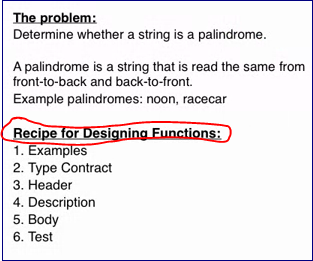
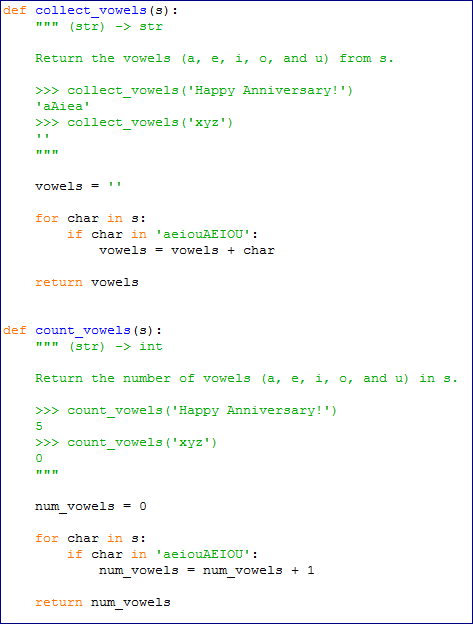
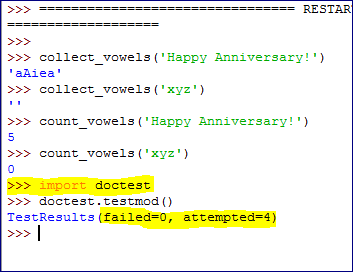
**Notes -- Crafting Quality Code -- LPT2**

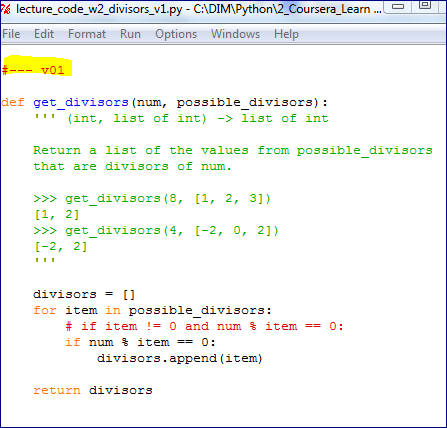


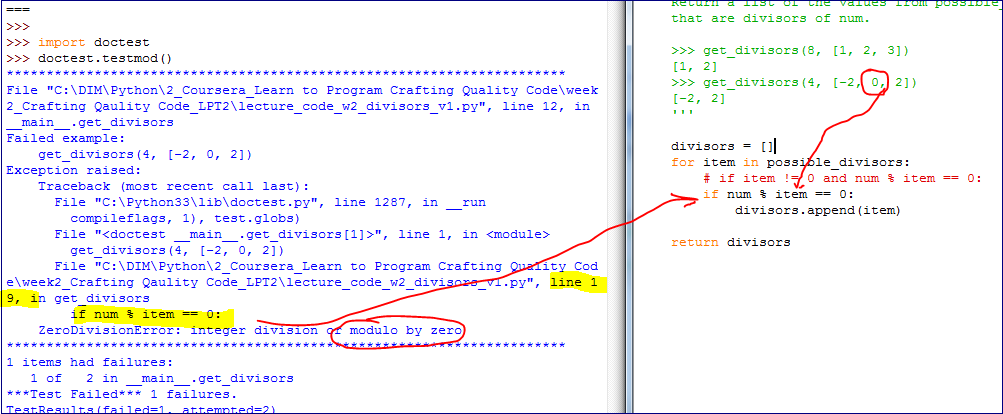
**Week2**

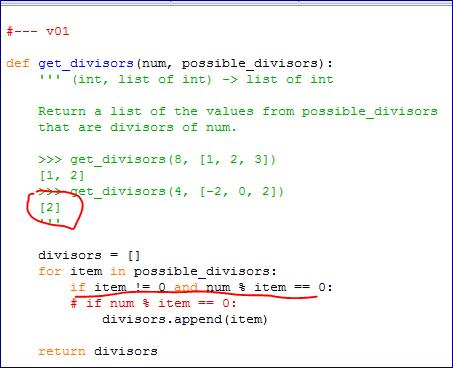
**Doc test.** Rather than running each example separately in the shell to test your function against the examples we may run them all at once.

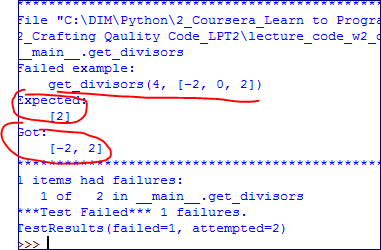


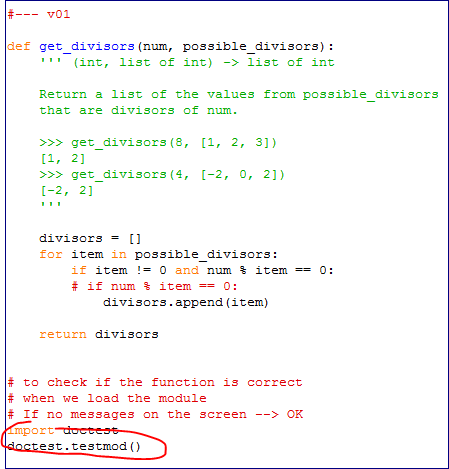


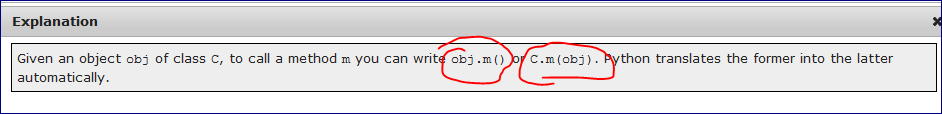


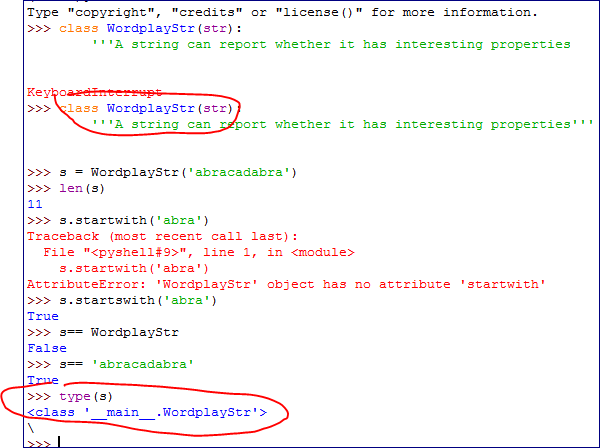


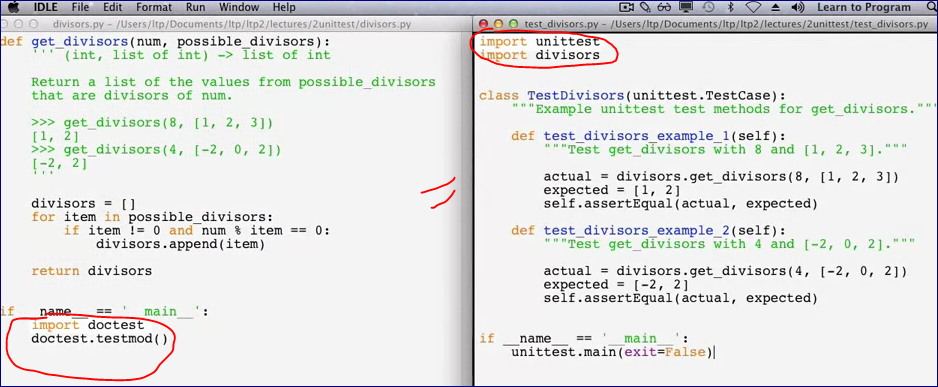


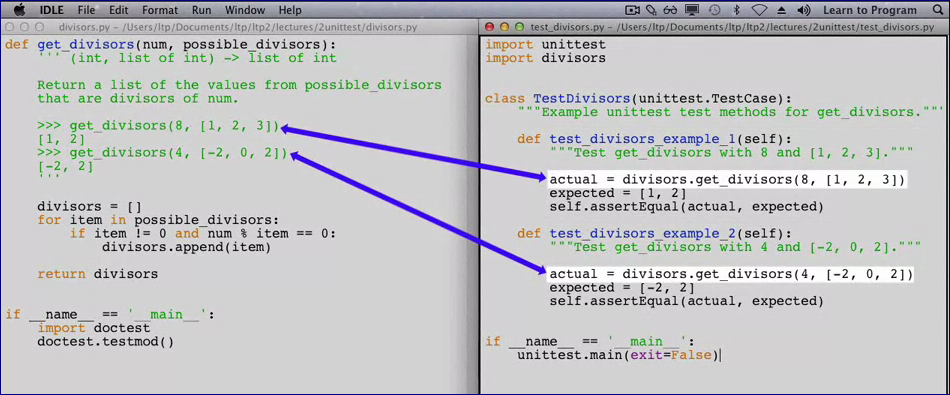


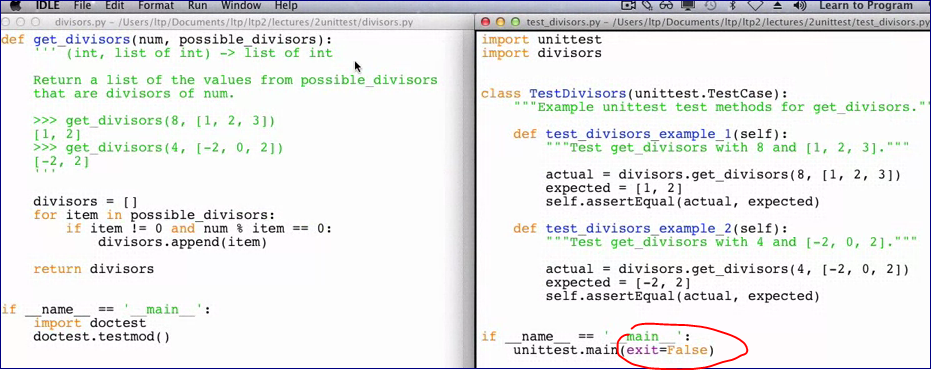


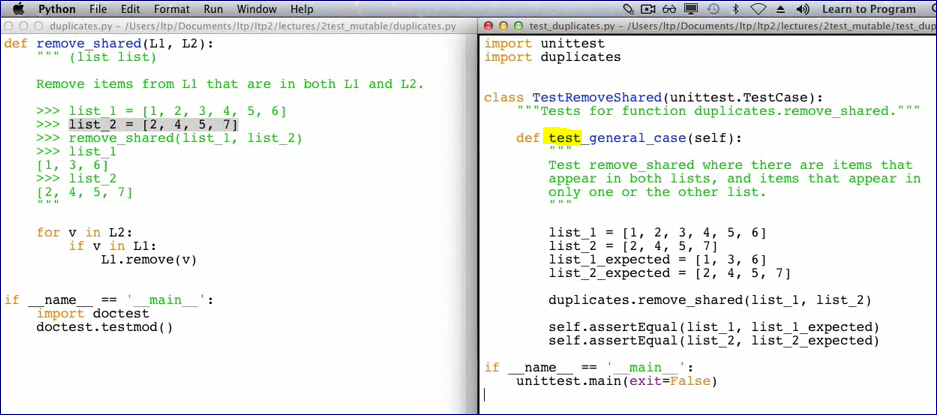


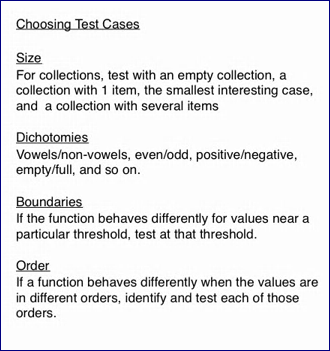






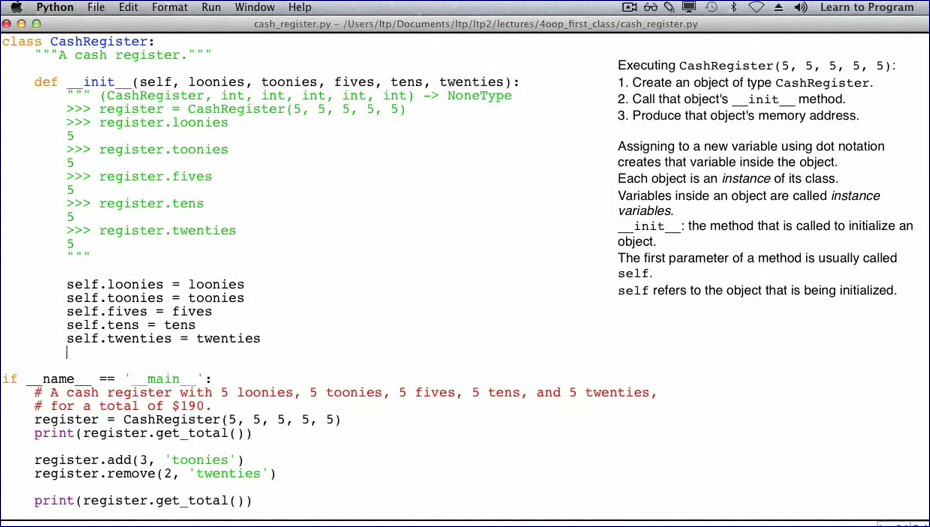


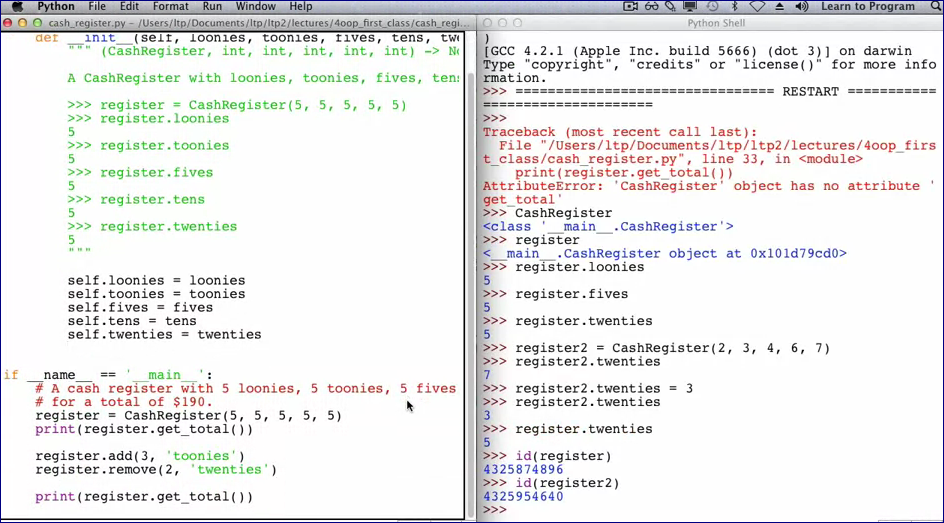


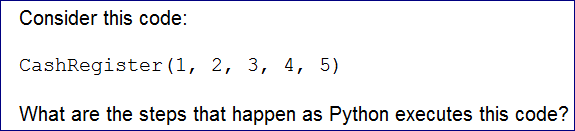


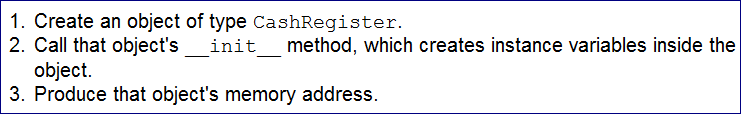
**Week4**

**Creating a New Type**

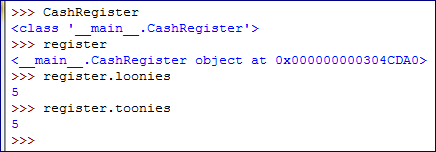




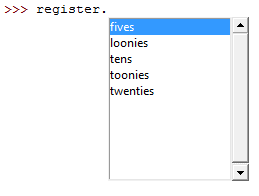




We created a class, an objects, and variables inside of the object.



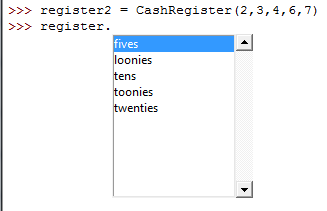
"register." shows the items inside of the object.



Let's create another object "register2"



New object also has all these variables

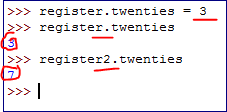


but with different values.

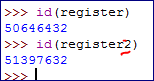


We can reassign the number of "twenties" in the 'register2' object

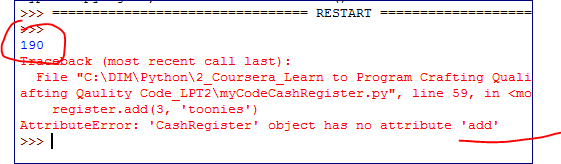
The number of 'twenties' in the first object stays the same

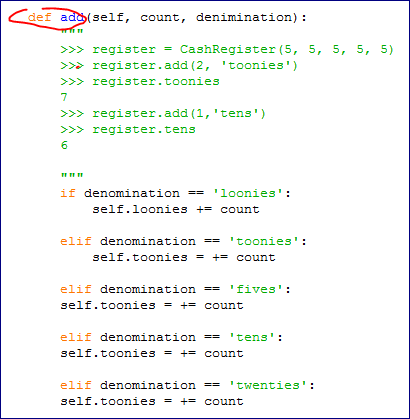


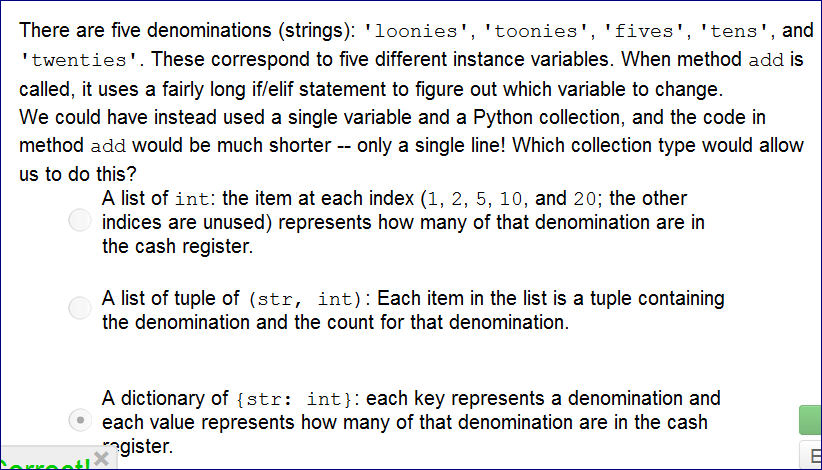
Because those two objects exist in different memory addresses

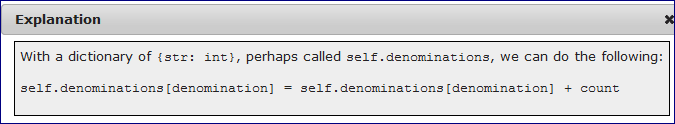


We got 190 from the get\_total method but still have an error from the "add" method.

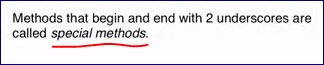


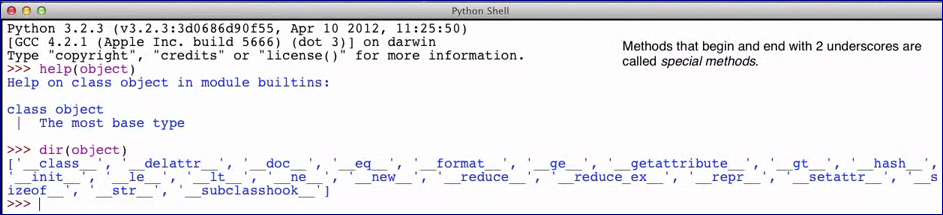


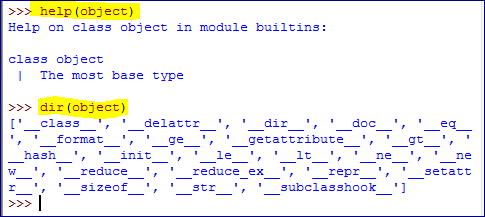


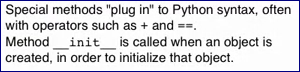


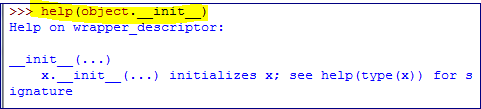
# Plugging Your Classes into Python Syntax

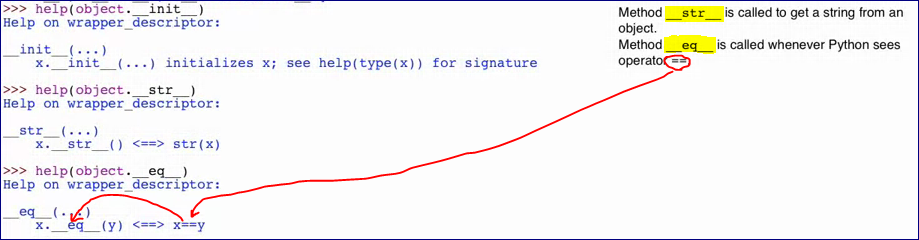




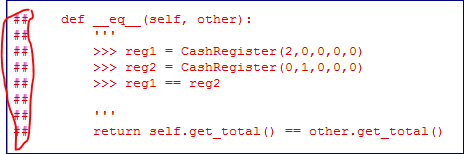




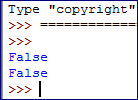




If we comment out our "compare" method

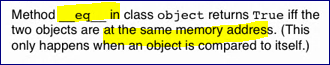


and run the program we get

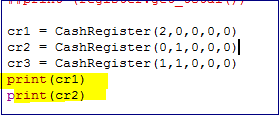


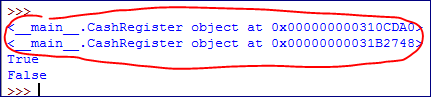
Because Python by default compares memory addresses for equality.

If no \_\_methods defined in the class it uses objects \_\_methods.



# Writing Special Method \_\_str\_\_

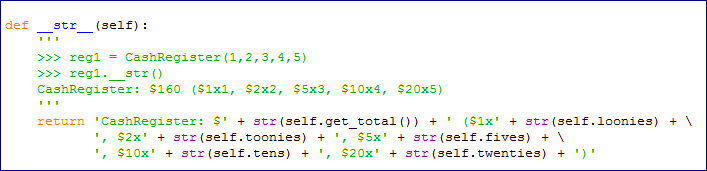




Because the function print() calls \_\_str\_\_ on the function it's printing



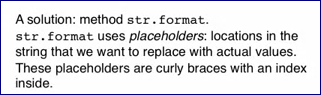
First solution:

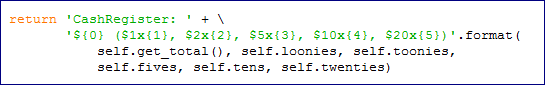




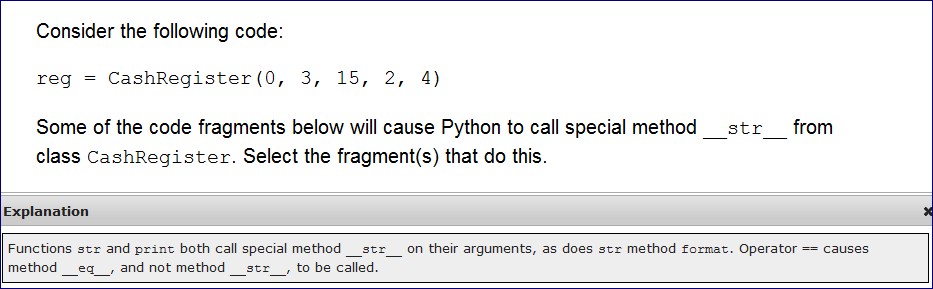


Another option is to use a **str.format** method.







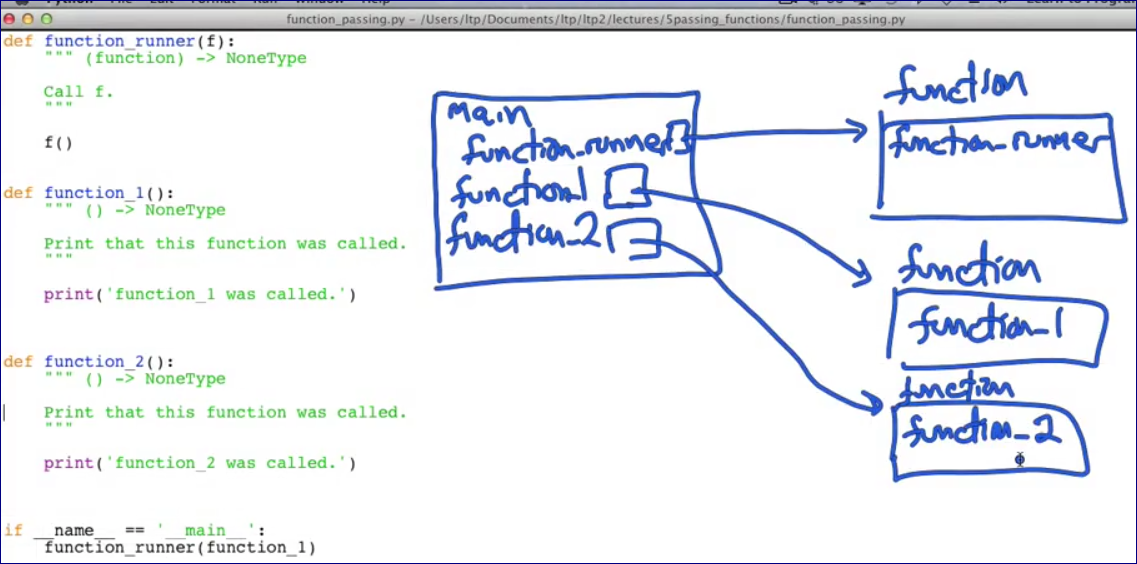


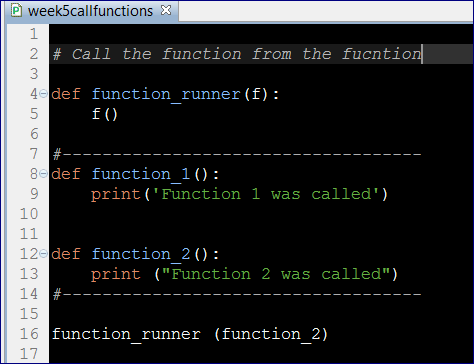
# Writing Classes That Interact

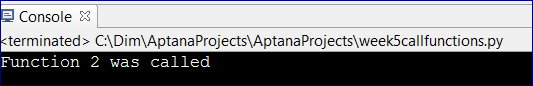
**Week5**

**Lecture 1 -- Passing functions as arguments**

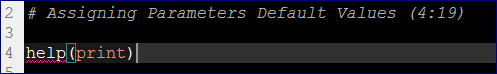
Call a function from a function

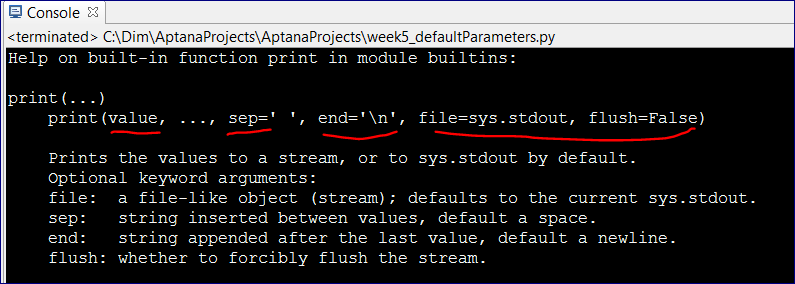


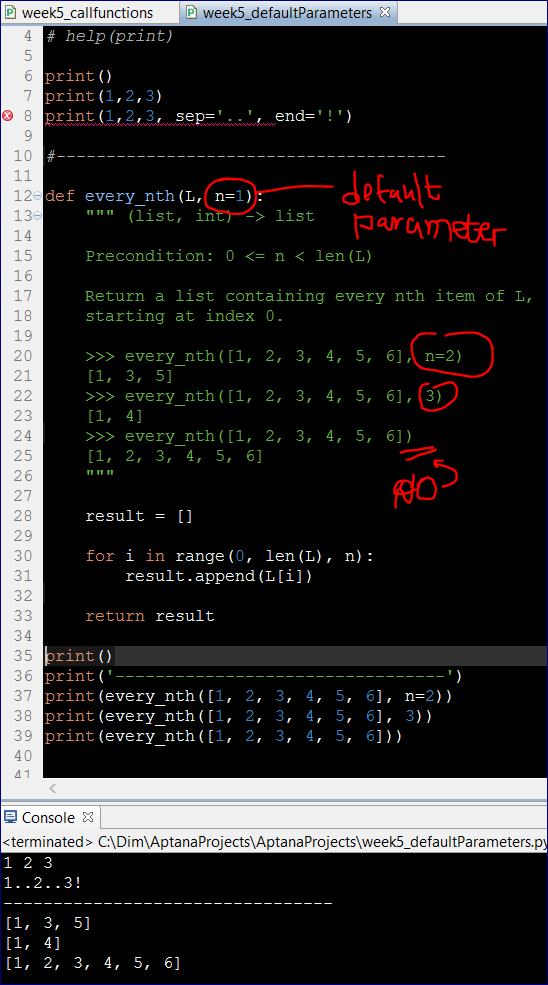


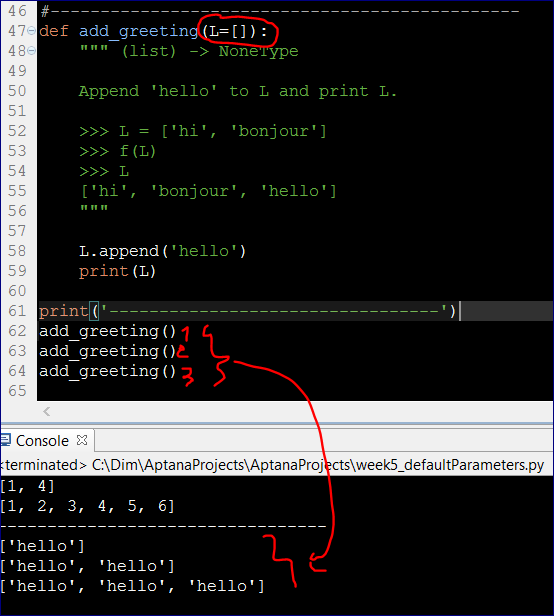


**Lecture 2 -- Assigning parameters default values**

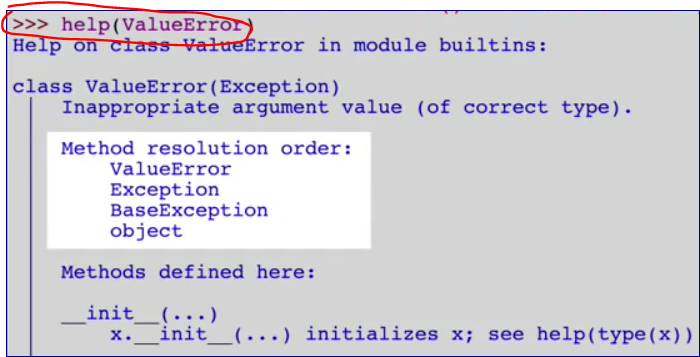


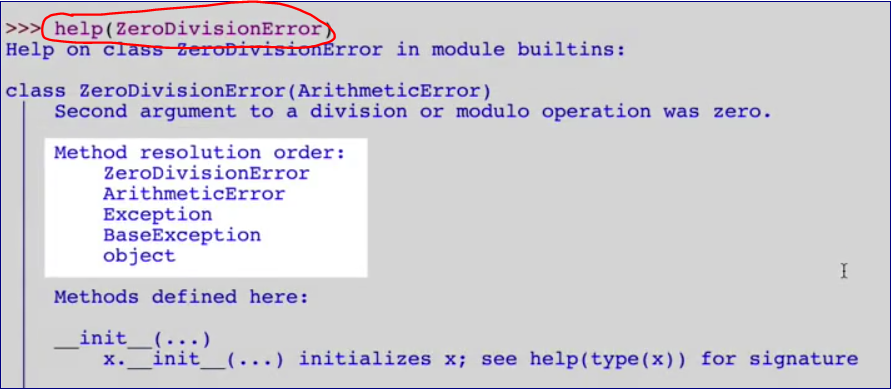


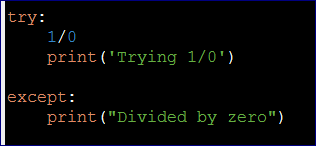




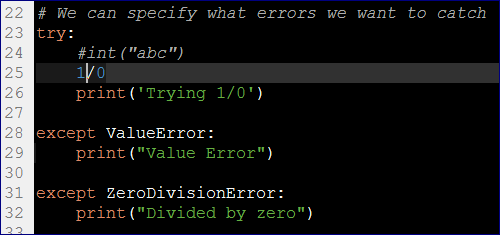
**Lecture 3 -- Dealing with Exceptional Situations (7:26)**



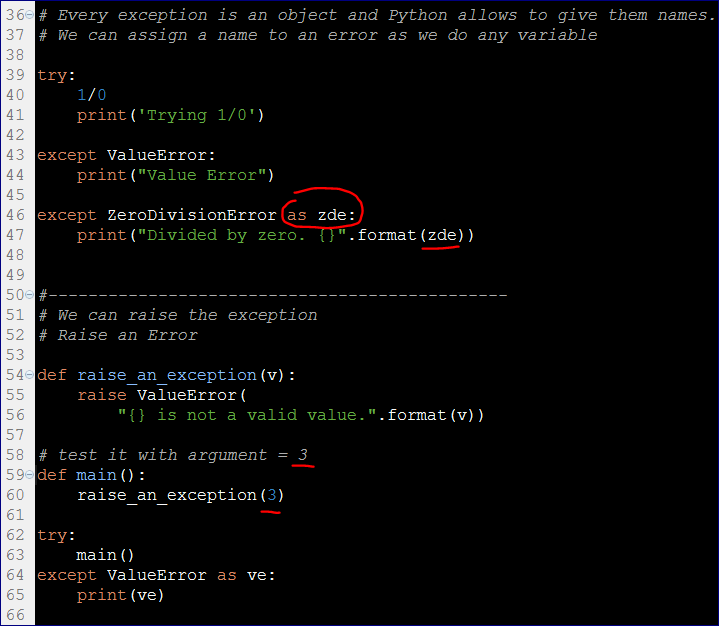




**We can specify what errors we want to catch**



Every exception is an object and Python allows to give them names.



#----------------------------------------------

# Assert

#----------------------------------------------

x = 3

assert x == 3

# assert x == 4

# We can use an assert statement to check preconditions

# To make sure the function uses a valid argument

