In [11]:
```
type(y)
```

Out[11]: function

In [12]:
```
(lambda x: x**3)(2)
```

Out[12]: 8

In [13]:
```
(lambda x: x**3)(5)
```

Out[13]: 125

In [14]:
```
lambda lst: lst[0]
```

Out[14]: <function __main__.<lambda>(lst)>

In [15]:
```
(lambda lst: lst[0])([9, 5, 6])
```

f(x)

Out[15]: <function __main__.<lambda>(lst)>

In [ ]:

Use case

In [20]:
```
coordinates = [(1, -1), (-1, 2), (3, 5), (5, 3), (7, 7), (2, 1)]
```

In [21]:
```
# we want to sort them on the basis of x coordinate
```

In [22]:
```
res = sorted(coordinates)
```

In [23]:
```
print(res)
```

```
[(-1, 2), (1, -1), (2, 1), (3, 5), (5, 3), (7, 7)]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [21]:
```
# we want to sort them on the basis of x coordinate
```

In [22]:
```
res = sorted(coordinates)
```

In [23]:
```
print(res)
```

```
[(-1, 2), (1, -1), (2, 1), (3, 5), (5, 3), (7, 7)]
```

In [24]:
```
# we want to sort them on the basis of y coordinate
```

In [25]:
```
l = [1, 4, 3, 2, 5]
```

In [27]:
```
x = sorted(l) # creates a copy of the list which is sorted
print(x)
print(l)
```

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

In [28]:
```
l.sort()
```

In [29]:
```
print(l)
```

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

In [ ]:
```

```

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

Solution

In [31]:
```
sorted?
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

pass fn as key

```
nature: sorted(iterable, /, *, key=None, reverse=False)
string:
Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the
reverse flag can be set to request the result in descending order.
Type:      builtin_function_or_method
```

[1, 2, 3, 4, 5]

Solution

Selection Sort

Bubble Sort

a > b => swap.

In [31]:
```
sorted?
```

In [32]:
```
coordinates = [(1, -1), (-1, 2), (3, 5), (5, 3), (7, 7), (2, 1)]
```

In [ ]:
```
def
```

In [ ]:

$l_1 = [1, -1]$ $l_2 = [-1, 2]$

$l_1[1]$? $l_2[1]$

In [ ]:

In [ ]:

```
nature: sorted(iterable, /, *, key=None, reverse=False)
string:
Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the
reverse flag can be set to request the result in descending order.
Type:      builtin_function_or_method
```

[1, 2, 3, 4, 5]

Solution

In [31]:
```
sorted?
```

In [32]:
```
coordinates = [(1, -1), (-1, 2), (3, 5), (5, 3), (7, 7), (2, 1)]
```

In [33]:
```
def get_y(lst):
    return lst[1]
```

In [34]:
```
sorted(coordinates, key=get_y)
```

Out[34]:
```
[(1, -1), (2, 1), (-1, 2), (5, 3), (3, 5), (7, 7)]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Out[38]: [(1, -1), (2, 1), (-1, 2), (5, 3), (3, 5), (7, 7)]

In [43]:
```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

In [44]:
```
l = [5, 1, 2, 3, 4]
bubble_sort(l)
```

Out[44]: [1, 2, 3, 4, 5]

In [45]:
```
(1, -1) > (-1, 2)
```

Out[45]: True

In [ ]:

In [ ]:

Out[38]:
```
[(1, -1), (2, 1), (-1, 2), (5, 3), (3, 5), (7, 7)]
```

In [43]:
```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

In [44]:
```
l = [5, 1, 2, 3, 4]
bubble_sort(l)
```

Out[44]:
```
[1, 2, 3, 4, 5]
```

In [45]:
```
(1, -1) > (-1, 2)
```

Out[45]:
```
True
```

In [46]:
```
bubble_sort(coordinates)
```

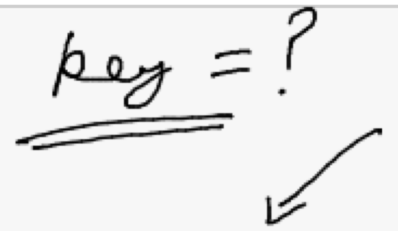
Out[46]:
```
[(-1, 2), (1, -1), (2, 1), (3, 5), (5, 3), (7, 7)]
```

In [ ]:

Modified bubble sort to sort based on y

In [47]:
```
def get_y(lst):
    return lst[1]
```

In [48]:
```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if get_y(arr[j]) > get_y(arr[j + 1]):
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```



In [49]:
```
coordinates = [(1, -1), (-1, 2), (3, 5), (5, 3), (7, 7), (2, 1)]
```

In [50]:
```
bubble_sort(coordinates)
```

Out[50]:
```
[(1, -1), (2, 1), (-1, 2), (5, 3), (3, 5), (7, 7)]
```

In [ ]:

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

Python 3.6
(known limitations)

```
1  def get_y(lst):
2      return lst[1]
3
4  def bubble_sort(arr):
5      n = len(arr)
6      for i in range(n - 1):
7          for j in range(n - i - 1):
8              if get_y(arr[j]) > get_y(arr[j + 1]):
9                  arr[j], arr[j + 1] = arr[j + 1], a
10     return arr
11
12 bubble_sort([(-1, -1), (2, 0)])
```

Edit this code

line that just executed
next line to execute

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

Step 8 of 18

Customize visualization

Frames Objects

Global frame
- get_y → function get_y(lst)
- bubble_sort → function bubble_sort(arr)

bubble_sort
- arr → list [0, 1]
- n: 2
- i: 0
- j: 0

tuple [0: -1, 1: -1]

tuple [0: 2, 1: 0]

10 / 10

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

Python 3.6
(known limitations)

```
1  def get_y(lst):
2      return lst[1]
3
4  def bubble_sort(arr):
5      n = len(arr)
6      for i in range(n - 1):
7          for j in range(n - i - 1):
8              if get_y(arr[j]) > get_y(arr[j + 1]):
9                  arr[j], arr[j + 1] = arr[j + 1], a
10     return arr
11
12 bubble_sort([(2, 0), (-1, -1)])
```

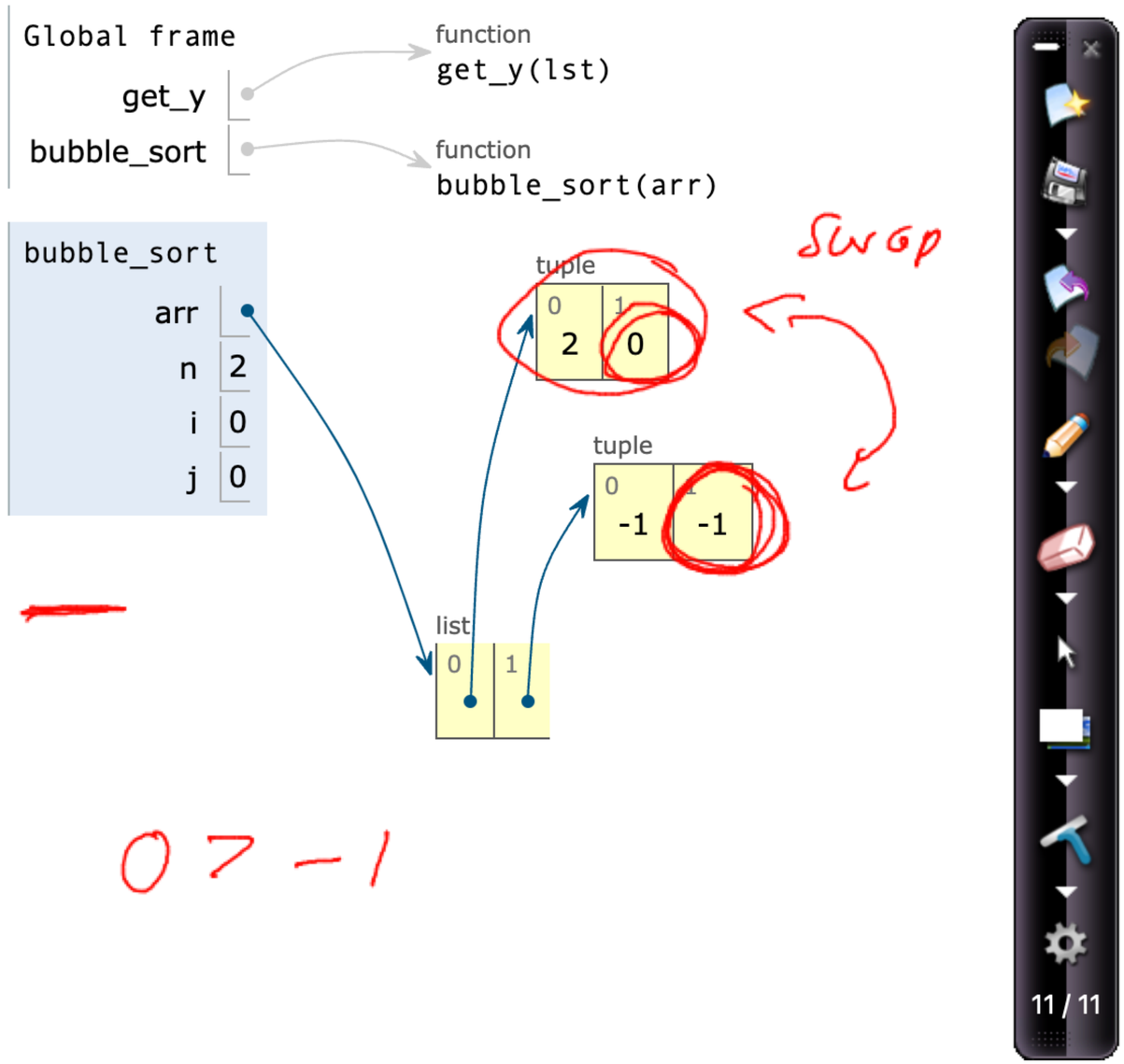
Edit this code

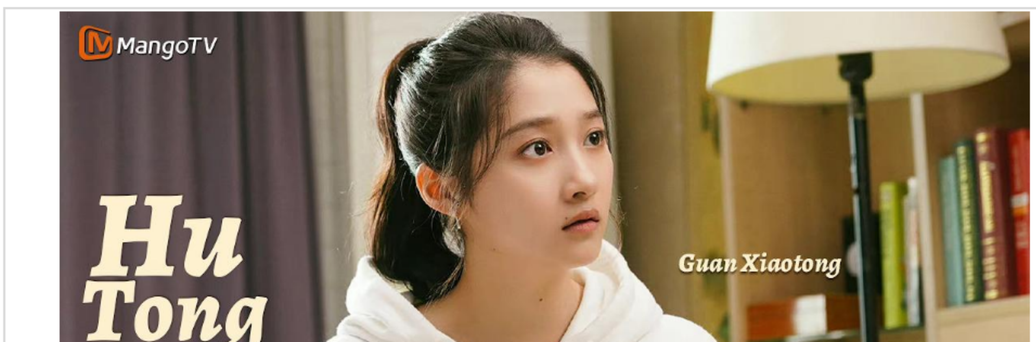
line that just executed
next line to execute

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

Step 15 of 19

Customize visualization





Hu Tong

Walk into the hutong and feel

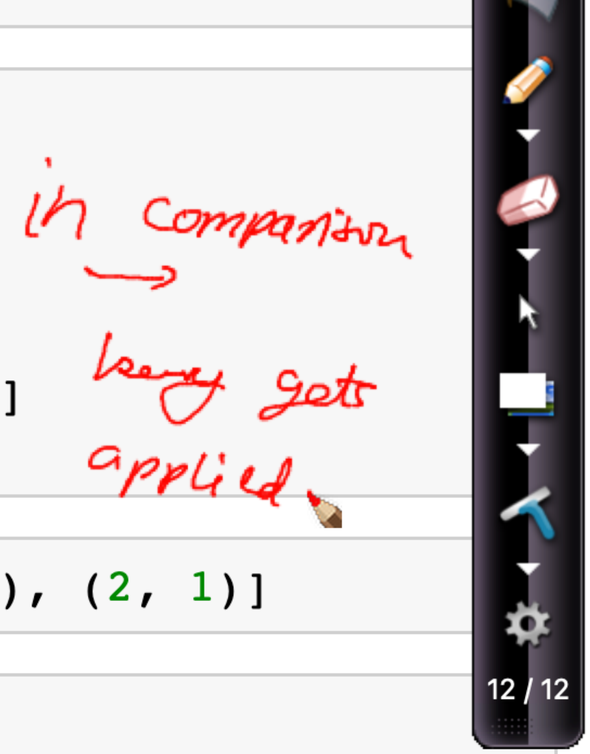
In [46]:
```
bubble_sort(coordinates)
```

Out[46]:
```
[(-1, 2), (1, -1), (2, 1), (3, 5), (5, 3), (7, 7)]
```

Modified bubble sort to sort based on y

In [47]:
```
def get_y(lst):
    return lst[1]
```

In [48]:
```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if get_y(arr[j]) > get_y(arr[j + 1]):
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```



In [49]:
```
coordinates = [(1, -1), (-1, 2), (3, 5), (5, 3), (7, 7), (2, 1)]
```

In [50]:
```
bubble_sort(coordinates)
```

Out[50]:
```
[(1, -1), (2, 1), (-1, 2), (5, 3), (3, 5), (7, 7)]
```

One more example

In [52]:
```
students = [
    {"name": "A", "marks": 50},
    {"name": "B", "marks": 100},
    {"name": "C", "marks": 40},
    {"name": "D", "marks": 70},
    {"name": "E", "marks": 60},
]
```

In [53]:
```
## HW : Sort based on the marks
```

In [54]:
```
sorted(students, key = lambda x: x['marks'])
```

Out[54]:
```
[{'name': 'C', 'marks': 40},
 {'name': 'A', 'marks': 50},
 {'name': 'E', 'marks': 60},
 {'name': 'D', 'marks': 70},
 {'name': 'B', 'marks': 100}]
```

In [ ]:

In [ ]:

Out[56]:
```
[{'name': 'B', 'marks': 100},
 {'name': 'D', 'marks': 70},
 {'name': 'E', 'marks': 60},
 {'name': 'A', 'marks': 50},
 {'name': 'C', 'marks': 40}]
```

In [55]:
```
sorted?
```

Higher Order Functions

In [57]:
```
# a function that returns another functions
```

In [58]:
```
def gen_exp(n):
    def exp(x):
        return x**n

    return exp
```

In [59]:
```
exp_5 = gen_exp(5)
```

In [60]:
```
type(exp_5)
```

Out[60]:
```
function
```

In [ ]:

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

Python 3.6
(known limitations)

```
1  def gen_exp(n):
2      def exp(x):
3          return x**n
4
5      return exp
6
7  exp_5 = gen_exp(5)
8  exp_5(2)
```

Edit this code

t executed
execute

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

Step 5 of 10

zation

Frames Objects

Global frame

gen_exp

function
gen_exp(n)

f1: gen_exp

n 5

exp

function
exp(x) [parent=f1]

exp = lambda x : x**5

15 / 15

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

Python 3.6
(known limitations)

```
1  def gen_exp(n):
2      def exp(x):
3          return x**n
4
5      return exp
6
7  exp_5 = gen_exp(5)
8  exp_5(2)
```

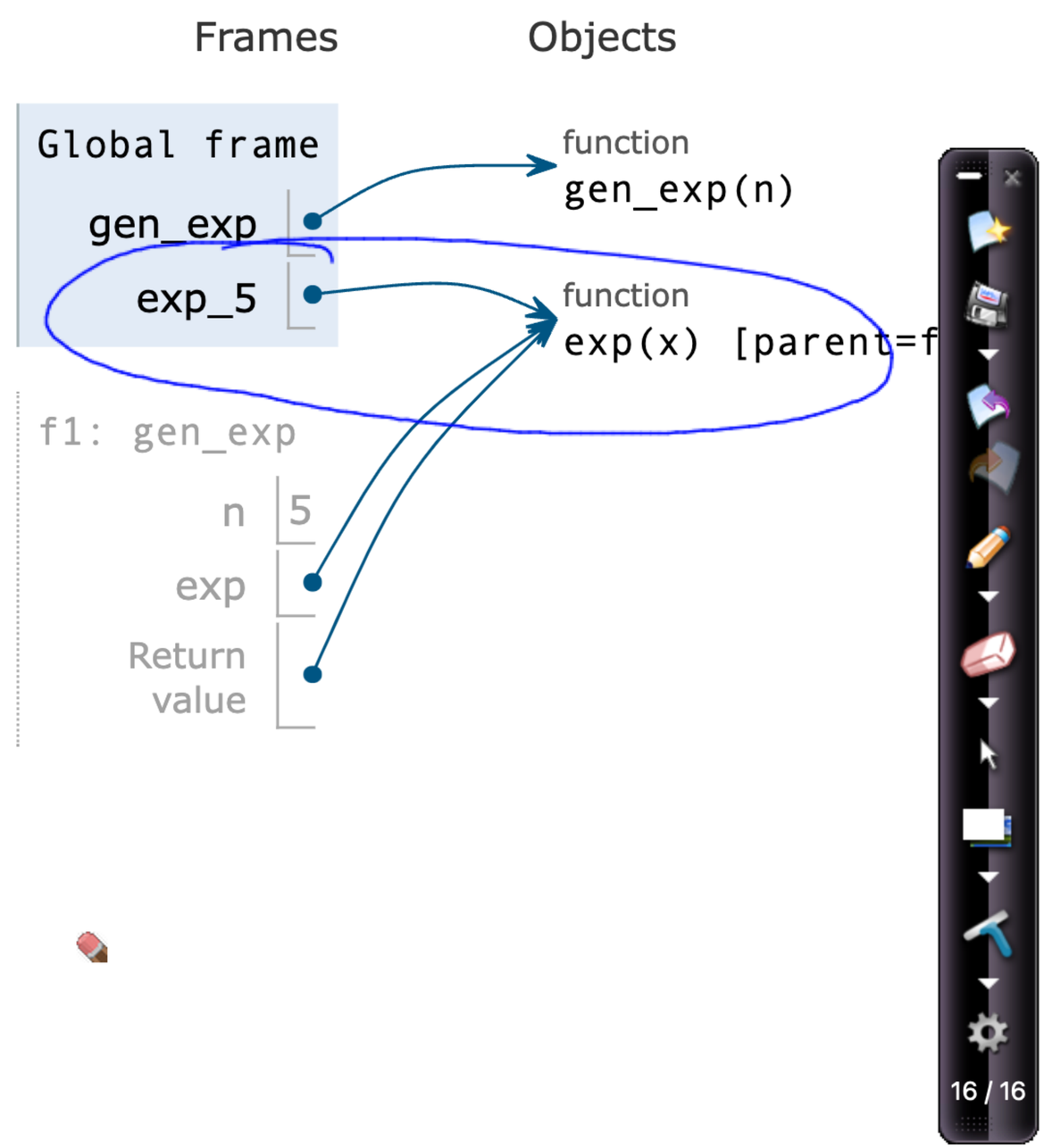
Edit this code

t executed
execute

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

Step 7 of 10

zation



In [61]:
```
exp_5(2)
```

Out[61]: 32

Quiz

lambda: func2 (c,d)

In [ ]:
```
def func2(c, d):
    return c, d

def func1(a, b):
    c = a**1
    d = b**2
    return lambda: func2(c,d)

result = func1(1, 2)

print(result())
```

a=1, b=2
c = a = 1
d = b² = 4
{ c = 1, d = 4

In [ ]:

In [ ]:

In [ ]:

In [ ]:

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

In [61]:
```
exp_5(2)
```

Out[61]: 32

Quiz

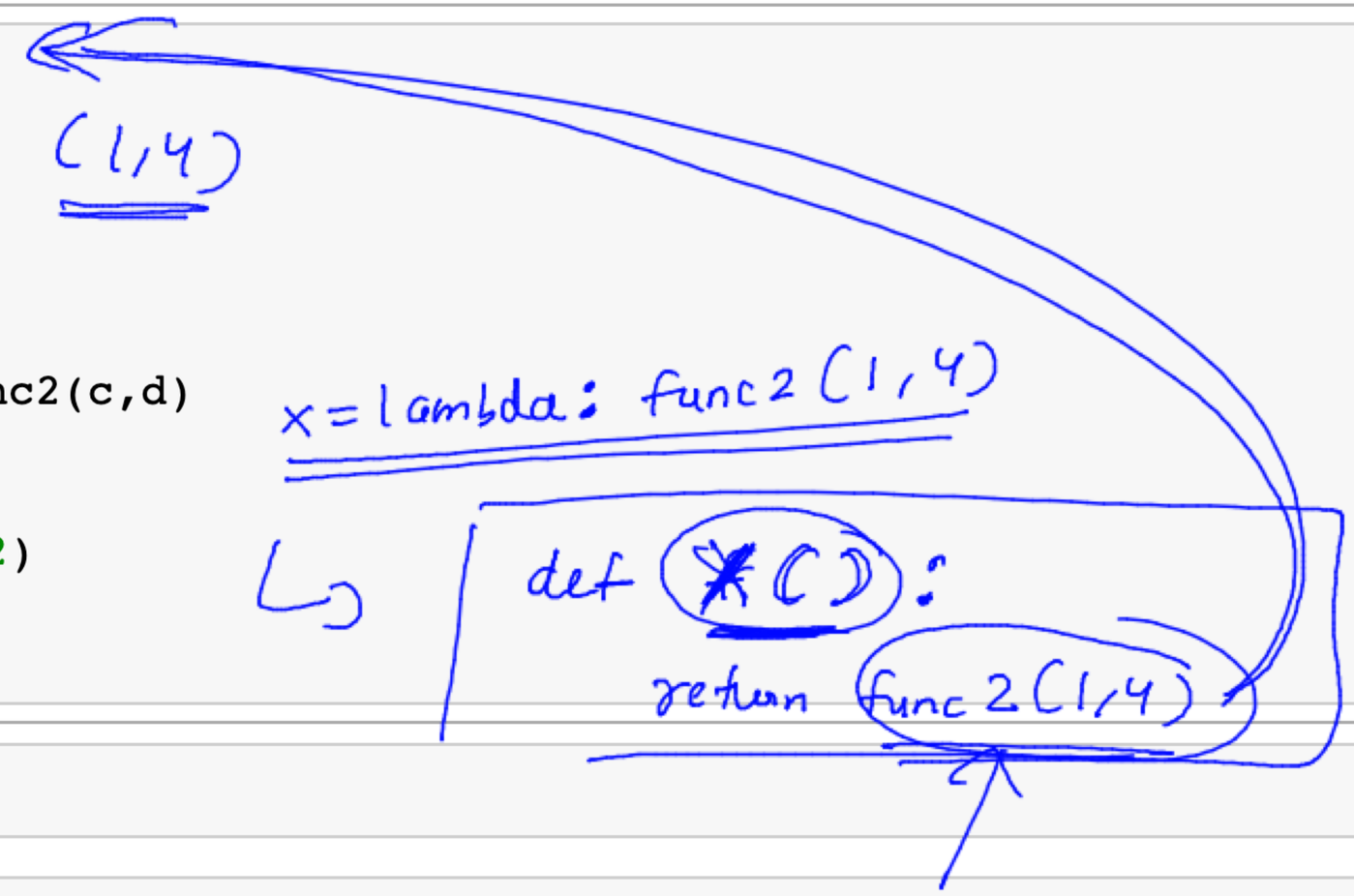
In [ ]:
```
def func2(c, d):
    return c, d

def func1(a, b):
    c = a**1
    d = b**2
    x = lambda: func2(c,d)
    return x

result = func1(1, 2)

print(result())
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:



```
    c = a**1
    d = b**2
    x = lambda: func2(c,d)
    return x

result = func1(1, 2)

print(result())
```

```
(1, 4)
```

In [63]:

```
def func2(c, d):
    return c, d

def func1(a, b):
    c = a**1
    d = b**2
    return func2(c,d)

result = func1(1, 2)

print(result())
```

(1,4)

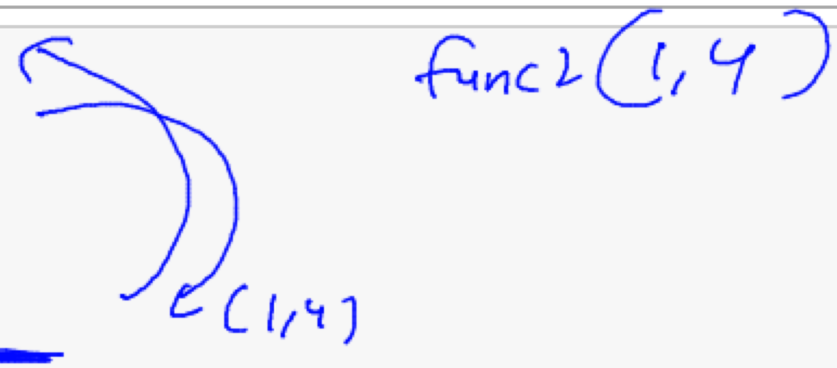
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [63], in <cell line: 11>()
      7     return func2(c,d)
      9 result = func1(1, 2)
---> 11 print(result())

TypeError: func1.<locals>.func2() missing 2 required positional arguments: 'c' and 'd'
```

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

```
(1, 4)
```

In [65]:

```
def func2(c, d):
    return c, d

def func1(a, b):
    c = a**1
    d = b**2
    return func2(c,d)

result = func1(1, 2)

print(result())
```




```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call las
t)
Input In [65], in <cell line: 11>()
      7     return func2(c,d)
      9 result = func1(1, 2)
---> 11 print(result())

TypeError: 'tuple' object is not callable
```

In [ ]:

```
---> 11 print(result())

TypeError: 'tuple' object is not callable
```

Example 2

In [68]:
```
def multiply_n(n):
    def multiply(x):
        return x*n

    return multiply
```

In [69]:
```
mul_5 = multiply_n(5)
```

In [71]:
```
mul_5(10)
```

Out[71]: 50

In [70]:
```
mul_6 = multiply_n(6)
```

In [72]:
```
mul_6(10)
```

Out[72]: 60

In [ ]:

Decorators

DRY

In [73]:
```
def foo():
    print('-'*50)
    print('Hello Everyone! How are you doing?')
    print('-'*50)
```

In [74]:
```
foo()
```

```
--------------------------------------------------
Hello Everyone! How are you doing?
--------------------------------------------------
```

In [75]:
```
def bar():
    print('-'*50)
    print('Random cliche print statement!')
    print('-'*50)
```

In [76]:
```
bar()
```

```
--------------------------------------------------
Random cliche print statement!
--------------------------------------------------
```

In [ ]:

In [ ]:

Decorators

In [73]:
```
def foo():
    print('-'*50)
    print('Hello Everyone! How are you doing?')
    print('-'*50)
```

In [74]:
```
foo()
```

```
--------------------------------------------------
Hello Everyone! How are you doing?
--------------------------------------------------
```

In [75]:
```
def bar():
    print('-'*50)
    print('Random cliche print statement!')
    print('-'*50)
```

In [76]:
```
bar()
```

```
--------------------------------------------------
Random cliche print statement!
--------------------------------------------------
```

In [78]:
```
# can I pass a function as argument also?
```

In [ ]:
```
def pretty(func):
    def inner():
```

```
        return inner
```

In [81]:
```
# is pretty a Higher order function? Yes because it returns another func
```

In [82]:
```
def foo():
    print("Hello Everyon! How are you doing?")
```

In [83]:
```
new_foo = pretty(foo)
```

In [84]:
```
new_foo()
```

```
--------------------------------------------------
Hello Everyon! How are you doing?
--------------------------------------------------
```

In [85]:
```
def bar():
    print("Random Cliche Statement!")
```

In [86]:
```
new_bar = pretty(bar)
```

In [87]:
```
new_bar()
```

```
--------------------------------------------------
Random Cliche Statement!
--------------------------------------------------
```

In [ ]:
```

```

```
--------------------------------------------------
Random cliche print statement!
--------------------------------------------------
```

In [78]:
```
# can I pass a function as argument also?
```

In [79]:
```
def pretty(func):
    def inner():
        print('-'*50)
        func()
        print('-'*50)
    return inner
```

In [81]:
```
# is pretty a Higher order function? Yes because it returns another func
```

In [88]:
```
def foo():
    print("Hello Everyone! How are you doing?")
```

In [83]:
```
new_foo = pretty(foo)
```

A fn taking fn as inp & giving fn as output => Decorator

In [84]:
```
new_foo()
```

```
--------------------------------------------------
Hello Everyon! How are you doing?
--------------------------------------------------
```

In [85]:
```
def bar():
    print("Random Cliche Statement!")
```

```
Random cliche print statement!
--------------------------------------------------
```

In [78]:
```
# can I pass a function as argument also?
```

In [79]:
```
def pretty(func):
    def inner():
        print('-'*50)
        func()
        print('-'*50)
    return inner
```

In [81]:
```
# is pretty a Higher order function? Yes because it returns another func
```

In [88]:
```
def foo():
    print("Hello Everyone! How are you doing?")
```

In [83]:
```
new_foo = pretty(foo)
```

In [84]:
```
new_foo()
```

```
--------------------------------------------------
Hello Everyon! How are you doing?
--------------------------------------------------
```

In [85]:
```
def bar():
    print("Random Cliche Statement!")
```

```
I AM AMAZED RIGHT NOW!! MIGHT JUST CRY!!!
--------------------------------------------------
```

Re-explain

In [108]:
```
def pretty(func):
    def inner():
        func()
    return inner
```

In [109]:
```
def foo():
    print("Hello Everyone! How are you?")
```

In [110]:
```
new_foo = pretty(foo)
```

In [111]:
```
new_foo()
```

```
Hello Everyone! How are you?
```

In [ ]:

```
I AM AMAZED RIGHT NOW!! MIGHT JUST CRY!!!
--------------------------------------------------
```

Re-explain

In [113]:
```
def pretty(func):
    def inner():
        print('-'*50)
        func()
        print('-'*50)
    return inner
```

In [114]:
```
def foo():
    print("Hello Everyone! How are you?")
```

In [117]:
```
new_foo = pretty(foo) # prettified version of foo
```

In [118]:
```
new_foo()
```

```
--------------------------------------------------
Hello Everyone! How are you?
--------------------------------------------------
```

In [ ]:

```
I AM AMAZED RIGHT NOW!! MIGHT JUST CRY!!!
--------------------------------------------------
```

Re-explain

In [113]:
```
def pretty(func):
    def inner():
        print('-'*50)
        func()
        print('-'*50)
    return inner
```

In [114]:
```
def foo():
    print("Hello Everyone! How are you?")
```

In [117]:
```
new_foo = pretty(foo) # prettified version of foo
```

In [118]:
```
new_foo()
```

```
--------------------------------------------------
Hello Everyone! How are you?
--------------------------------------------------
```

In [ ]:
```

```

In [150]:
```
magic = pretty(magic)
```

In [152]:
```
magic()
```

```
--------------------------------------------------
--------------------------------------------------
This is amazing!
--------------------------------------------------
--------------------------------------------------
```

In [158]:
```
def pretty(func):
    def inner(a):
        print('-'*50)
        func(a)
        print('-'*50)
    return inner
```

In [159]:
```
def foo(x):
    print(x**2)
```

In [160]:
```
foo = pretty(foo)
```

In [161]:
```
foo(5)
```

```
--------------------------------------------------
25
--------------------------------------------------
```