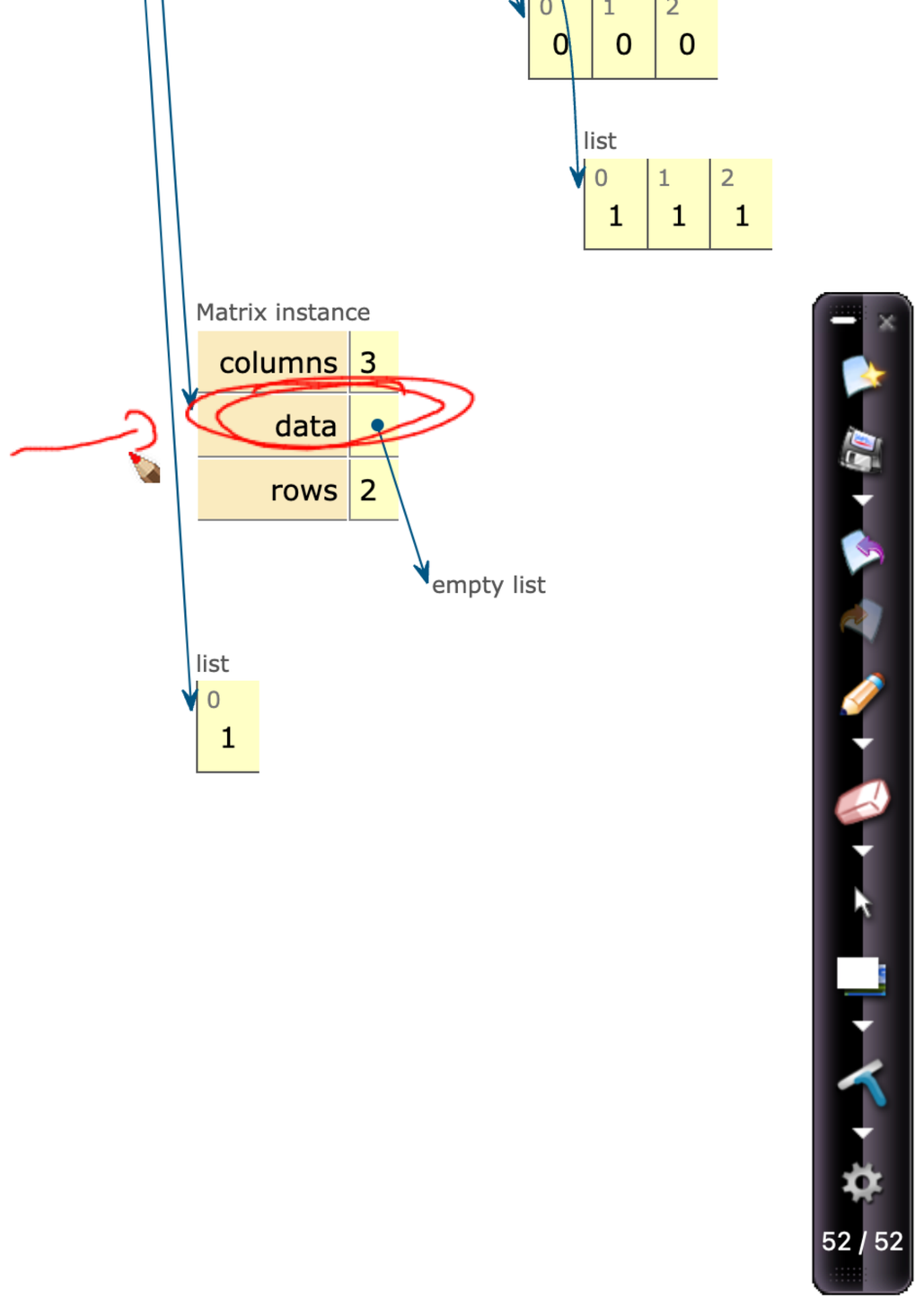
next line to execute

<< First | < Prev | Next > | Last >>

Step 47 of 86

Customize visualization





Python 3.6
(known limitations)

```
11          # Complete the function
12          self.data = []
13          for _ in range(self.rows):
14              cur_list = list(map(int, input().split
15              self.data.append(cur_list)
16          # [[1, 2, 3], [4, 5, 6]]
17
18      def add(self, x):
19          # Complete the function
20          res = Matrix(self.rows, self.columns)
21          res.data = []
22          for i in range(self.rows):
23              cur_list = []
24              for j in range(self.columns):
25                  cur_list.append(x.data[i][j] + sel
26              res.data.append(cur_list)
27          return res
28
29      def subtract(self, x):
30          # Complete the function
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 47 of 86

Customize visualization

Print output (drag lower right corner to resize)

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

Frames Objects

Global frame
- Matrix
- a
- b

add
- self
- x
- res
- i 0
- cur_list
- j 1

Matrix class

__init__	function `__init__(self, r, c)`
add	function `add(self, x)`
input	function `input(self)`
print	function `print(self)`
subtract	function `subtract(self, x)`
transpose	function `transpose(self)`

Matrix instance
- columns 3
- data
- rows 2

list
- 0
- 1

list
- 0 1
- 1 2
- 2 3

X = Matrix(->
x[i][j]

53 / 53

1692.7 ms ×12

Input Format

List of Strings

Output Format

Sorted list of strings based on length

I/P: ccc bbb ddd
3 3 3
3

Example Input

cccc b dd aaa
4 1 2 3

O/P: ['bbb', 'ccc', 'ddd']

Example Output

['b', 'dd', 'aaa', 'cccc']

HINTS | Hint 1 | Complete Solution

Please wait, fetching code...

Editor Mode - Normal | Python 3 (python-3.5)

```
1 def sort_strings(str_list):
2     #str_list-> list of strings
```

54 / 54

- Hide Solution Suffix

```
str_list = input().split()
print(sort_strings(str_list))
```

Save | Reset

Submit | Test | Test with Custom Inpu

Compilation

Your input:

```
bbb ccc dddd aaa
```

sorted (—— ?

Your function returned the following output:

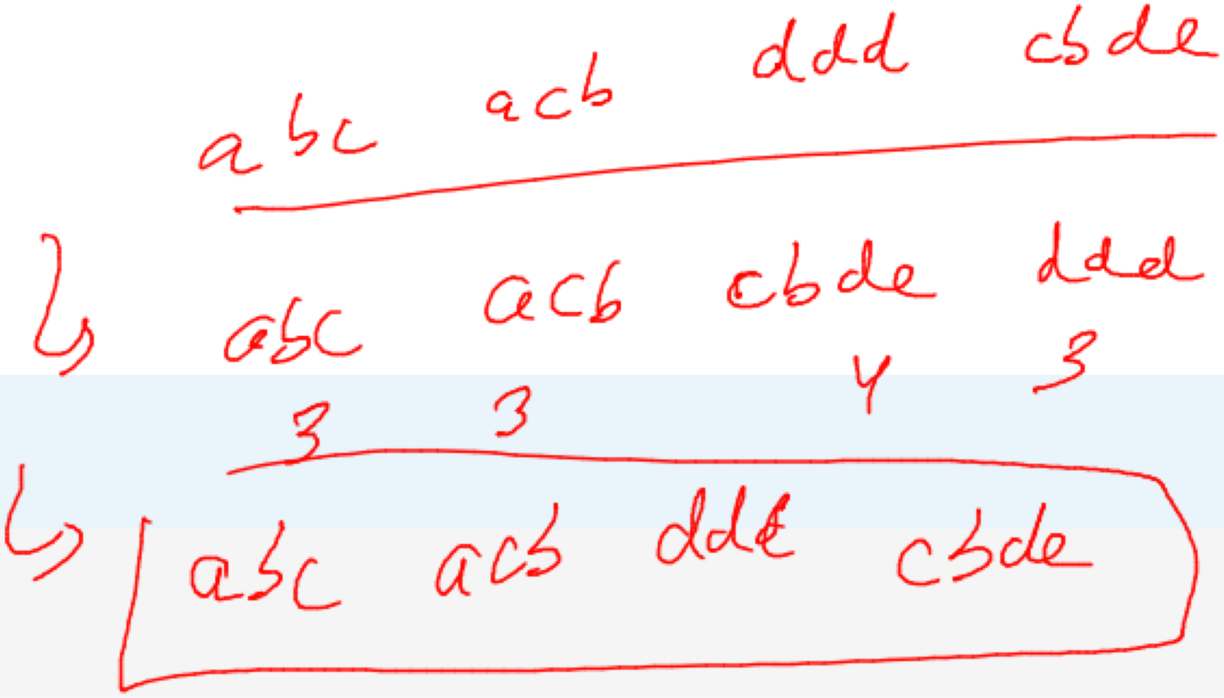
```
['bbb', 'ccc', 'aaa', 'dddd']
```

55 / 55

Tip 1: Unexpected result ? Put debug statements to see what's going wrong.

Tip 2: Different result than the Test/Submit? Check for invalid memory access or uninitialized global variables.

```python
def sort_strings(str_list):
    #str_list-> list of strings

    # return the sorted list of strings on the basis of criteria mentioned
    res = sorted(str_list)
    res = sorted(res, key = lambda x: len(x))

    return res


# return the sorted list of strings on the basis of criteria mentioned # code starts here n = le
```



- Hide Solution Suffix

```
str_list = input().split()
print(sort_strings(str_list))
```

Save Reset

Submit Test Test with Custom Input

56 / 56

Compilation

Your input:

Finder File Edit View Go Window Help

Sat 8 Oct 0:13

57 / 57