Review

by Deborah R. Fowler

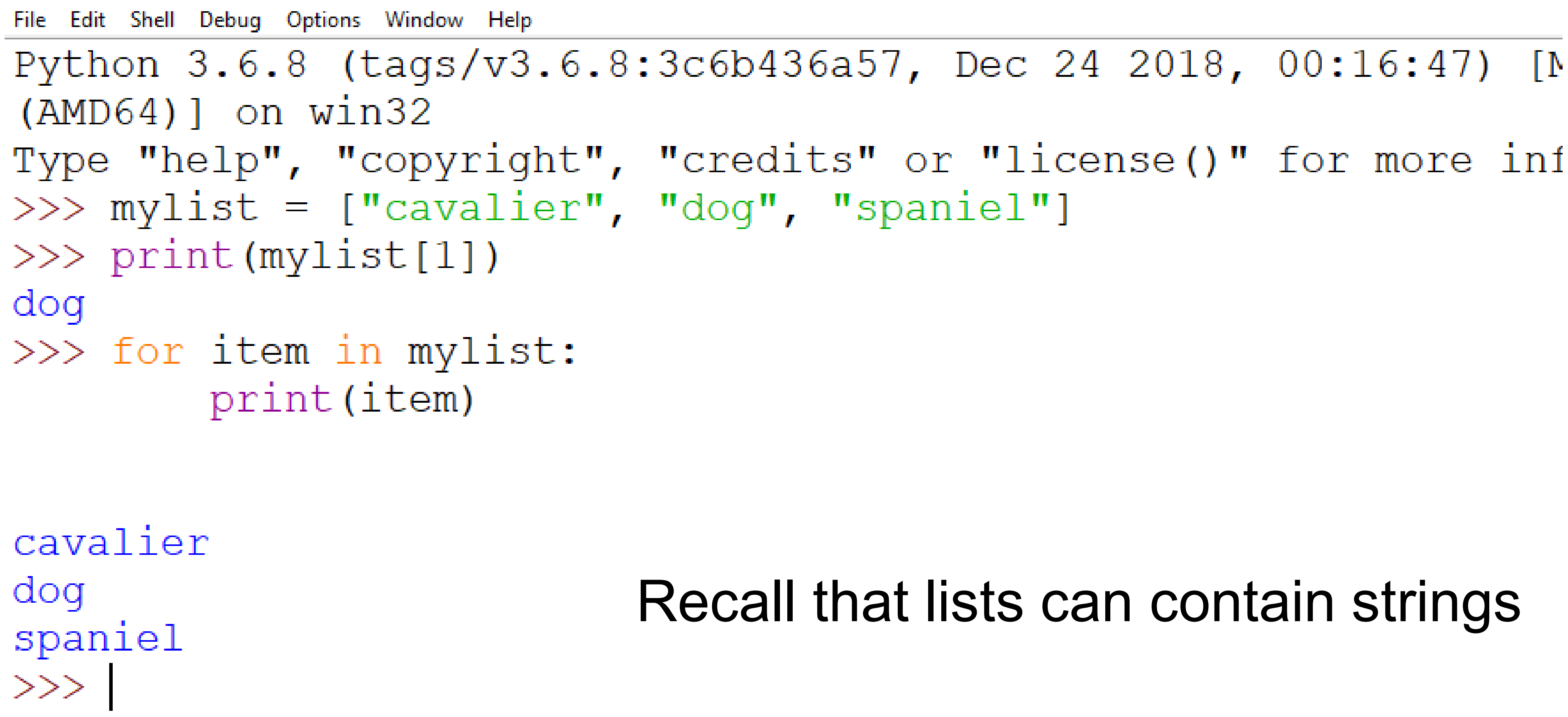
updated from python 2 to 3
ie. print() versus print

KEY CONCEPTS

- ✓ variables
- ✓ truth statements
- ✓ looping
- ✓ functions
- ✓ I/O
- ✓ lists
- classes/objects
- OOP

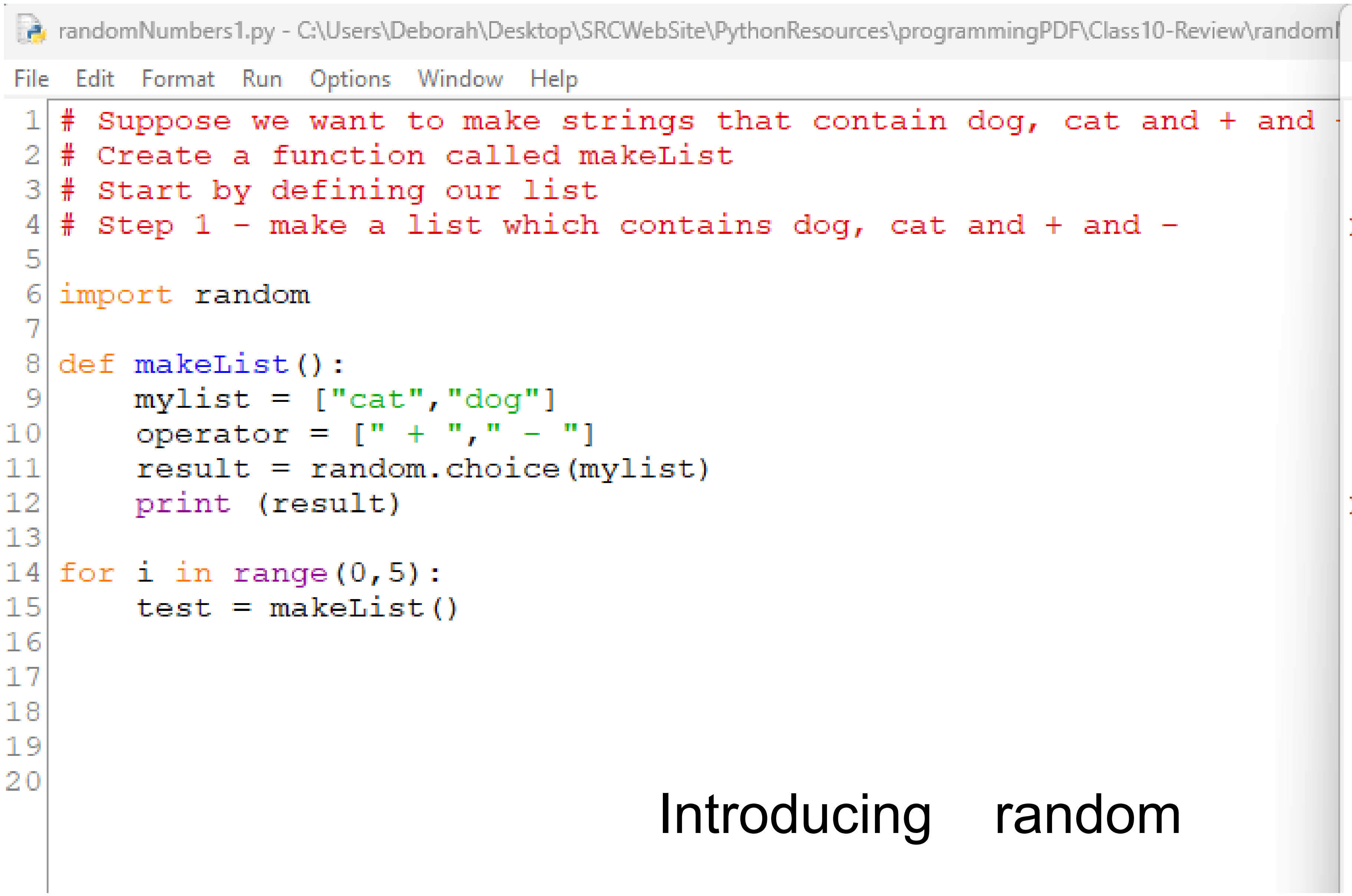
Today:

- E3 – strings – lists
- Review

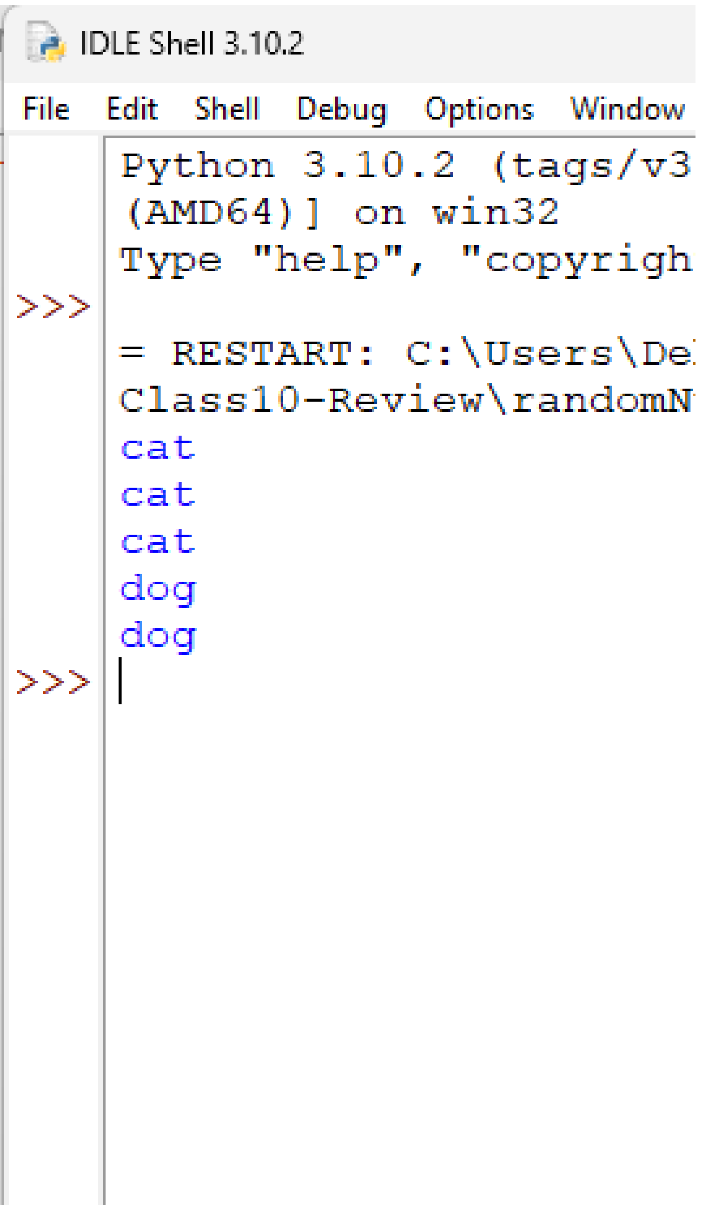


```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [M
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inf
>>> mylist = ["cavalier", "dog", "spaniel"]
>>> print(mylist[1])
dog
>>> for item in mylist:
        print(item)



cavalier
dog
spaniel
>>> 
```

Recall that lists can contain strings

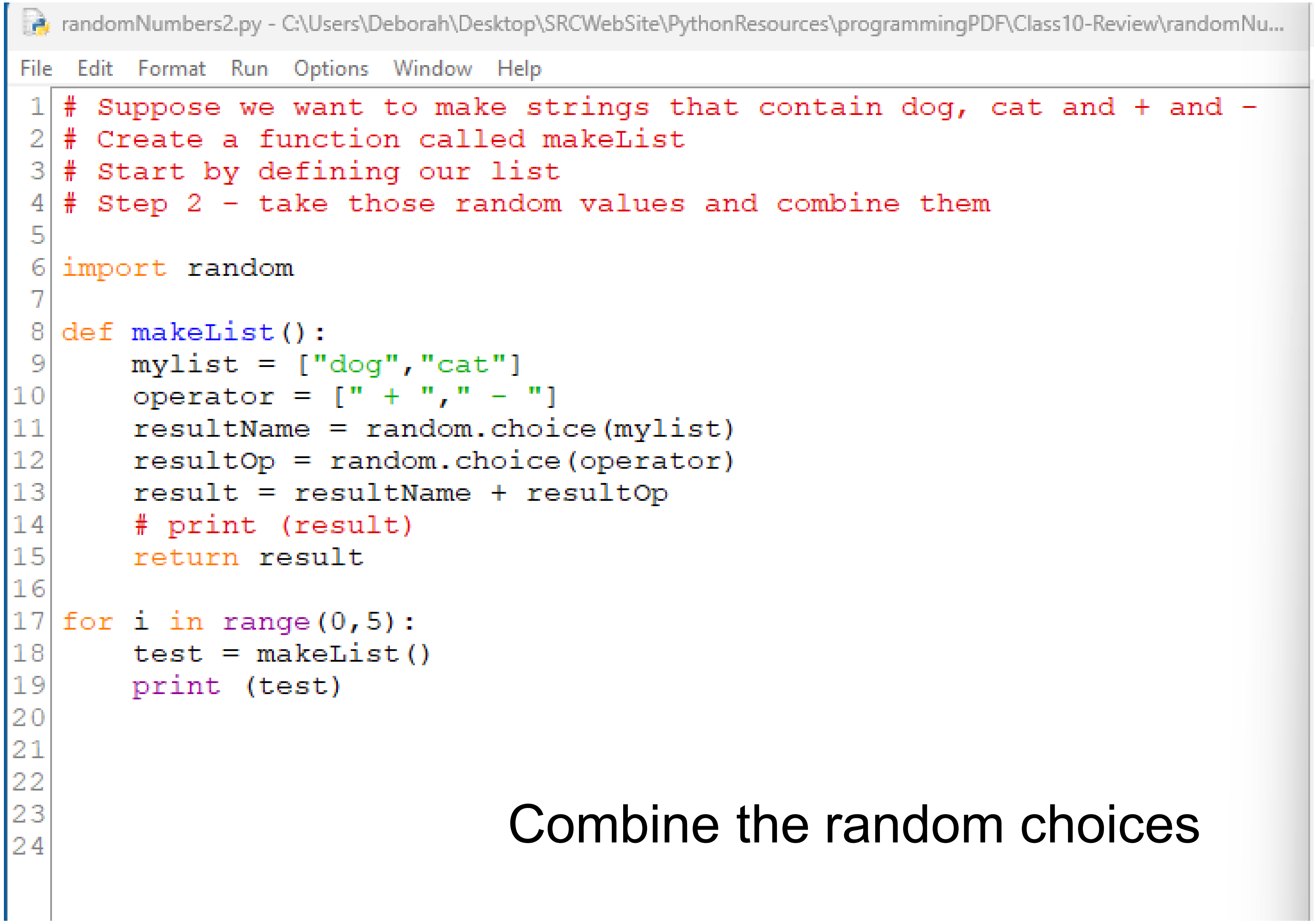

randomNumbers1.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomN
File Edit Format Run Options Window Help

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 1 - make a list which contains dog, cat and + and -

import random

def makeList():
    mylist = ["cat","dog"]
    operator = [" + "," - "]
    result = random.choice(mylist)
    print (result)

for i in range(0,5):
    test = makeList()
```

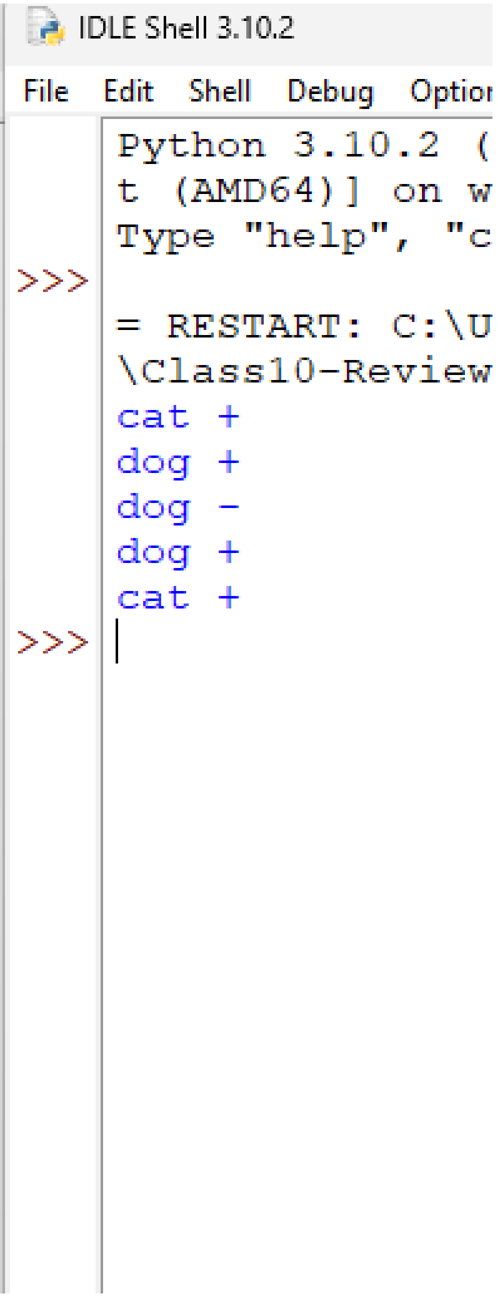


```
    Python 3.10.2 (tags/v3
    (AMD64)] on win32
    Type "help", "copyrigh
>>>
    = RESTART: C:\Users\De
    Class10-Review\randomN
    cat
    cat
    cat
    dog
    dog
>>> 
```

Introducing random

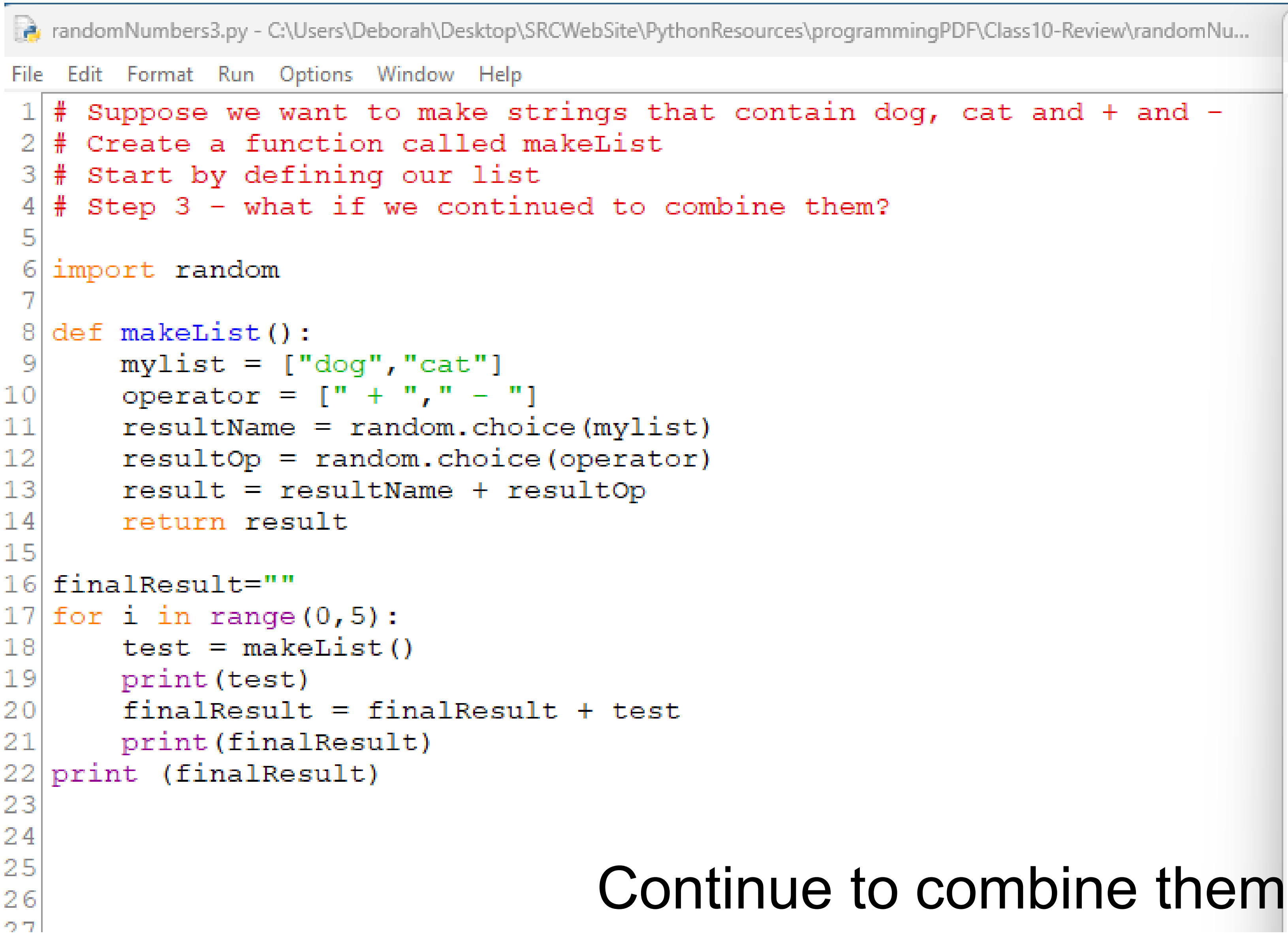

randomNumbers2.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNu...
File Edit Format Run Options Window Help

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 2 - take those random values and combine them

import random

def makeList():
    mylist = ["dog","cat"]
    operator = [" + "," - "]
    resultName = random.choice(mylist)
    resultOp = random.choice(operator)
    result = resultName + resultOp
    # print (result)
    return result

for i in range(0,5):
    test = makeList()
    print (test)
```



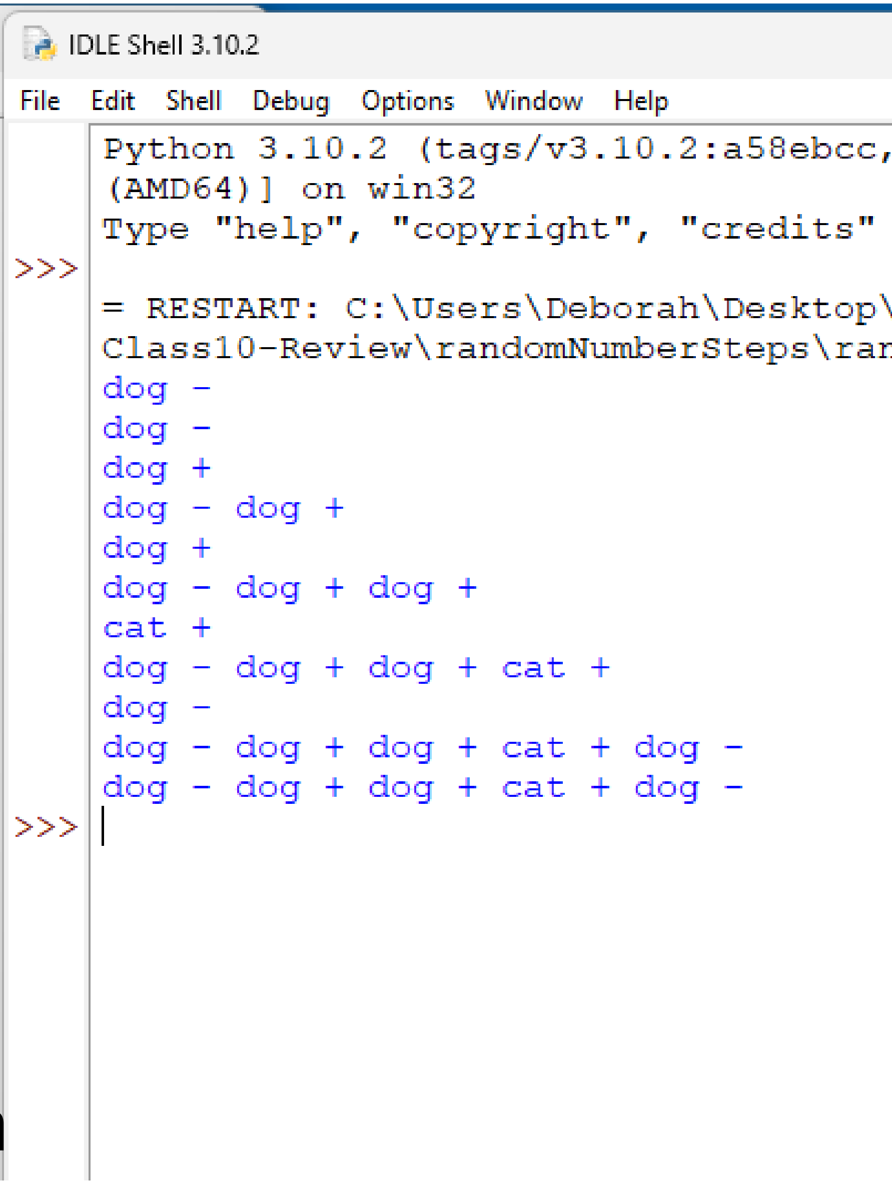
```
    Python 3.10.2 (
    t (AMD64)] on w
    Type "help", "c
>>>
    = RESTART: C:\U
    \Class10-Review
    cat +
    dog +
    dog -
    dog +
    cat +
>>> 
```

Combine the random choices



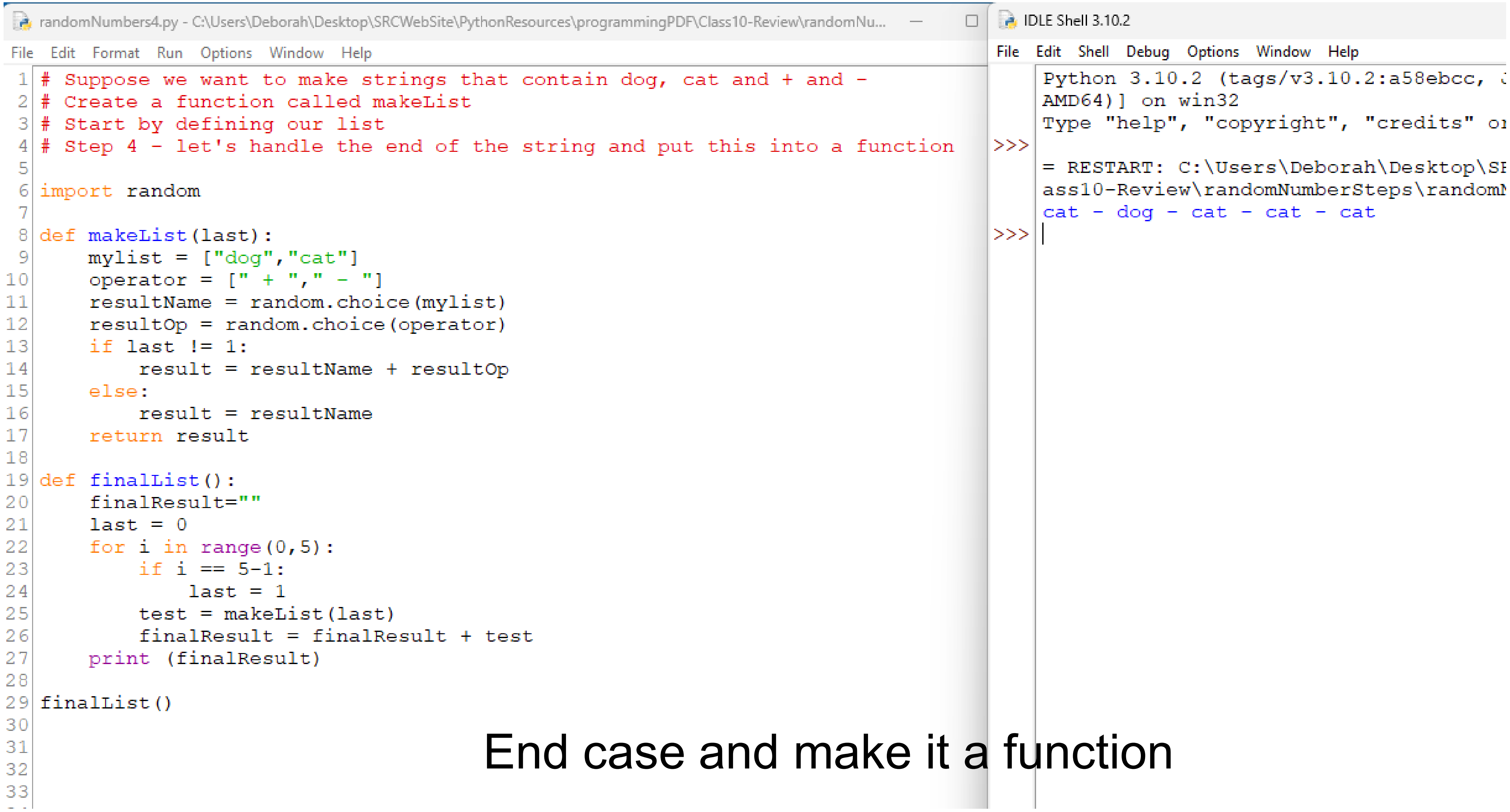
randomNumbers3.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNu...
File Edit Format Run Options Window Help

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 3 - what if we continued to combine them?

import random

def makeList():
    mylist = ["dog","cat"]
    operator = [" + "," - "]
    resultName = random.choice(mylist)
    resultOp = random.choice(operator)
    result = resultName + resultOp
    return result

finalResult=""
for i in range(0,5):
    test = makeList()
    print(test)
    finalResult = finalResult + test
    print(finalResult)
print (finalResult)
```



```
Python 3.10.2 (tags/v3.10.2:a58ebcc,
(AMD64)] on win32
Type "help", "copyright", "credits"
>>>
= RESTART: C:\Users\Deborah\Desktop\
Class10-Review\randomNumberSteps\ran
dog -
dog -
dog +
dog - dog +
dog +
dog - dog + dog +
cat +
dog - dog + dog + cat +
dog -
dog - dog + dog + cat + dog -
dog - dog + dog + cat + dog -
>>>
```

Continue to combine them

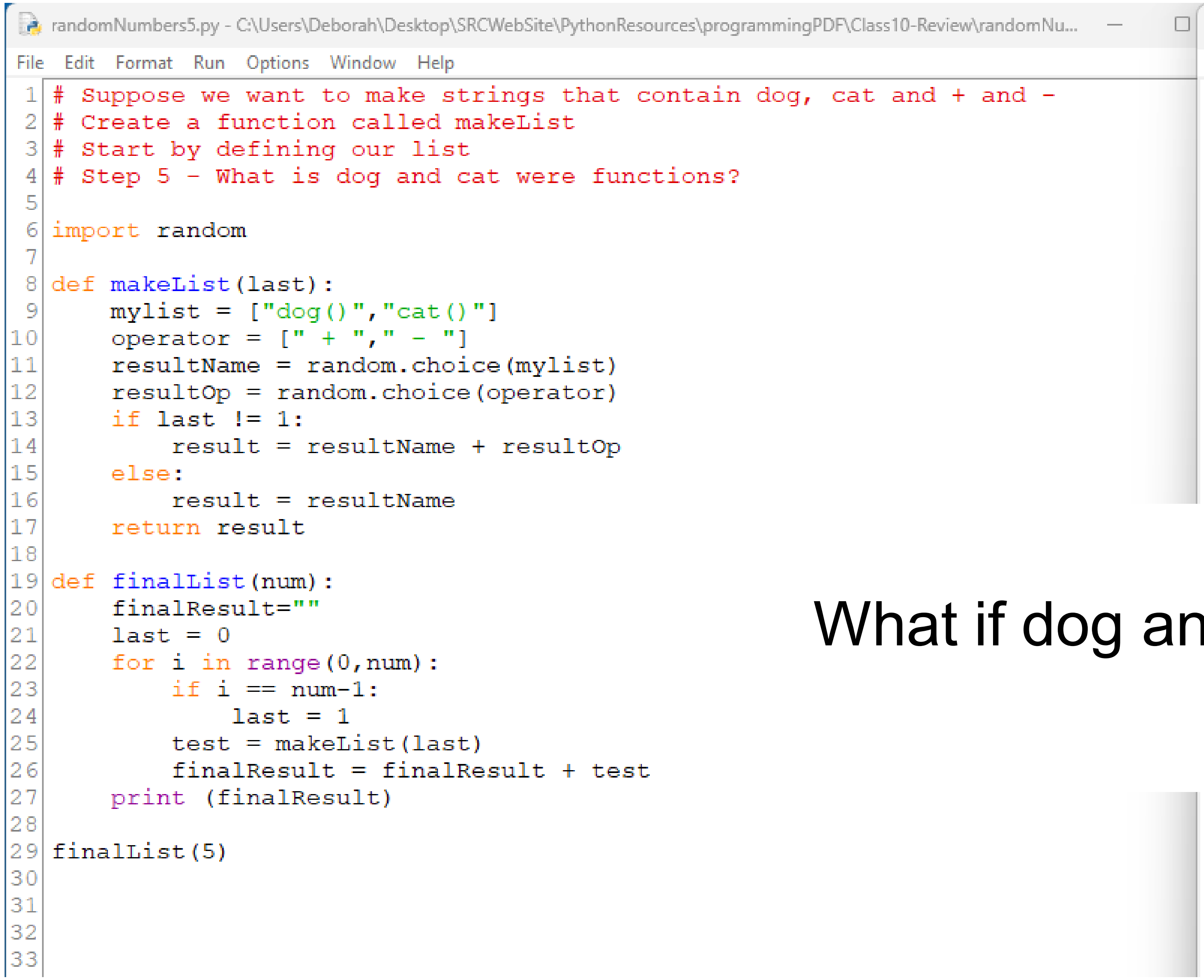

randomNumbers4.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNu...

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 4 - let's handle the end of the string and put this into a function

import random

def makeList(last):
    mylist = ["dog","cat"]
    operator = [" + "," - "]
    resultName = random.choice(mylist)
    resultOp = random.choice(operator)
    if last != 1:
        result = resultName + resultOp
    else:
        result = resultName
    return result

def finalList():
    finalResult=""
    last = 0
    for i in range(0,5):
        if i == 5-1:
            last = 1
        test = makeList(last)
        finalResult = finalResult + test
    print (finalResult)

finalList()
```

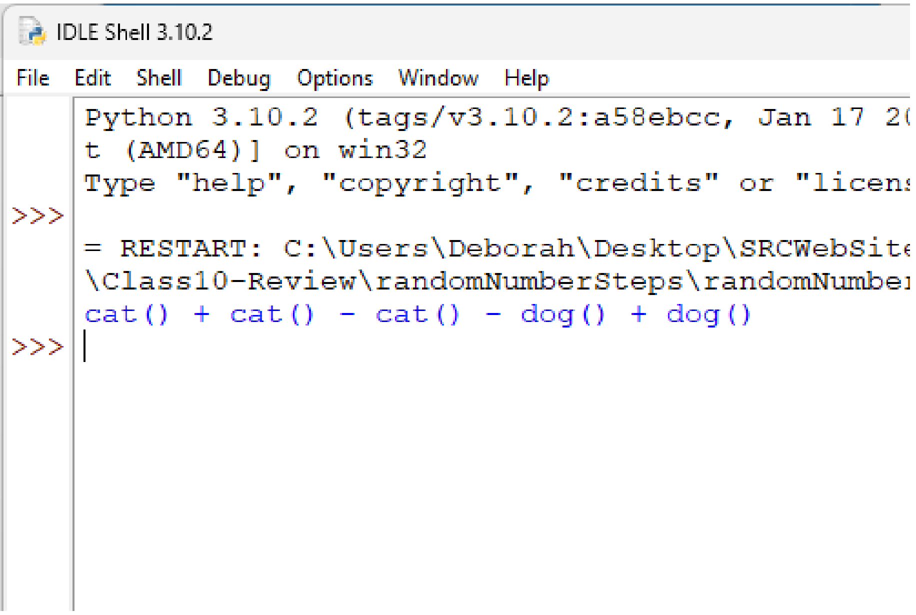
IDLE Shell 3.10.2

```
Python 3.10.2 (tags/v3.10.2:a58ebcc,
AMD64)] on win32
Type "help", "copyright", "credits" o
>>>
= RESTART: C:\Users\Deborah\Desktop\S
ass10-Review\randomNumberSteps\random
cat - dog - cat - cat - cat
>>>
```

End case and make it a function

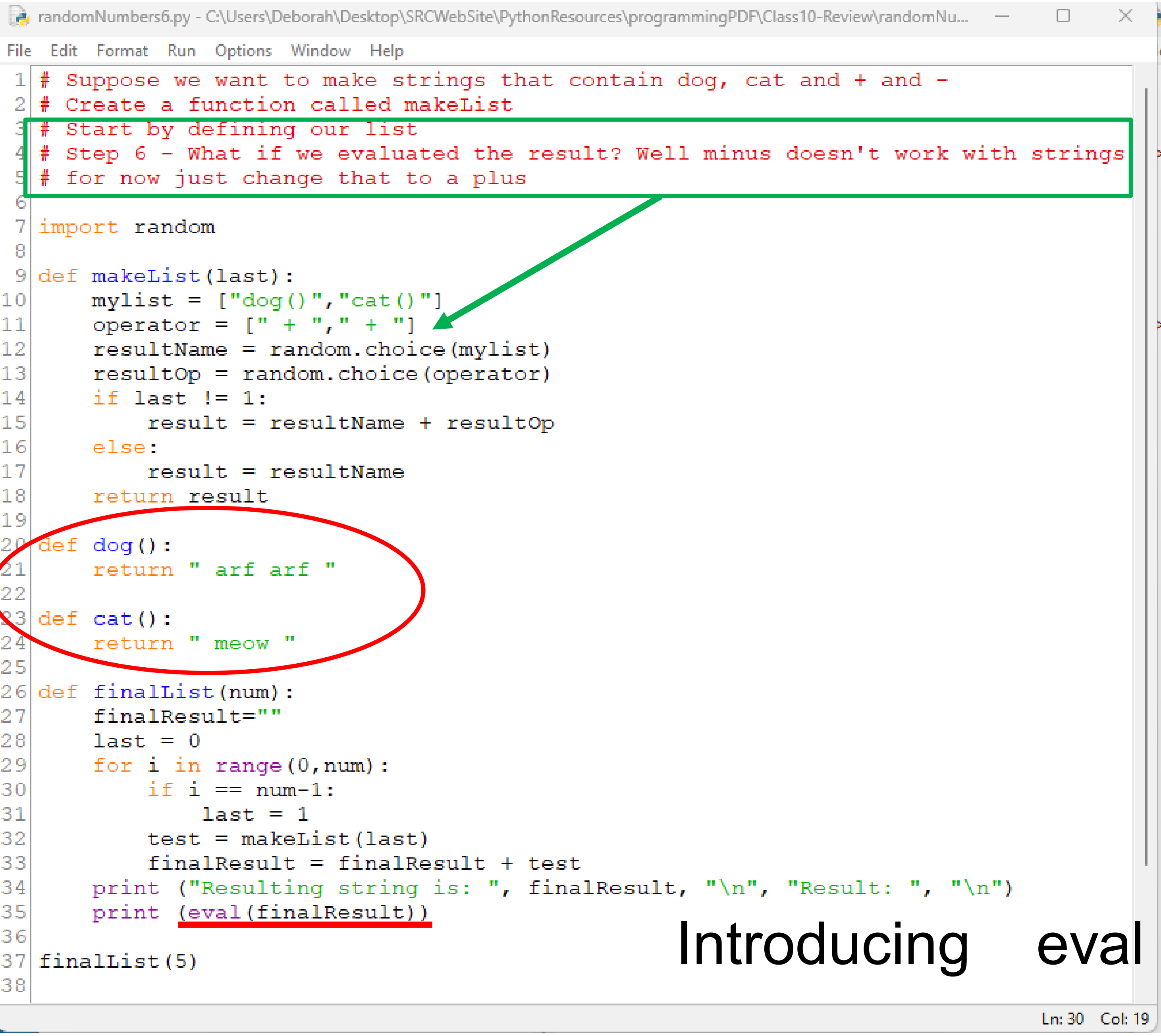

randomNumbers5.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNu...

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 5 - What is dog and cat were functions?

import random

def makeList(last):
    mylist = ["dog()","cat()"]
    operator = [" + "," - "]
    resultName = random.choice(mylist)
    resultOp = random.choice(operator)
    if last != 1:
        result = resultName + resultOp
    else:
        result = resultName
    return result

def finalList(num):
    finalResult=""
    last = 0
    for i in range(0,num):
        if i == num-1:
            last = 1
        test = makeList(last)
        finalResult = finalResult + test
    print (finalResult)

finalList(5)
```

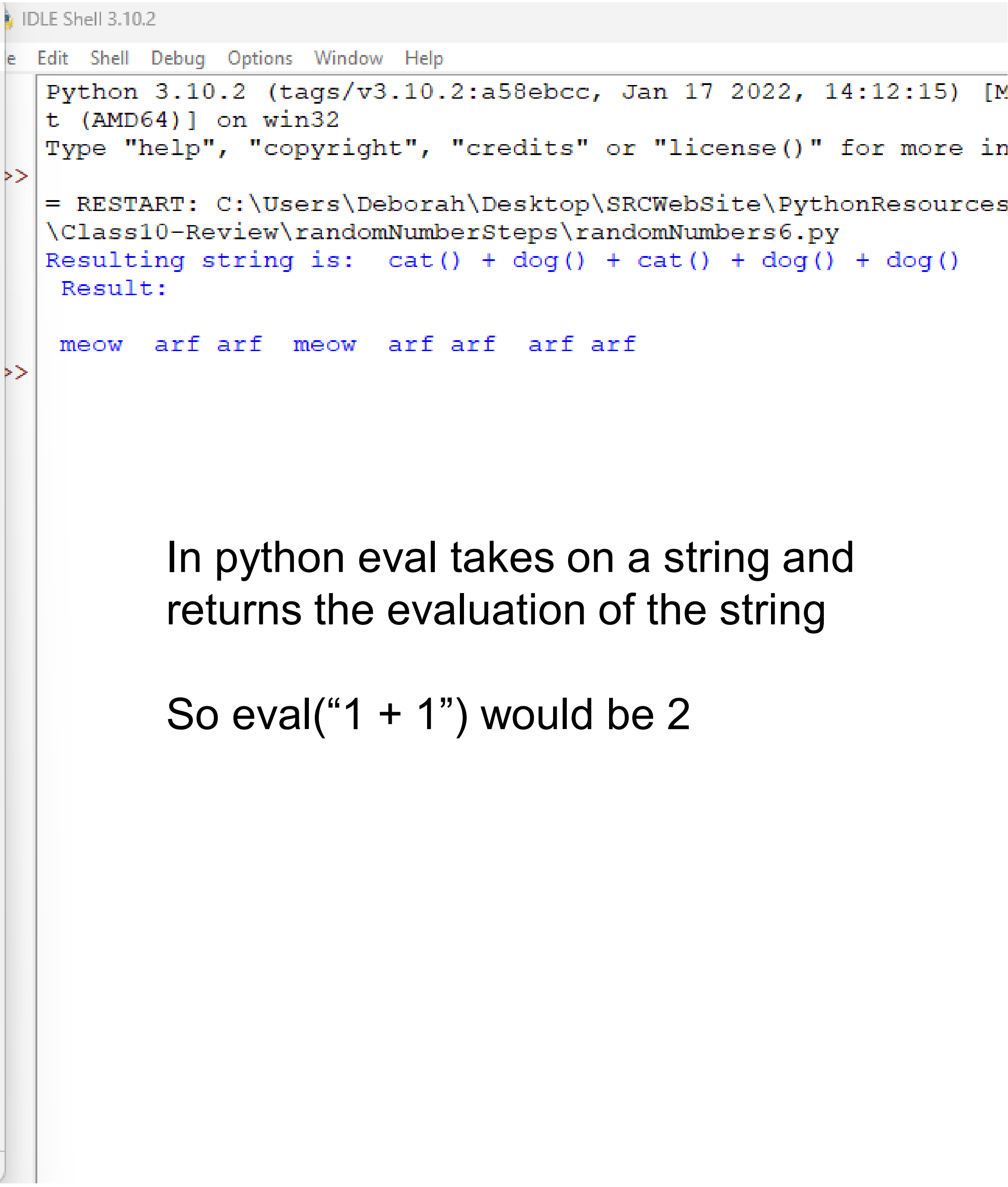

IDLE Shell 3.10.2

```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2(
t (AMD64)] on win32
Type "help", "copyright", "credits" or "licens
>>>
= RESTART: C:\Users\Deborah\Desktop\SRCWebSite
\Class10-Review\randomNumberSteps\randomNumber
cat() + cat() - cat() - dog() + dog()
>>> 
```

What if dog and cat were functions?


randomNumbers6.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNu...
File Edit Format Run Options Window Help
Ln: 30 Col: 19

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 6 - What if we evaluated the result? Well minus doesn't work with strings
# for now just change that to a plus

import random

def makeList(last):
    mylist = ["dog()","cat()"]
    operator = [" + "," + "]
    resultName = random.choice(mylist)
    resultOp = random.choice(operator)
    if last != 1:
        result = resultName + resultOp
    else:
        result = resultName
    return result

def dog():
    return " arf arf "

def cat():
    return " meow "

def finalList(num):
    finalResult=""
    last = 0
    for i in range(0,num):
        if i == num-1:
            last = 1
        test = makeList(last)
        finalResult = finalResult + test
    print ("Resulting string is: ", finalResult, "\n", "Result: ", "\n")
    print (eval(finalResult))

finalList(5)
```

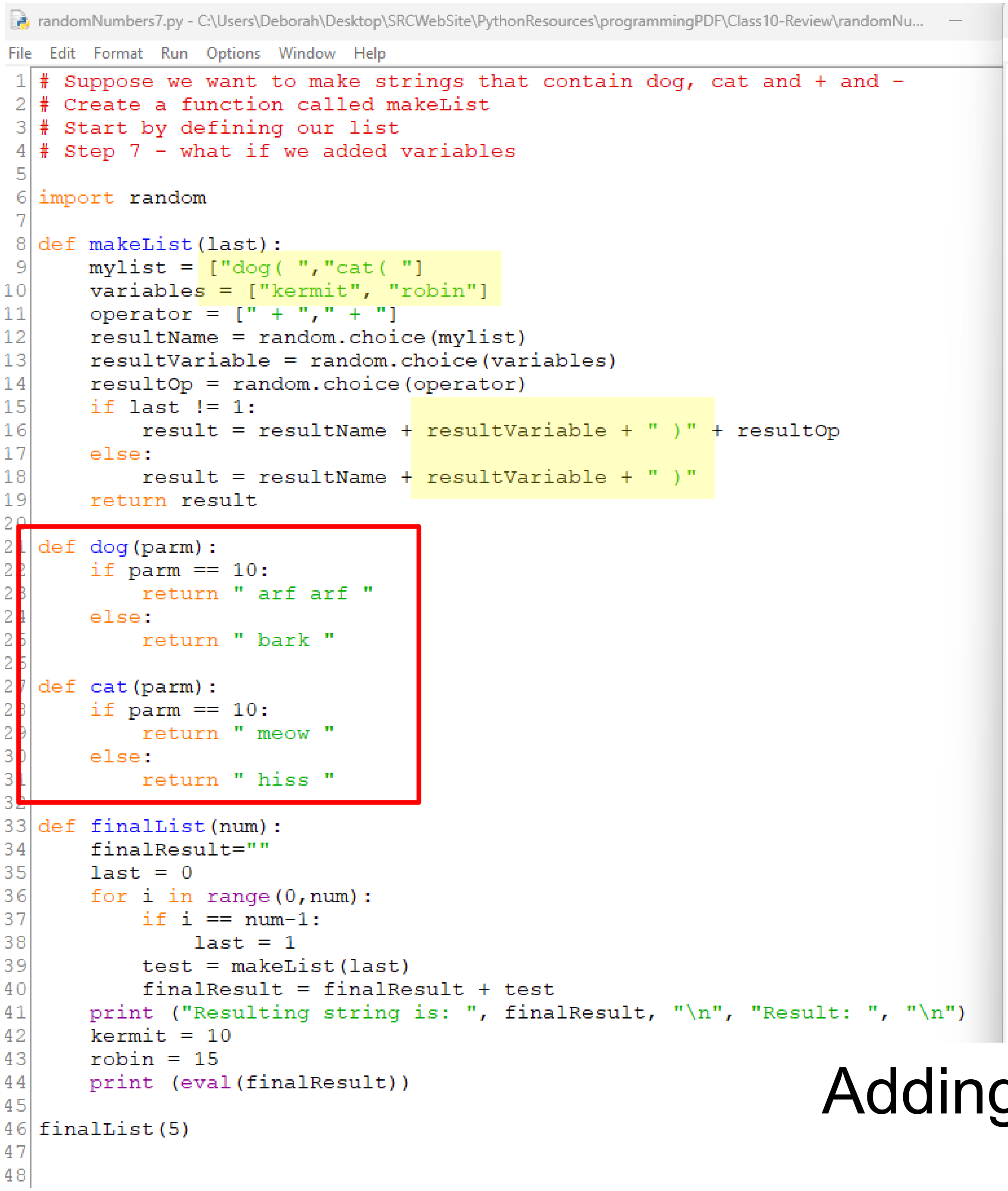


```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [M
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
>>
= RESTART: C:\Users\Deborah\Desktop\SRCWebSite\PythonResources
\Class10-Review\randomNumberSteps\randomNumbers6.py
Resulting string is:  cat() + dog() + cat() + dog() + dog()
 Result:

 meow  arf arf  meow  arf arf  arf arf
>>
```

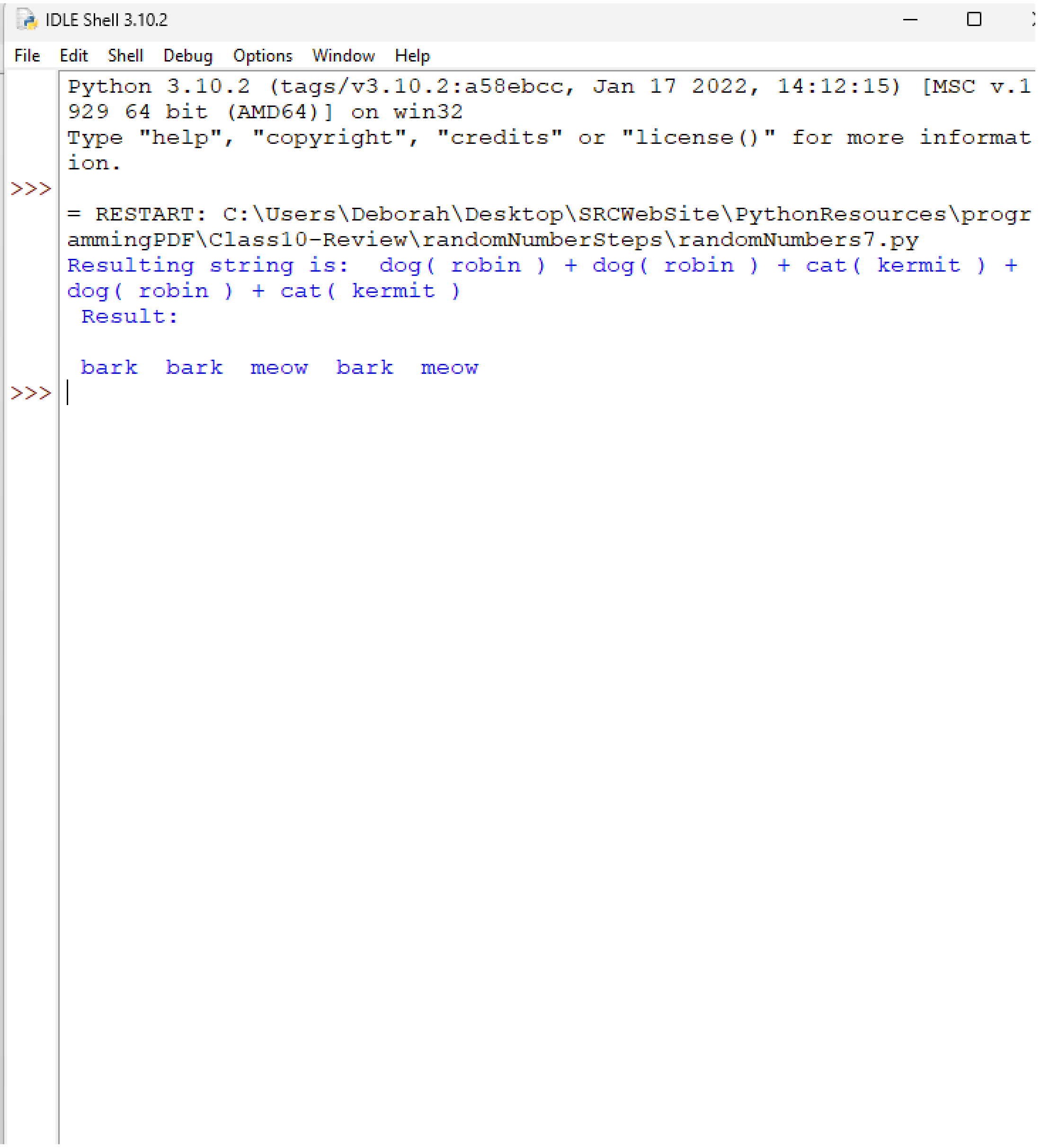
In python eval takes on a string and returns the evaluation of the string

So eval(“1 + 1”) would be 2

Introducing eval

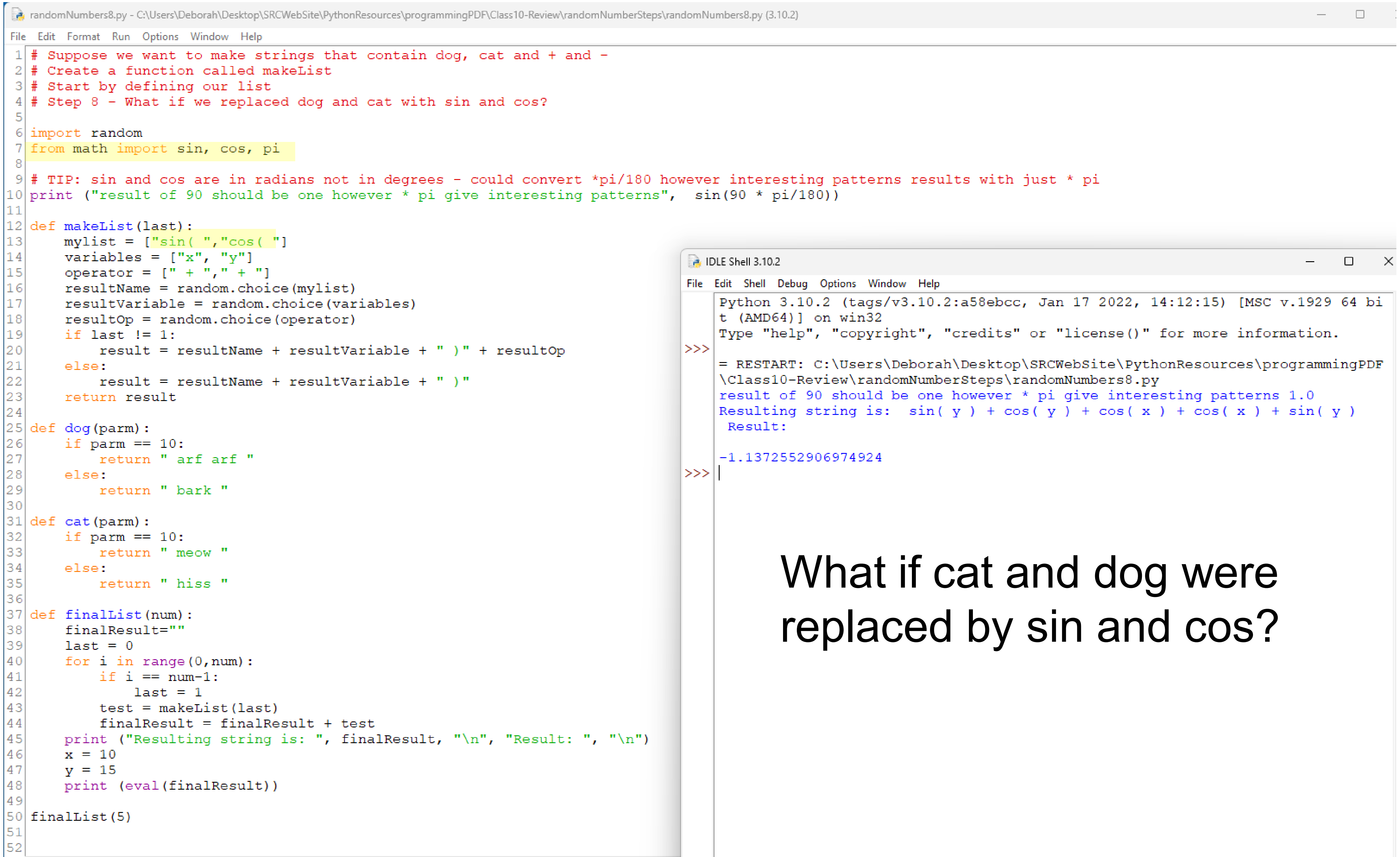

randomNumbers7.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNu...
File Edit Format Run Options Window Help

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 7 - what if we added variables

import random

def makeList(last):
    mylist = ["dog( ","cat( "]
    variables = ["kermit", "robin"]
    operator = [" + "," + "]
    resultName = random.choice(mylist)
    resultVariable = random.choice(variables)
    resultOp = random.choice(operator)
    if last != 1:
        result = resultName + resultVariable + " )" + resultOp
    else:
        result = resultName + resultVariable + " )"
    return result

def dog(parm):
    if parm == 10:
        return " arf arf "
    else:
        return " bark "

def cat(parm):
    if parm == 10:
        return " meow "
    else:
        return " hiss "

def finalList(num):
    finalResult=""
    last = 0
    for i in range(0,num):
        if i == num-1:
            last = 1
        test = makeList(last)
        finalResult = finalResult + test
    print ("Resulting string is: ", finalResult, "\n", "Result: ", "\n")
    kermit = 10
    robin = 15
    print (eval(finalResult))

finalList(5)
```



```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1
929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informat
ion.
>>>
= RESTART: C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\progr
ammingPDF\Class10-Review\randomNumberSteps\randomNumbers7.py
Resulting string is:  dog( robin ) + dog( robin ) + cat( kermit ) +
dog( robin ) + cat( kermit )
 Result:

 bark  bark  meow  bark  meow
>>> |
```

Adding variables


randomNumbers8.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\randomNumberSteps\randomNumbers8.py (3.10.2)
File Edit Format Run Options Window Help

```python
# Suppose we want to make strings that contain dog, cat and + and -
# Create a function called makeList
# Start by defining our list
# Step 8 - What if we replaced dog and cat with sin and cos?

import random
from math import sin, cos, pi

# TIP: sin and cos are in radians not in degrees - could convert *pi/180 however interesting patterns results with just * pi
print ("result of 90 should be one however * pi give interesting patterns",  sin(90 * pi/180))

def makeList(last):
    mylist = ["sin( ","cos( "]
    variables = ["x", "y"]
    operator = [" + "," + "]
    resultName = random.choice(mylist)
    resultVariable = random.choice(variables)
    resultOp = random.choice(operator)
    if last != 1:
        result = resultName + resultVariable + " )" + resultOp
    else:
        result = resultName + resultVariable + " )"
    return result

def dog(parm):
    if parm == 10:
        return " arf arf "
    else:
        return " bark "

def cat(parm):
    if parm == 10:
        return " meow "
    else:
        return " hiss "

def finalList(num):
    finalResult=""
    last = 0
    for i in range(0,num):
        if i == num-1:
            last = 1
        test = makeList(last)
        finalResult = finalResult + test
    print ("Resulting string is: ", finalResult, "\n", "Result: ", "\n")
    x = 10
    y = 15
    print (eval(finalResult))

finalList(5)
```

```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF
\Class10-Review\randomNumberSteps\randomNumbers8.py
result of 90 should be one however * pi give interesting patterns 1.0
Resulting string is:  sin( y ) + cos( y ) + cos( x ) + cos( x ) + sin( y )
 Result:

-1.1372552906974924
>>>
```

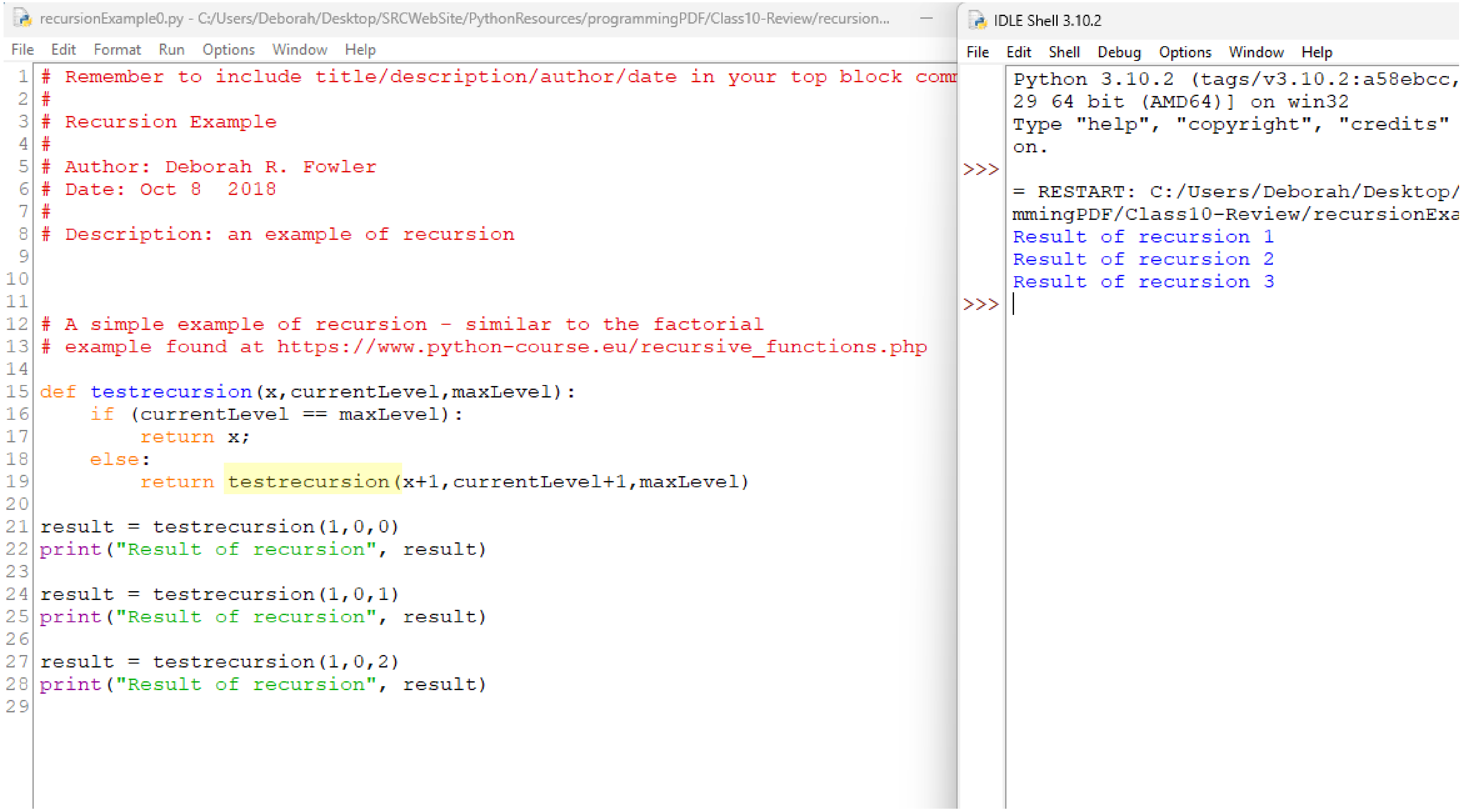
What if cat and dog were replaced by sin and cos?

In-class Exercise

- Create a string that is comprised of the words cat, dog and operators + and –
- Next try a cat and dog function,
- Remember to change to + and + for cat and dog
- Evaluate your string

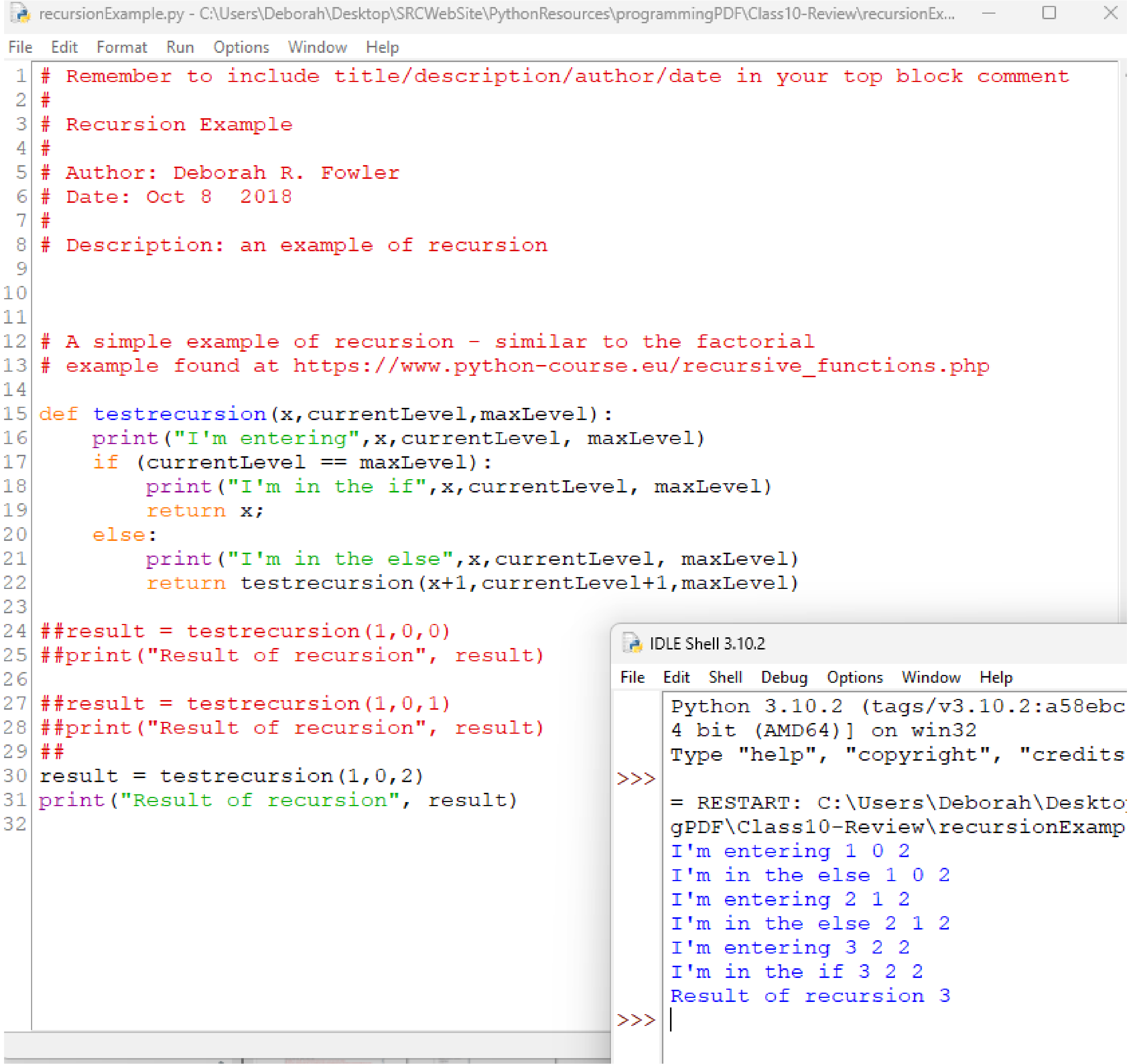
Now change it to two other animals and hand in to Dailies/FirstnameAnimalSounds

Recursion


recursionExample0.py - C:/Users/Deborah/Desktop/SRCWebSite/PythonResources/programmingPDF/Class10-Review/recursion...
File Edit Format Run Options Window Help
IDLE Shell 3.10.2
File Edit Shell Debug Options Window Help

```python
# Remember to include title/description/author/date in your top block comm
#
# Recursion Example
#
# Author: Deborah R. Fowler
# Date: Oct 8  2018
#
# Description: an example of recursion



# A simple example of recursion - similar to the factorial
# example found at https://www.python-course.eu/recursive_functions.php

def testrecursion(x,currentLevel,maxLevel):
    if (currentLevel == maxLevel):
        return x;
    else:
        return testrecursion(x+1,currentLevel+1,maxLevel)

result = testrecursion(1,0,0)
print("Result of recursion", result)

result = testrecursion(1,0,1)
print("Result of recursion", result)

result = testrecursion(1,0,2)
print("Result of recursion", result)
```

```
Python 3.10.2 (tags/v3.10.2:a58ebcc,
29 64 bit (AMD64)] on win32
Type "help", "copyright", "credits"
on.
>>>
= RESTART: C:/Users/Deborah/Desktop/
mmingPDF/Class10-Review/recursionExa
Result of recursion 1
Result of recursion 2
Result of recursion 3
>>> |
```

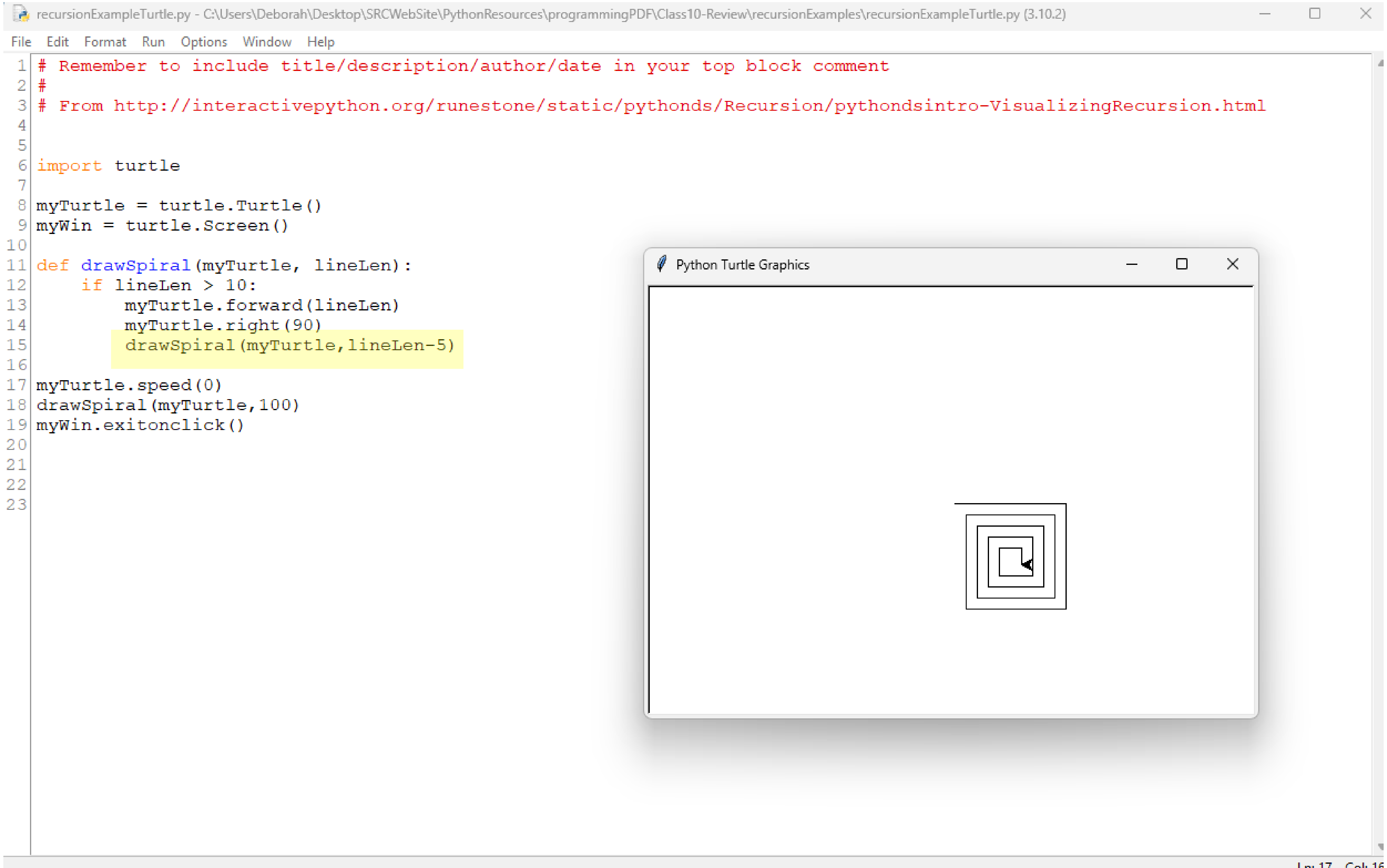

recursionExample.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\recursionEx...
File Edit Format Run Options Window Help

```python
# Remember to include title/description/author/date in your top block comment
#
# Recursion Example
#
# Author: Deborah R. Fowler
# Date: Oct 8  2018
#
# Description: an example of recursion



# A simple example of recursion - similar to the factorial
# example found at https://www.python-course.eu/recursive_functions.php

def testrecursion(x,currentLevel,maxLevel):
    print("I'm entering",x,currentLevel, maxLevel)
    if (currentLevel == maxLevel):
        print("I'm in the if",x,currentLevel, maxLevel)
        return x;
    else:
        print("I'm in the else",x,currentLevel, maxLevel)
        return testrecursion(x+1,currentLevel+1,maxLevel)

##result = testrecursion(1,0,0)
##print("Result of recursion", result)

##result = testrecursion(1,0,1)
##print("Result of recursion", result)
##
result = testrecursion(1,0,2)
print("Result of recursion", result)
```

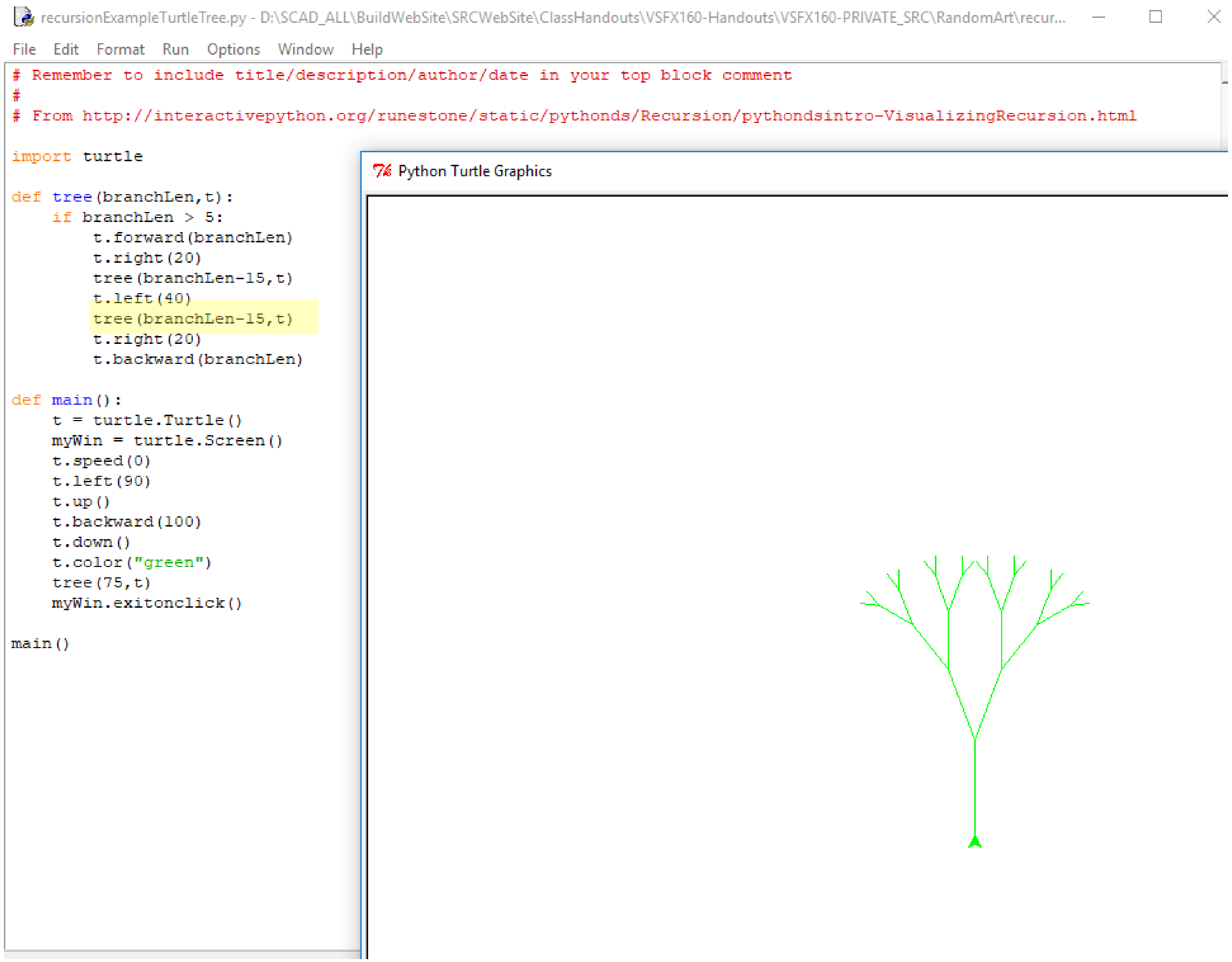
```
Python 3.10.2 (tags/v3.10.2:a58ebc
4 bit (AMD64)] on win32
Type "help", "copyright", "credits
>>>
= RESTART: C:\Users\Deborah\Deskto
gPDF\Class10-Review\recursionExamp
I'm entering 1 0 2
I'm in the else 1 0 2
I'm entering 2 1 2
I'm in the else 2 1 2
I'm entering 3 2 2
I'm in the if 3 2 2
Result of recursion 3
>>>
```

Recursion is when a function calls itself

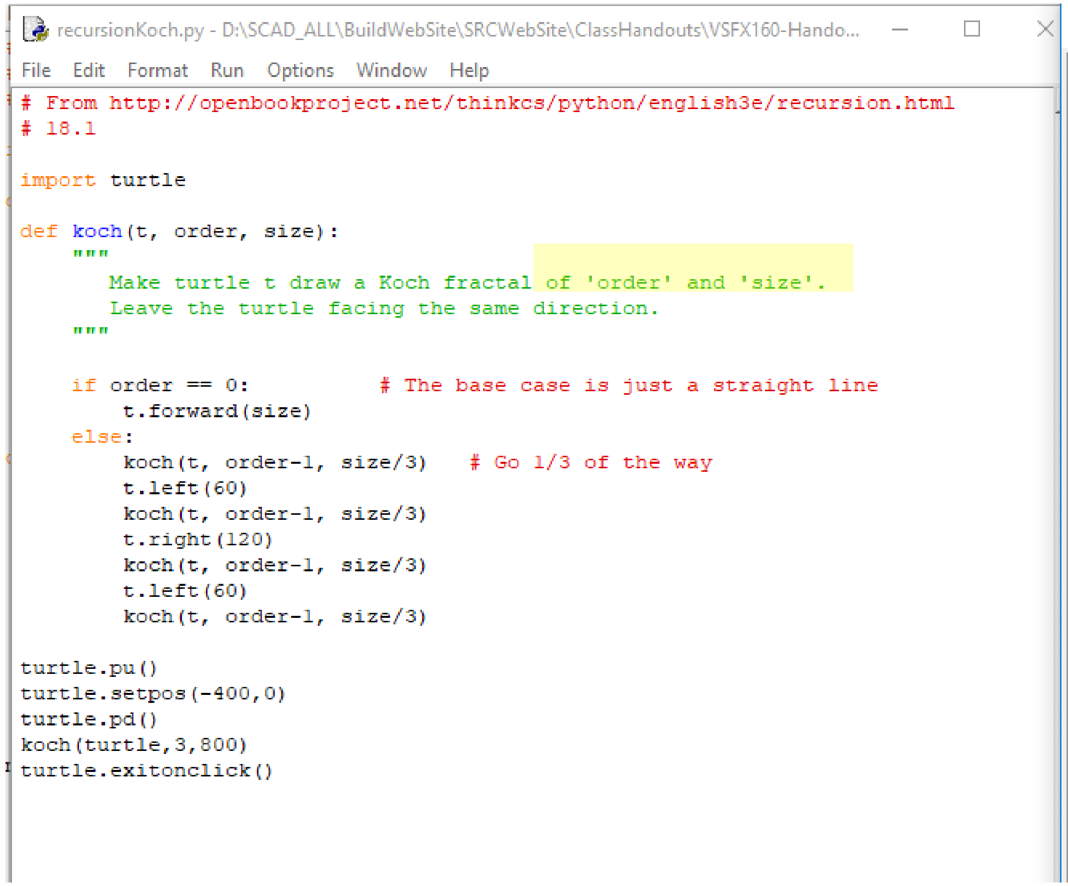
This can be very useful


recursionExampleTurtle.py - C:\Users\Deborah\Desktop\SRCWebSite\PythonResources\programmingPDF\Class10-Review\recursionExamples\recursionExampleTurtle.py (3.10.2)

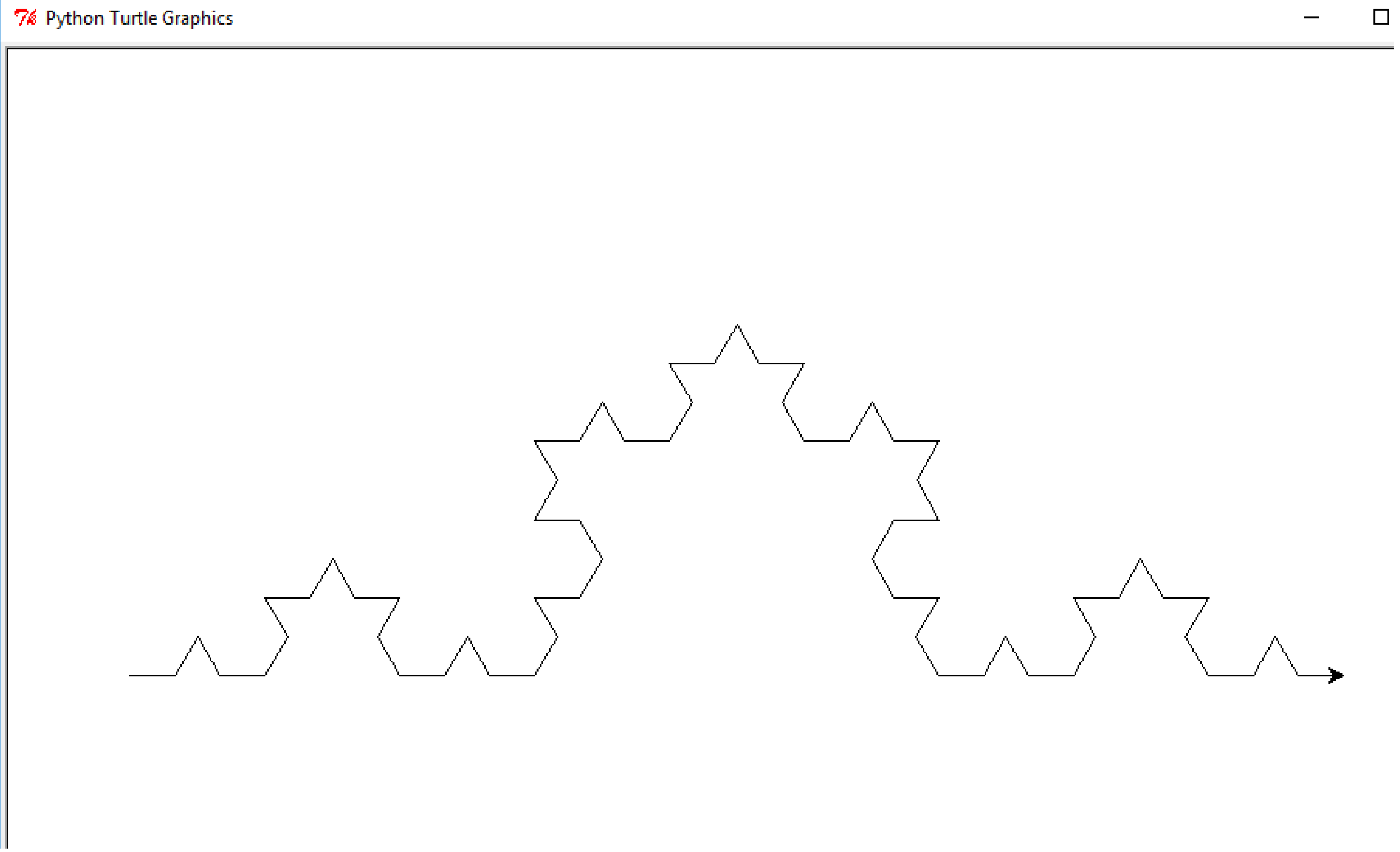
```python
# Remember to include title/description/author/date in your top block comment
#
# From http://interactivepython.org/runestone/static/pythonds/Recursion/pythondsintro-VisualizingRecursion.html


import turtle

myTurtle = turtle.Turtle()
myWin = turtle.Screen()

def drawSpiral(myTurtle, lineLen):
    if lineLen > 10:
        myTurtle.forward(lineLen)
        myTurtle.right(90)
        drawSpiral(myTurtle,lineLen-5)

myTurtle.speed(0)
drawSpiral(myTurtle,100)
myWin.exitonclick()
```


recursionExampleTurtleTree.py - D:\SCAD_ALL\BuildWebSite\SRCWebSite\ClassHandouts\VSFX160-Handouts\VSFX160-PRIVATE_SRC\RandomArt\recur...
File Edit Format Run Options Window Help
Python Turtle Graphics

```
# Remember to include title/description/author/date in your top block comment
#
# From http://interactivepython.org/runestone/static/pythonds/Recursion/pythondsintro-VisualizingRecursion.html

import turtle

def tree(branchLen,t):
    if branchLen > 5:
        t.forward(branchLen)
        t.right(20)
        tree(branchLen-15,t)
        t.left(40)
        tree(branchLen-15,t)
        t.right(20)
        t.backward(branchLen)

def main():
    t = turtle.Turtle()
    myWin = turtle.Screen()
    t.speed(0)
    t.left(90)
    t.up()
    t.backward(100)
    t.down()
    t.color("green")
    tree(75,t)
    myWin.exitonclick()

main()
```


recursionKoch.py - D:\SCAD_ALL\BuildWebSite\SRCWebSite\ClassHandouts\VSFX160-Hando...
File Edit Format Run Options Window Help

```python
# From http://openbookproject.net/thinkcs/python/english3e/recursion.html
# 18.1

import turtle

def koch(t, order, size):
    """
       Make turtle t draw a Koch fractal of 'order' and 'size'.
       Leave the turtle facing the same direction.
    """

    if order == 0:          # The base case is just a straight line
        t.forward(size)
    else:
        koch(t, order-1, size/3)   # Go 1/3 of the way
        t.left(60)
        koch(t, order-1, size/3)
        t.right(120)
        koch(t, order-1, size/3)
        t.left(60)
        koch(t, order-1, size/3)

turtle.pu()
turtle.setpos(-400,0)
turtle.pd()
koch(turtle,3,800)
turtle.exitonclick()
```



Homework:

- Dailies Exercise due Monday
- Work on the Random Art (E3) Exercise

KEY CONCEPTS

- ✓ variables
- ✓ truth statements
- ✓ looping
- ✓ functions
- ✓ I/O
- ✓ lists
- classes/objects
- OOP

KEY CONCEPTS

Continue to keep up with your reading in the online textbook

If any of these key concepts are not clear – see me!