Python:
Print, Variables, Numbers

Hina Arora

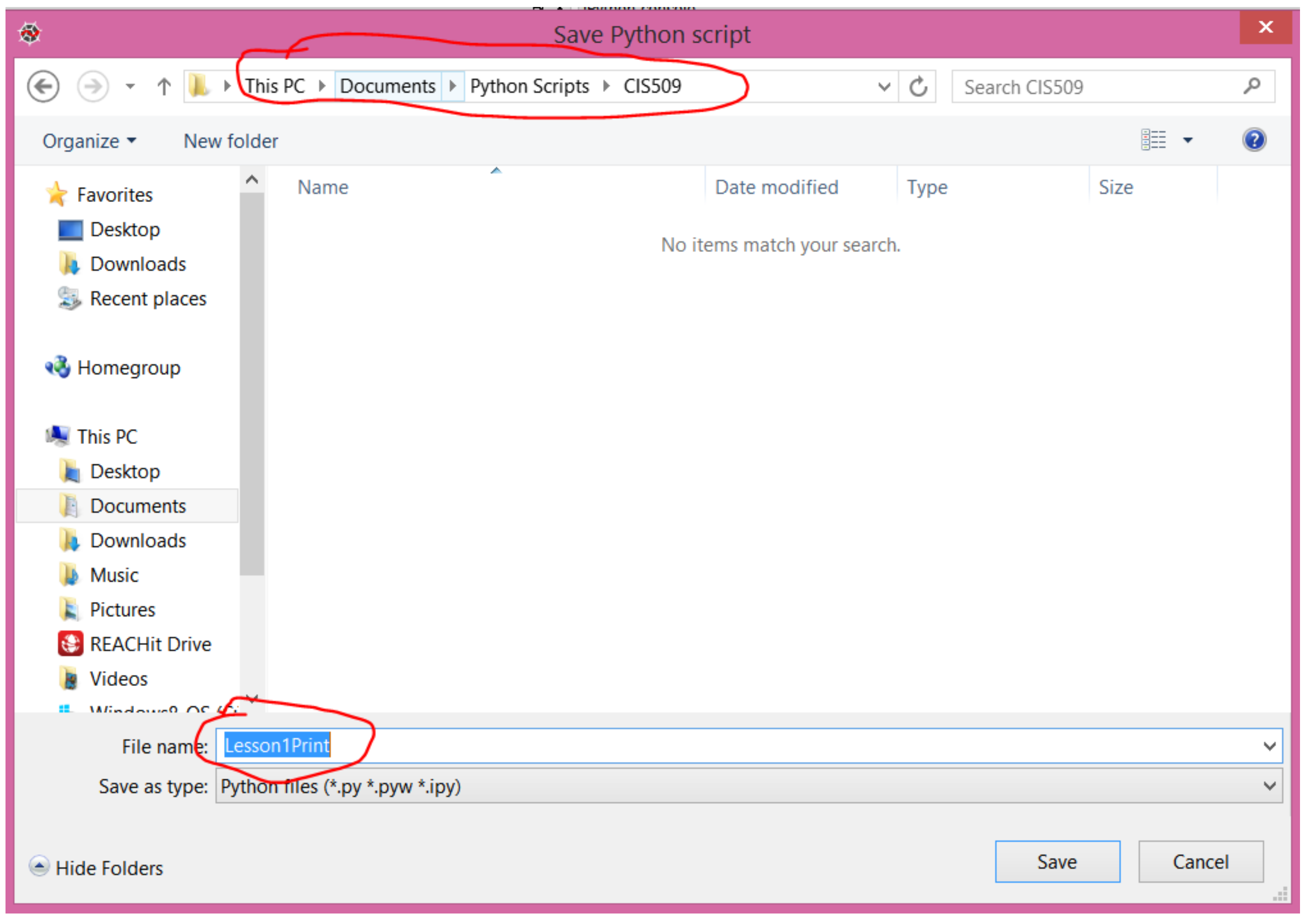
Print

follow along!

(Reference: https://docs.python.org/3/tutorial/index.html)

Create a new file called "Lesson1Print.py"

- Go to File -> New file…
- Go to File -> Save as…
- Create directory called CISPy
- Click into CISPy directory
- Save file as "Lesson1Print.py"

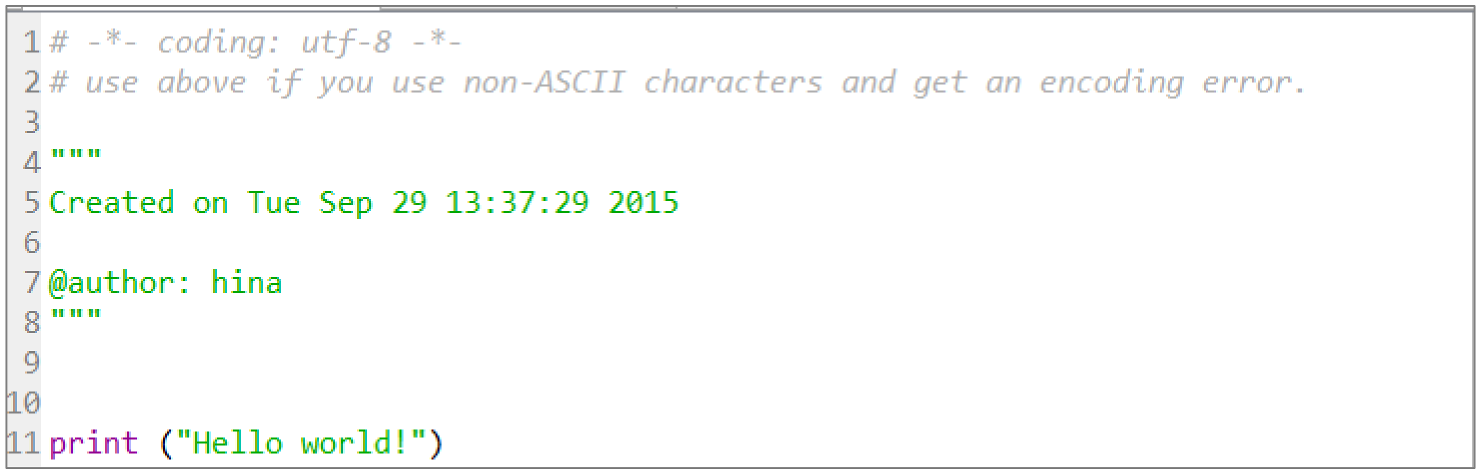


No items match your search.

File name: Lesson1Print

Save as type: Python files (*.py *.pyw *.ipy)

Put down your first line of Python code

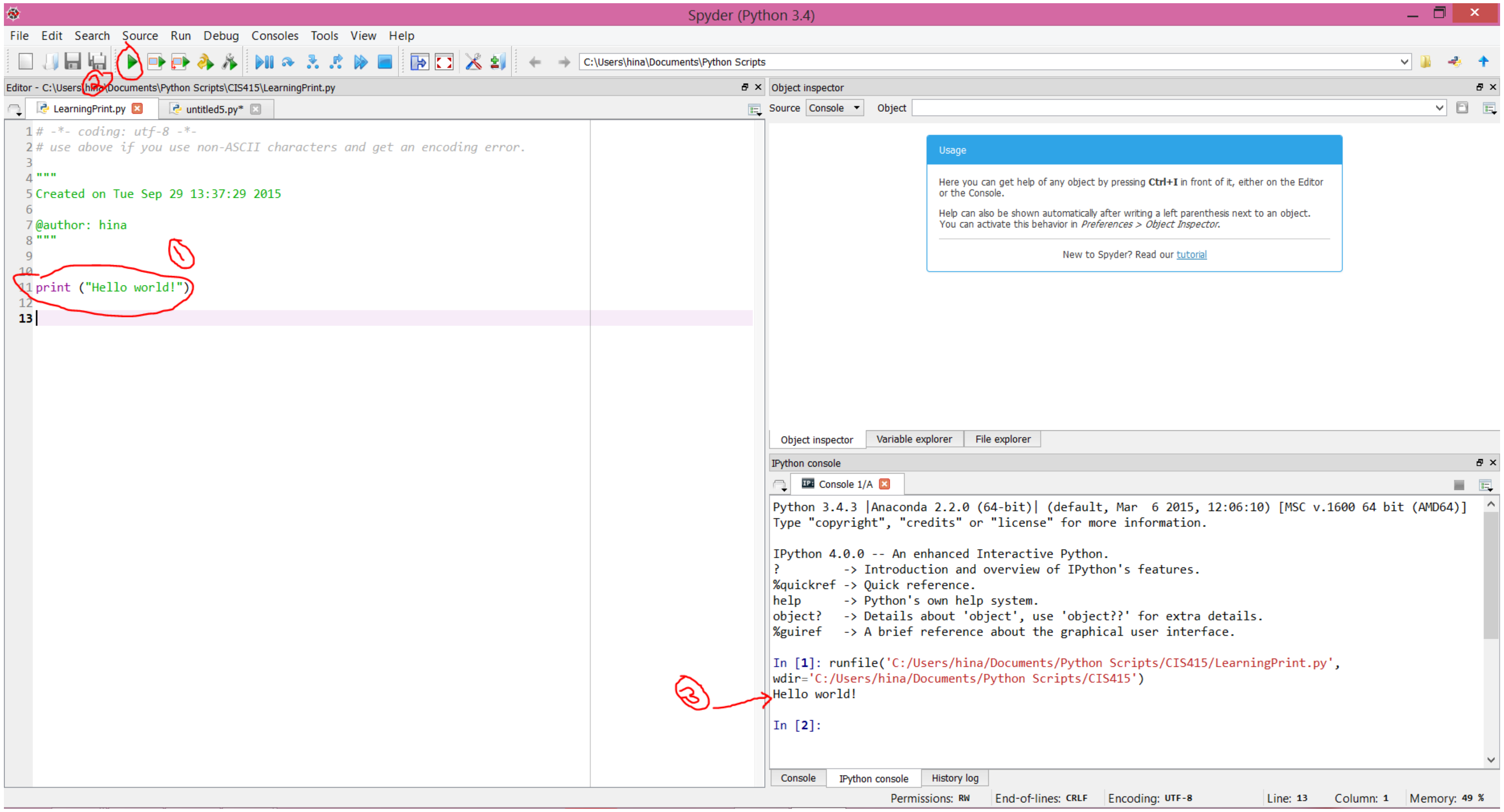


```
 1 # -*- coding: utf-8 -*-
 2 # use above if you use non-ASCII characters and get an encoding error.
 3
 4 """
 5 Created on Tue Sep 29 13:37:29 2015
 6
 7 @author: hina
 8 """
 9
10
11 print ("Hello world!")
```

Remember that the line numbers above are just for reference during the lecture… you don’t have to maintain the same line numbers for your code to work! ☺

Also, remember that indentation is used specifically for “code blocks” in Python (more about that in the next lecture)… so do not use leading spaces or tabs in your code unless it’s specifically for a code block since that will result in a syntax error.

Execute and View Output

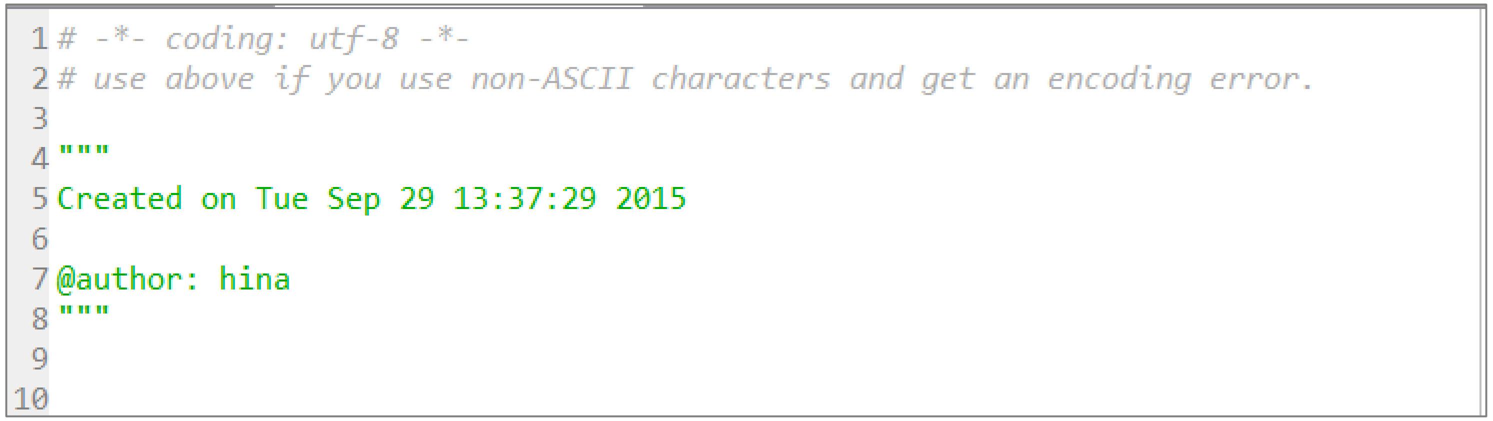

Spyder (Python 3.4)
File Edit Search Source Run Debug Consoles Tools View Help
C:\Users\hina\Documents\Python Scripts
Editor - C:\Users\hina\Documents\Python Scripts\CIS415\LearningPrint.py
LearningPrint.py
untitled5.py*
Object inspector
Source Console Object
Usage
New to Spyder? Read our tutorial
Object inspector Variable explorer File explorer
IPython console
Console 1/A
Console IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 13 Column: 1 Memory: 49 %
①
②
③

```
# -*- coding: utf-8 -*-
# use above if you use non-ASCII characters and get an encoding error.

"""
Created on Tue Sep 29 13:37:29 2015

@author: hina
"""


print ("Hello world!")

```

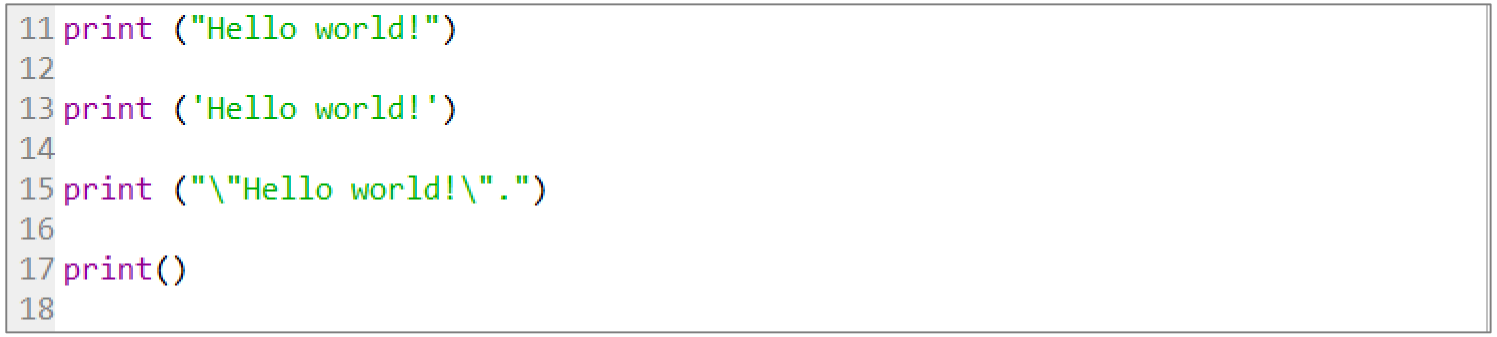
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Object Inspector.*

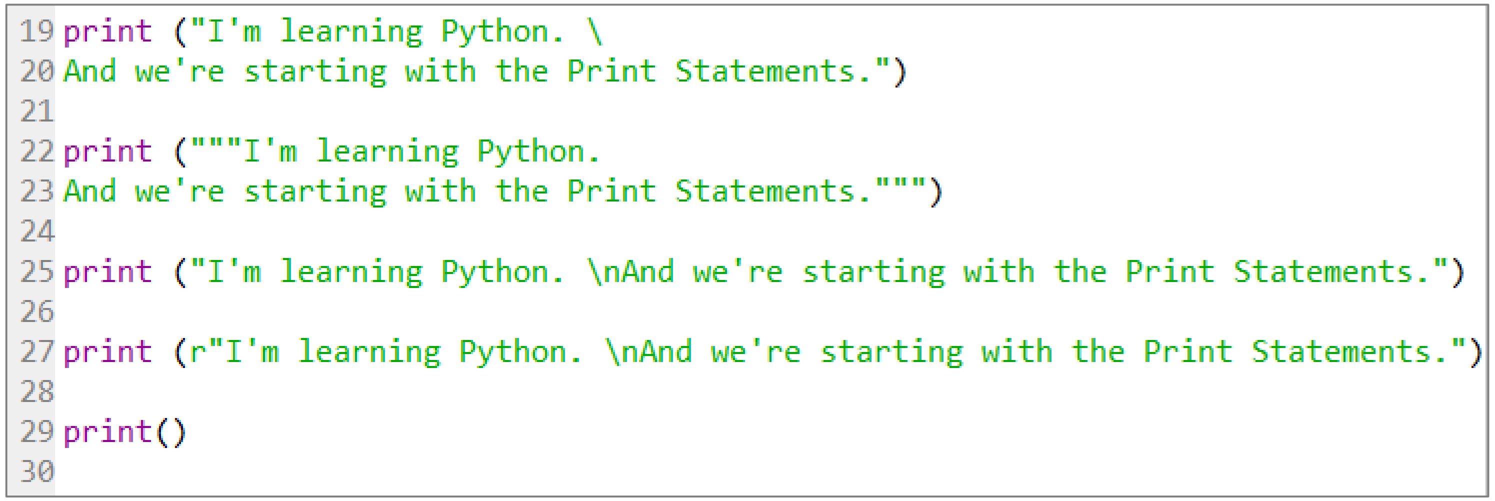
```
Python 3.4.3 |Anaconda 2.2.0 (64-bit)| (default, Mar  6 2015, 12:06:10) [MSC v.1600 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.0.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]: runfile('C:/Users/hina/Documents/Python Scripts/CIS415/LearningPrint.py',
wdir='C:/Users/hina/Documents/Python Scripts/CIS415')
Hello world!

In [2]:
```



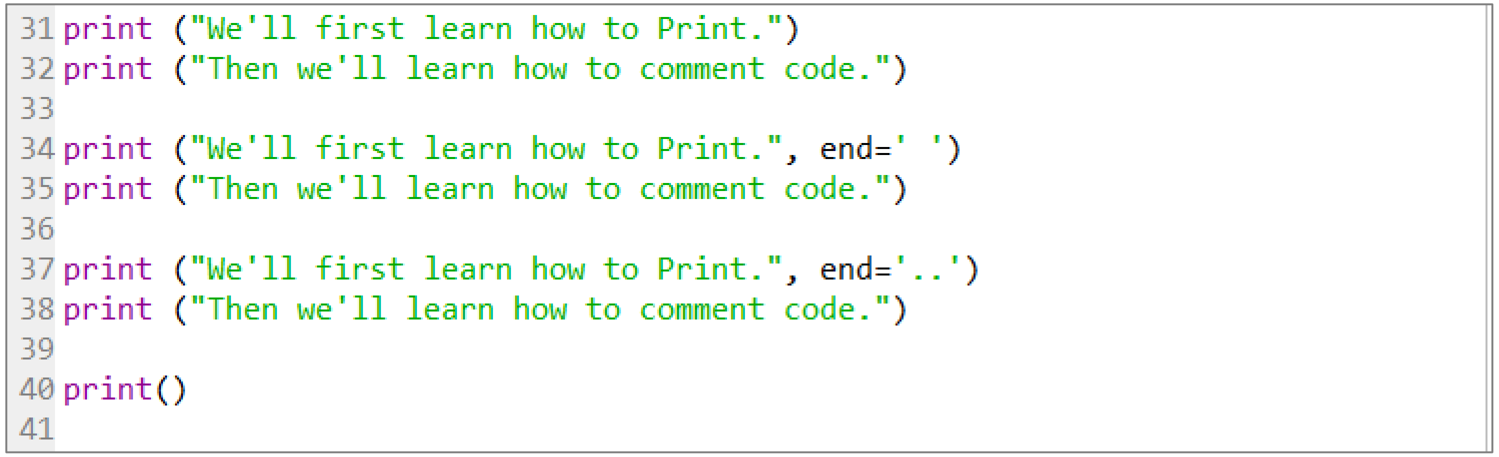
```python
# -*- coding: utf-8 -*-
# use above if you use non-ASCII characters and get an encoding error.

"""
Created on Tue Sep 29 13:37:29 2015

@author: hina
"""


```



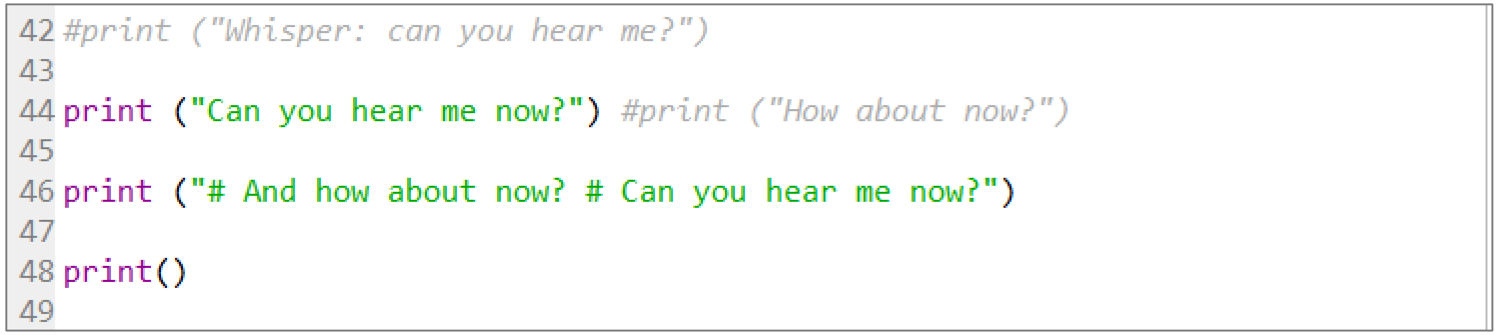
```python
print ("Hello world!")

print ('Hello world!')

print ("\"Hello world!\".")

print()

```



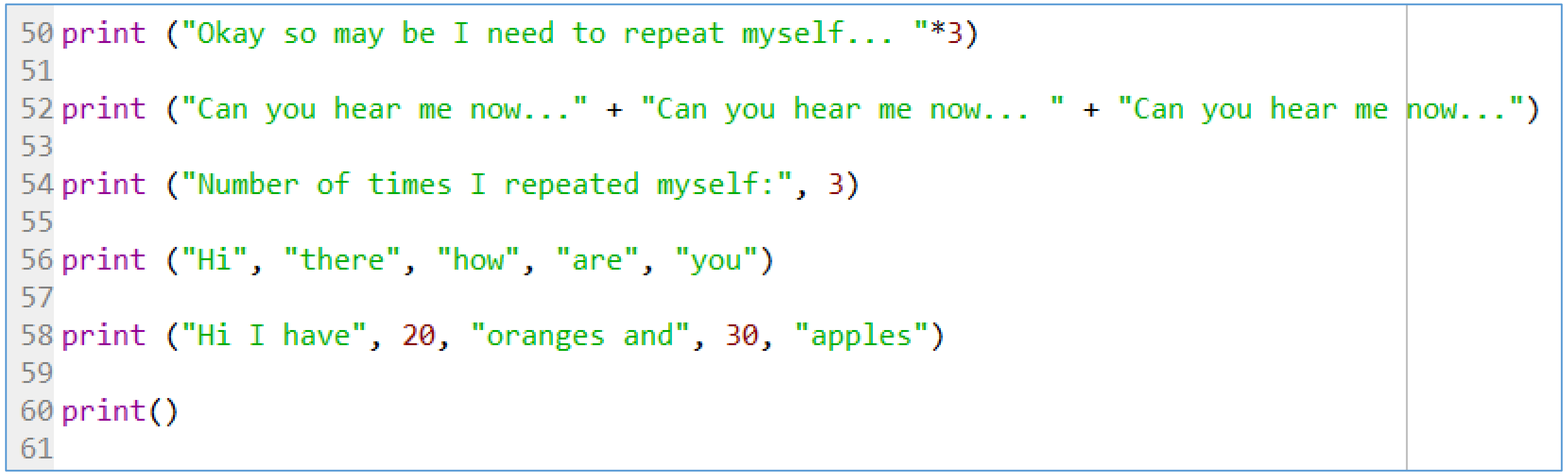
```python
print ("I'm learning Python. \
And we're starting with the Print Statements.")

print ("""I'm learning Python.
And we're starting with the Print Statements.""")

print ("I'm learning Python. \nAnd we're starting with the Print Statements.")

print (r"I'm learning Python. \nAnd we're starting with the Print Statements.")

print()

```



```python
print ("We'll first learn how to Print.")
print ("Then we'll learn how to comment code.")

print ("We'll first learn how to Print.", end=' ')
print ("Then we'll learn how to comment code.")

print ("We'll first learn how to Print.", end='..')
print ("Then we'll learn how to comment code.")

print()

```

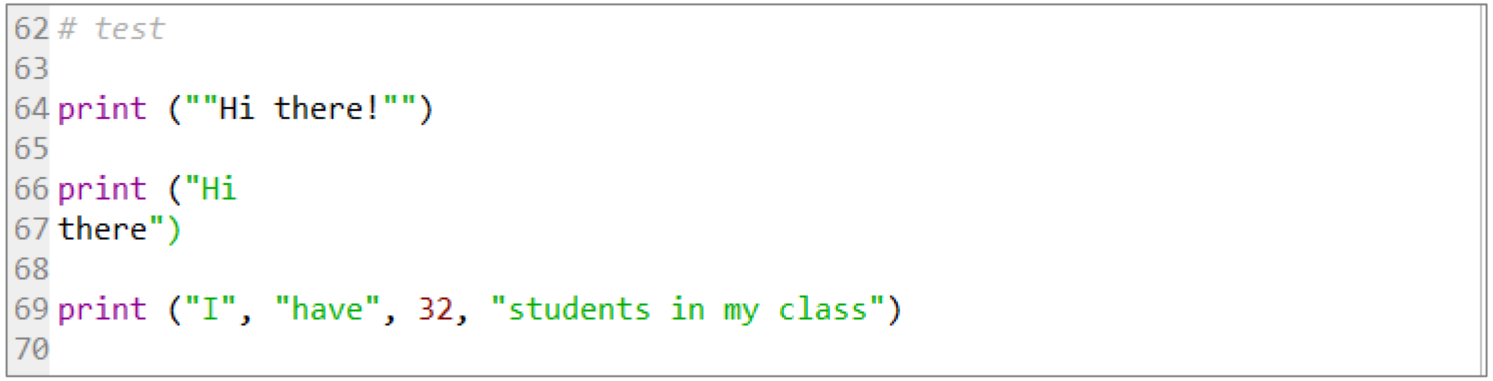


```python
#print ("Whisper: can you hear me?")

print ("Can you hear me now?") #print ("How about now?")

print ("# And how about now? # Can you hear me now?")

print()

```



```python
print ("Okay so may be I need to repeat myself... "*3)

print ("Can you hear me now..." + "Can you hear me now... " + "Can you hear me now...")

print ("Number of times I repeated myself:", 3)

print ("Hi", "there", "how", "are", "you")

print ("Hi I have", 20, "oranges and", 30, "apples")

print()

```

Will these statements work?



```
# test

print (""Hi there!"")

print ("Hi
there")

print ("I", "have", 32, "students in my class")

```

Variables

follow along!

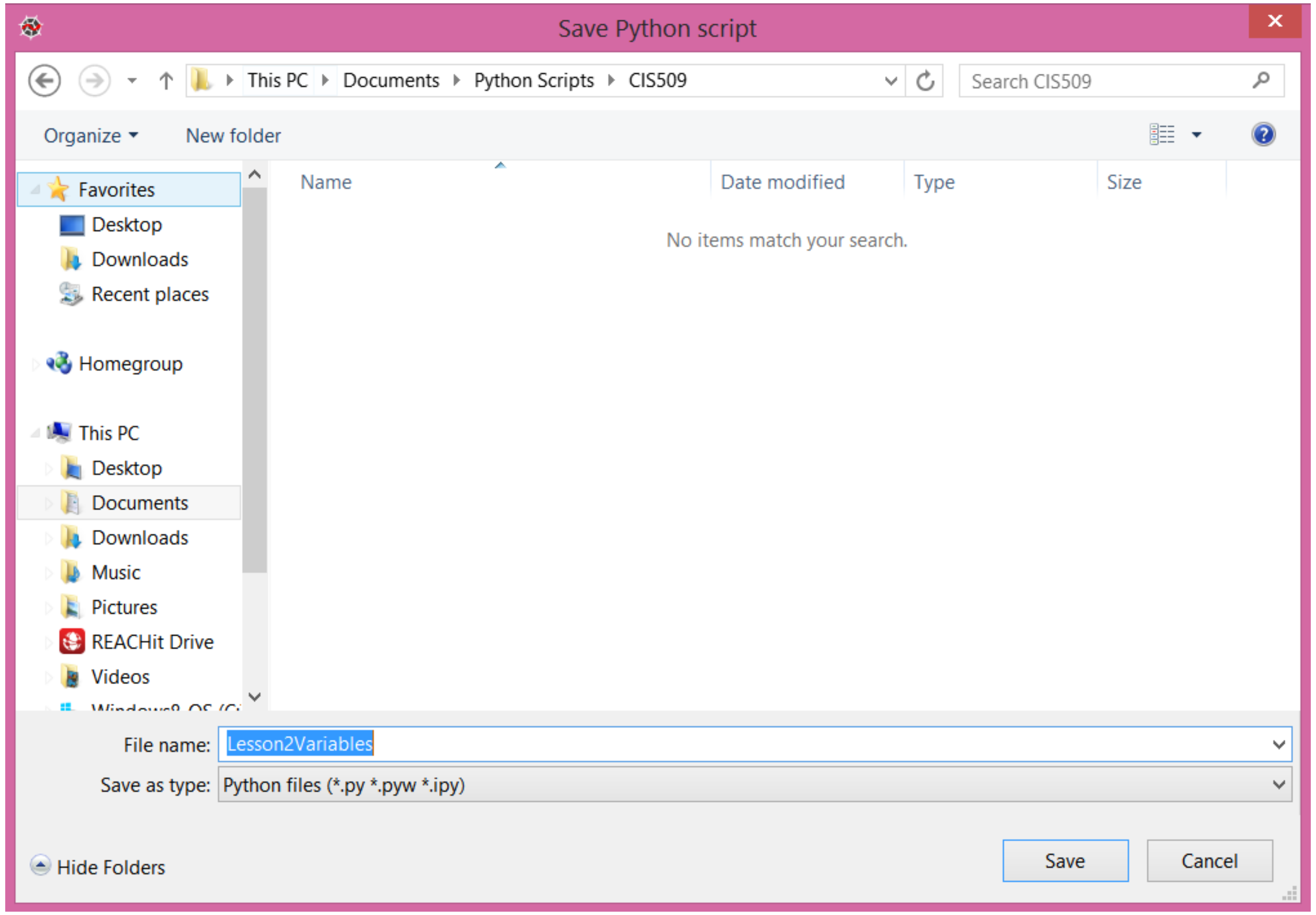
(Reference: https://docs.python.org/3/tutorial/index.html)

Variable Naming Rules

- Variable names can be any combination of letters, numbers and '_'
  - Variable names cannot start with a number
- Variable names are case sensitive
- Variable names cannot use reserved words
  - Example of reserved words: print, while, if, elif, else, and, break, class, def, …
- You should strive to have easily readable, meaningful variable names, with consistent naming style:
  - Example: NumberOfMiles, number_of_hours, speedMPH

Create a new file called "Lesson2Variables.py"

- Go to File -> New file…
- Go to File -> Save as…
- Go to CISPy directory
- Save file as "Lesson2Variables.py"


Save Python script

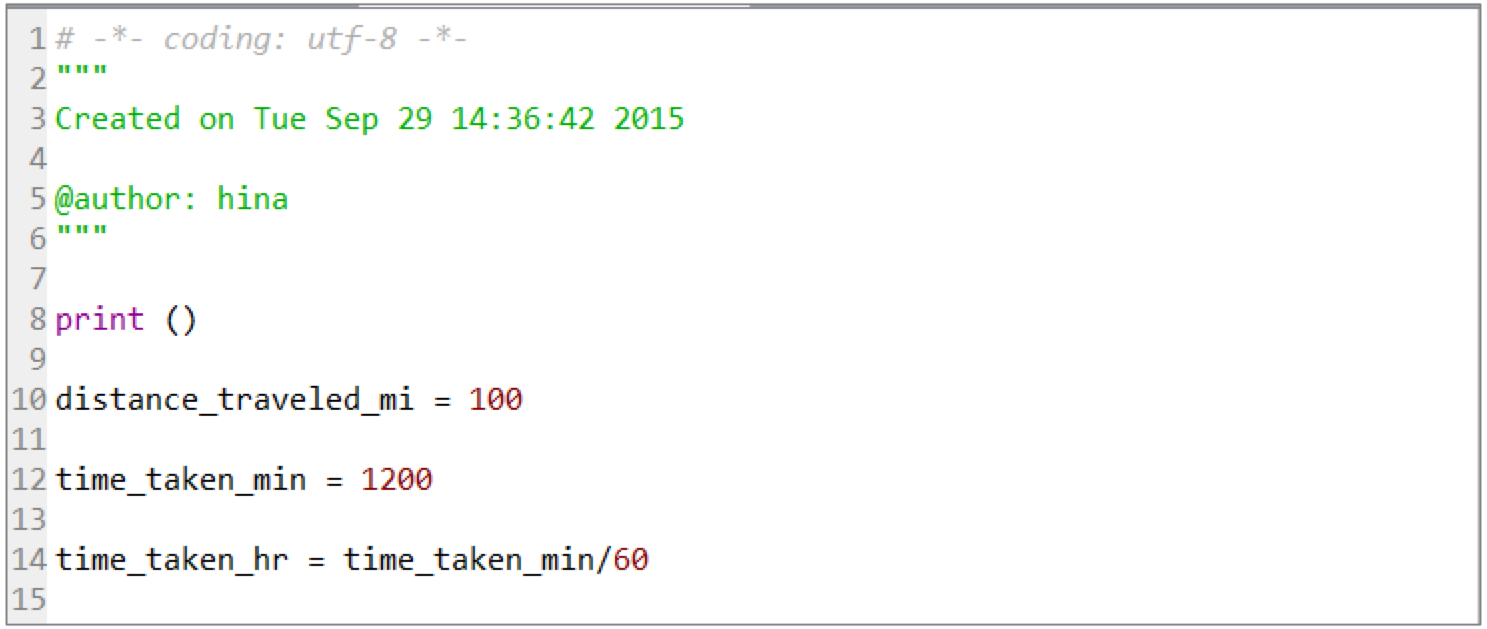
This PC ▸ Documents ▸ Python Scripts ▸ CIS509

Search CIS509

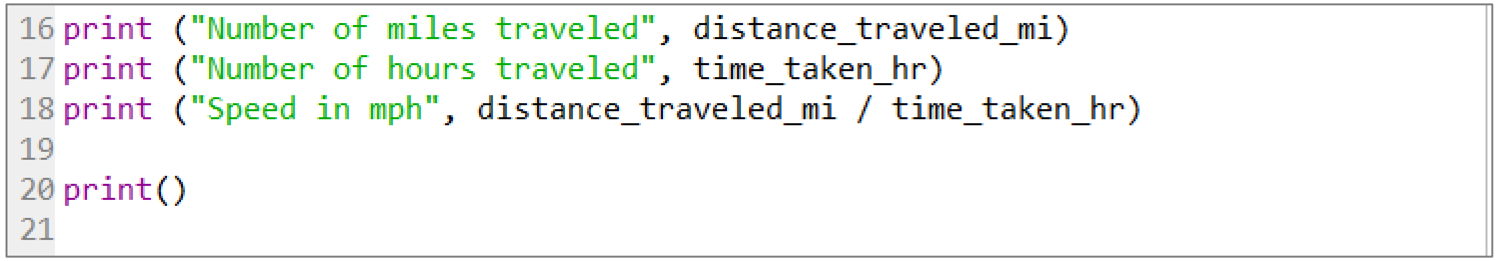
No items match your search.

File name: Lesson2Variables

Save as type: Python files (*.py *.pyw *.ipy)

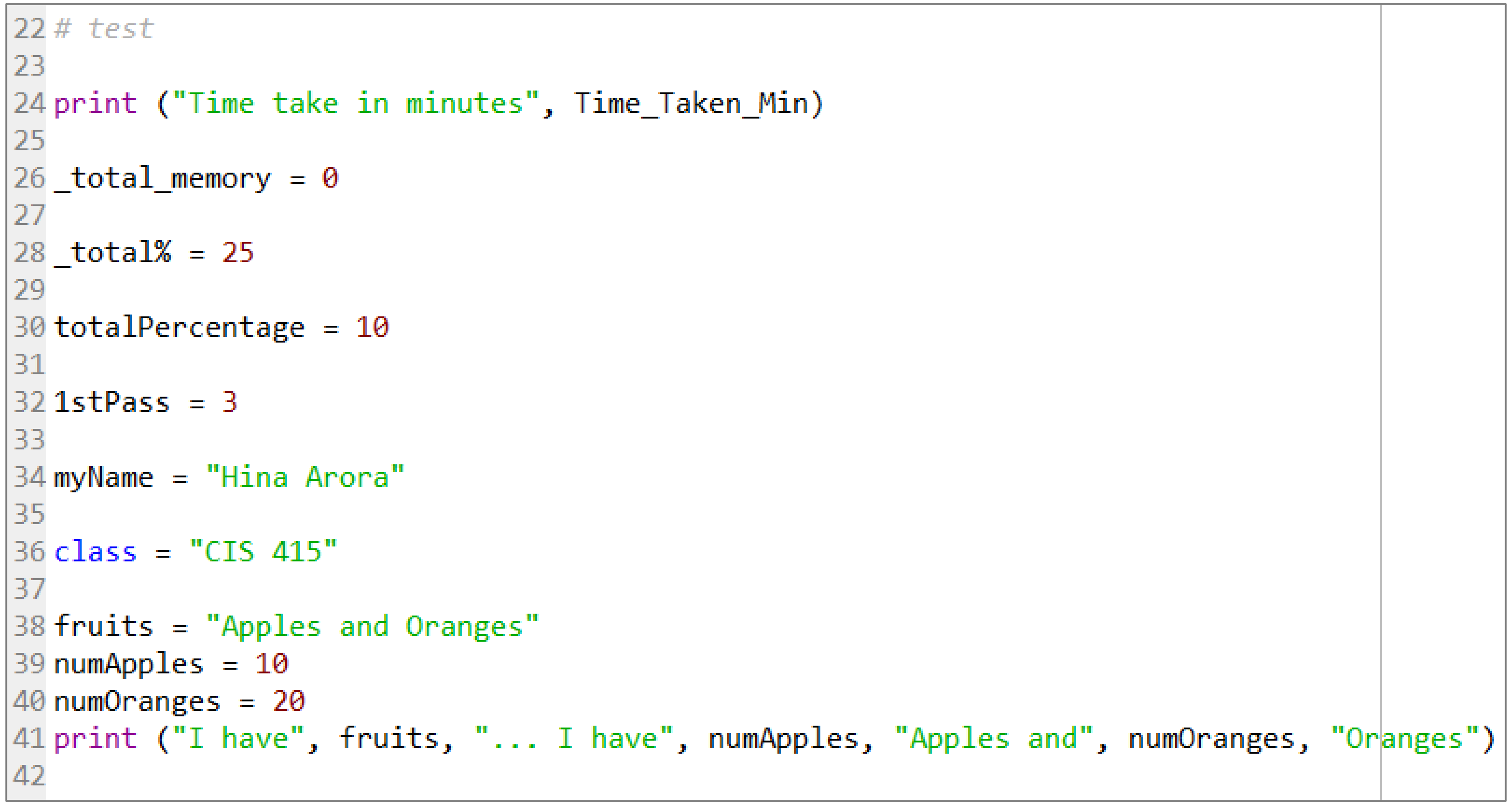


```python
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 29 14:36:42 2015

@author: hina
"""

print ()

distance_traveled_mi = 100

time_taken_min = 1200

time_taken_hr = time_taken_min/60

```



```python
print ("Number of miles traveled", distance_traveled_mi)
print ("Number of hours traveled", time_taken_hr)
print ("Speed in mph", distance_traveled_mi / time_taken_hr)

print()

```

Will these statements work?



```
# test

print ("Time take in minutes", Time_Taken_Min)

_total_memory = 0

_total% = 25

totalPercentage = 10

1stPass = 3

myName = "Hina Arora"

class = "CIS 415"

fruits = "Apples and Oranges"
numApples = 10
numOranges = 20
print ("I have", fruits, "... I have", numApples, "Apples and", numOranges, "Oranges")

```

Numbers and Operators

follow along!

(Reference: https://docs.python.org/3/tutorial/index.html)

Operator Precedence

Best to just use brackets () to disambiguate!

Operator precedence in Python, from lowest precedence (least binding) to highest precedence (most binding). Operators in the same box have the same precedence. Operators in the same box group left to right (except for comparisons, including tests, which all have the same precedence and chain from left to right and exponentiation, which groups from right to left).

or

and

not

in, not in, is, is not, <, <=, >, >=, !=, ==

|

^

&

<<, >>

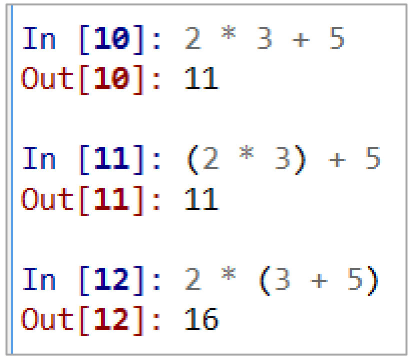
+, -

*, /, //, %

+x, -x, ~x

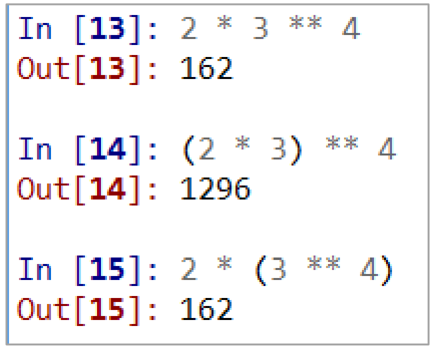
**

2 * 3 + 5



```
In [10]: 2 * 3 + 5
Out[10]: 11

In [11]: (2 * 3) + 5
Out[11]: 11

In [12]: 2 * (3 + 5)
Out[12]: 16
```

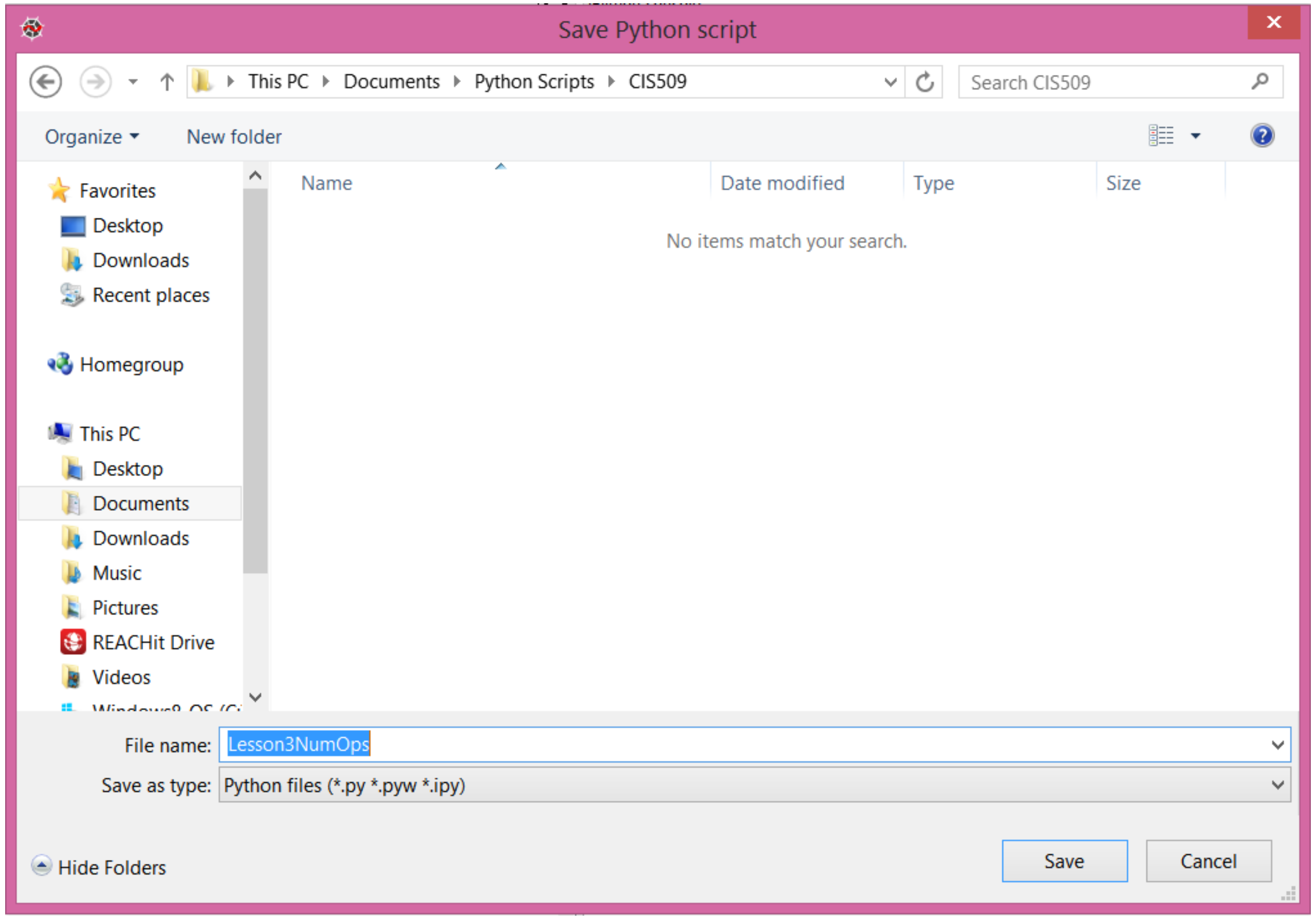
2 * 3 ** 4



```
In [13]: 2 * 3 ** 4
Out[13]: 162

In [14]: (2 * 3) ** 4
Out[14]: 1296

In [15]: 2 * (3 ** 4)
Out[15]: 162
```

Create a new file called "Lesson3NumOps.py"

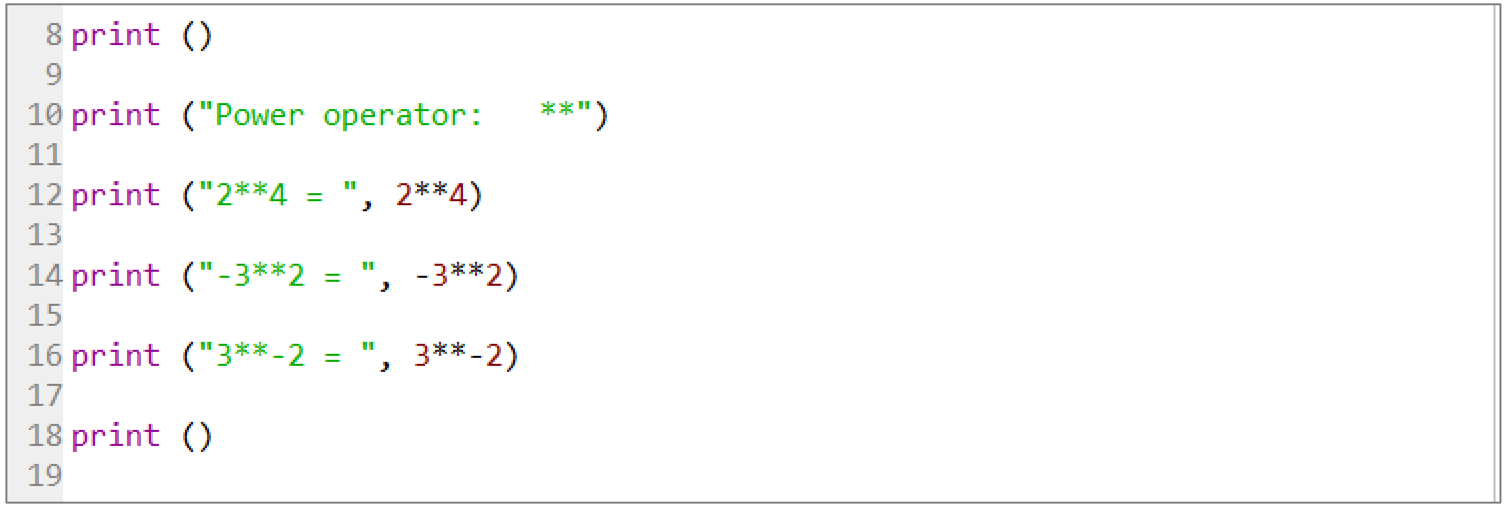
- Go to File -> New file…
- Go to File -> Save as…
- Go to CISPy directory
- Save file as "Lesson3NumOps.py"



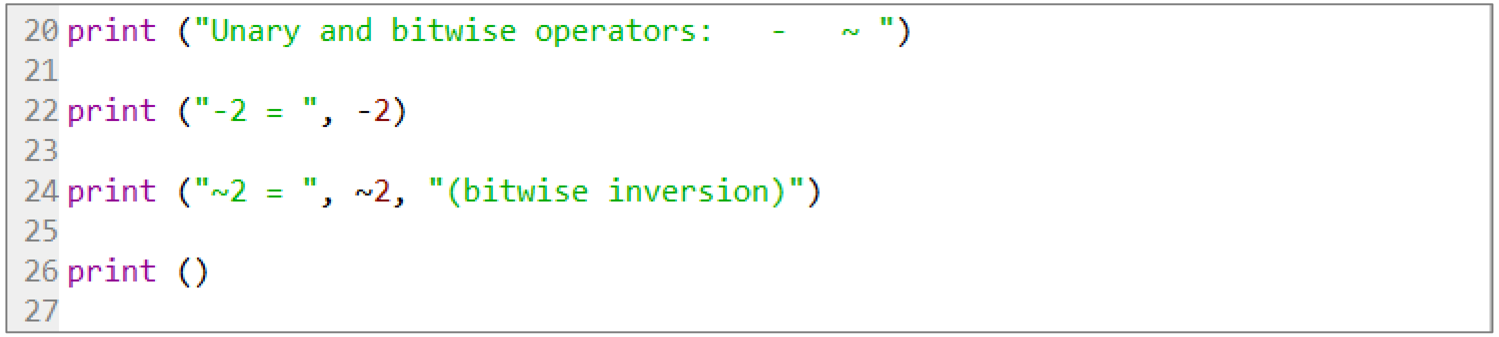
No items match your search.

File name: Lesson3NumOps

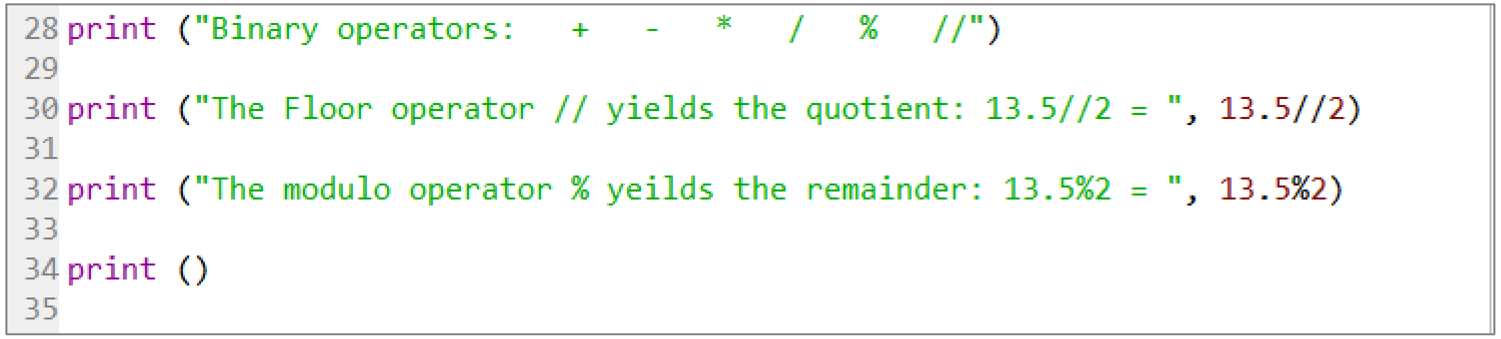
Save as type: Python files (*.py *.pyw *.ipy)



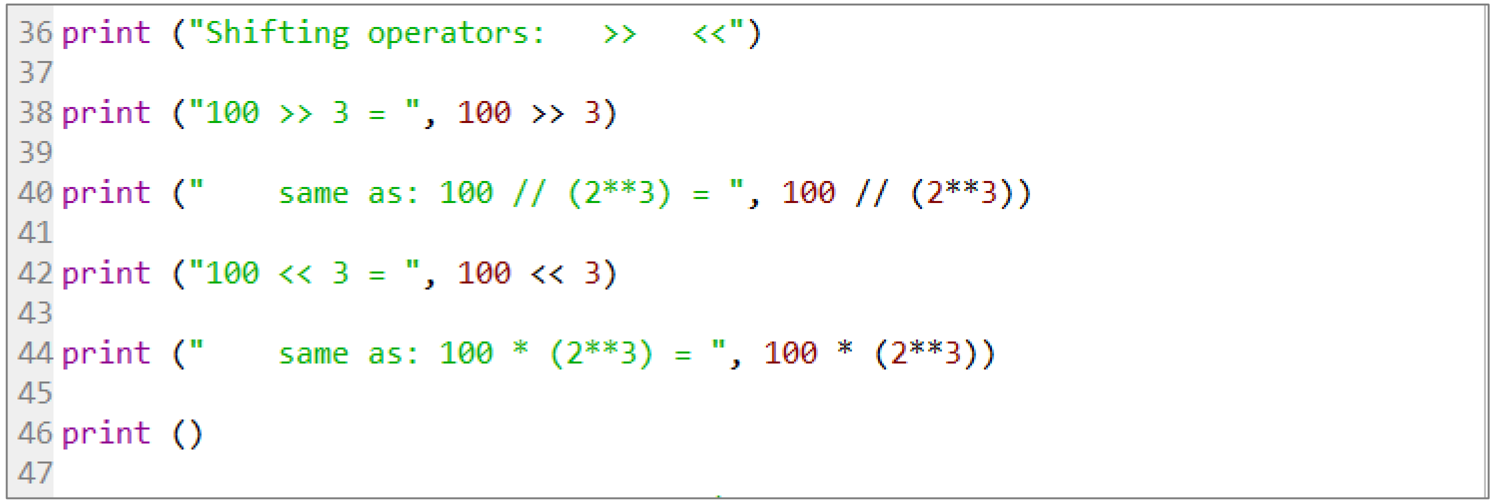
```python
print ()

print ("Power operator:   **")

print ("2**4 = ", 2**4)

print ("-3**2 = ", -3**2)

print ("3**-2 = ", 3**-2)

print ()

```



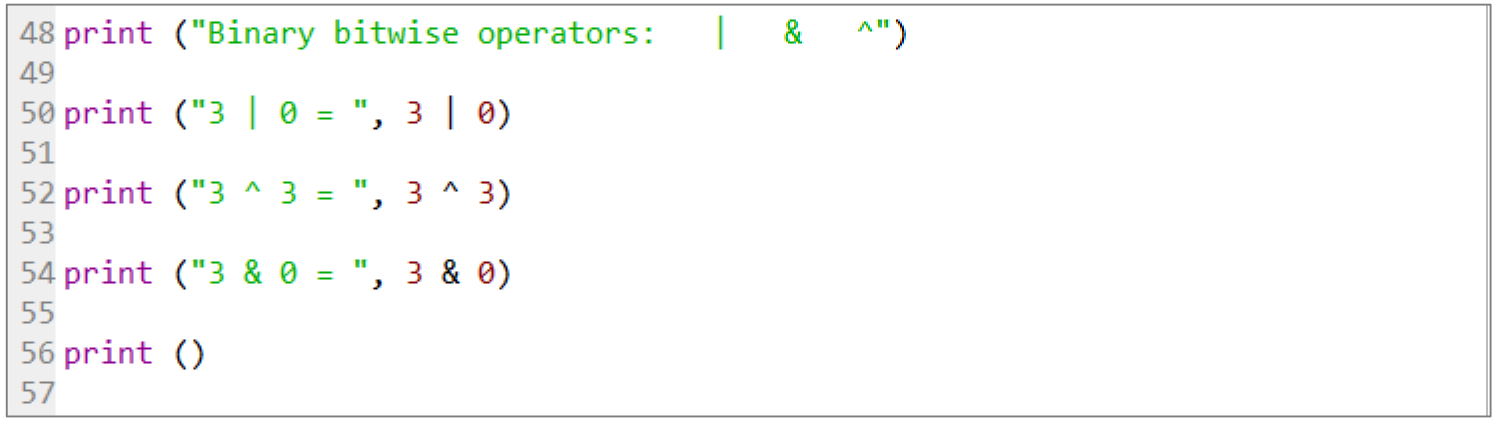
```python
print ("Unary and bitwise operators:    -    ~ ")

print ("-2 = ", -2)

print ("~2 = ", ~2, "(bitwise inversion)")

print ()

```



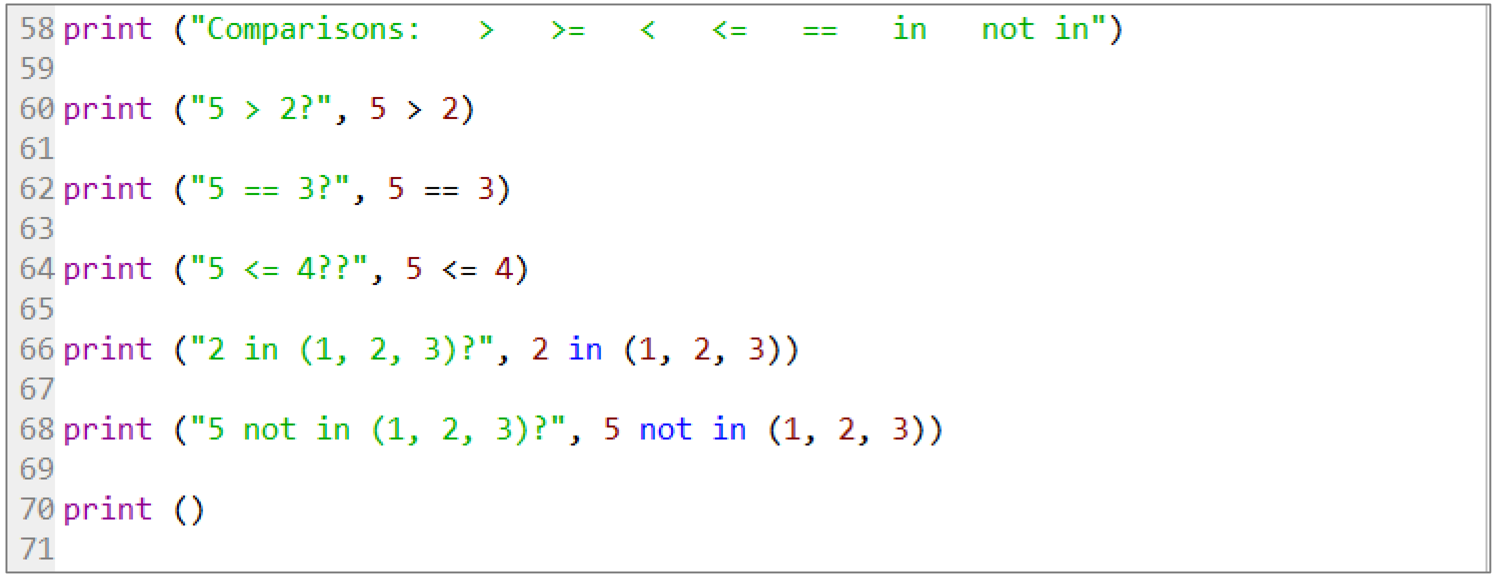
```python
print ("Binary operators:   +   -   *   /   %   //")

print ("The Floor operator // yields the quotient: 13.5//2 = ", 13.5//2)

print ("The modulo operator % yeilds the remainder: 13.5%2 = ", 13.5%2)

print ()

```



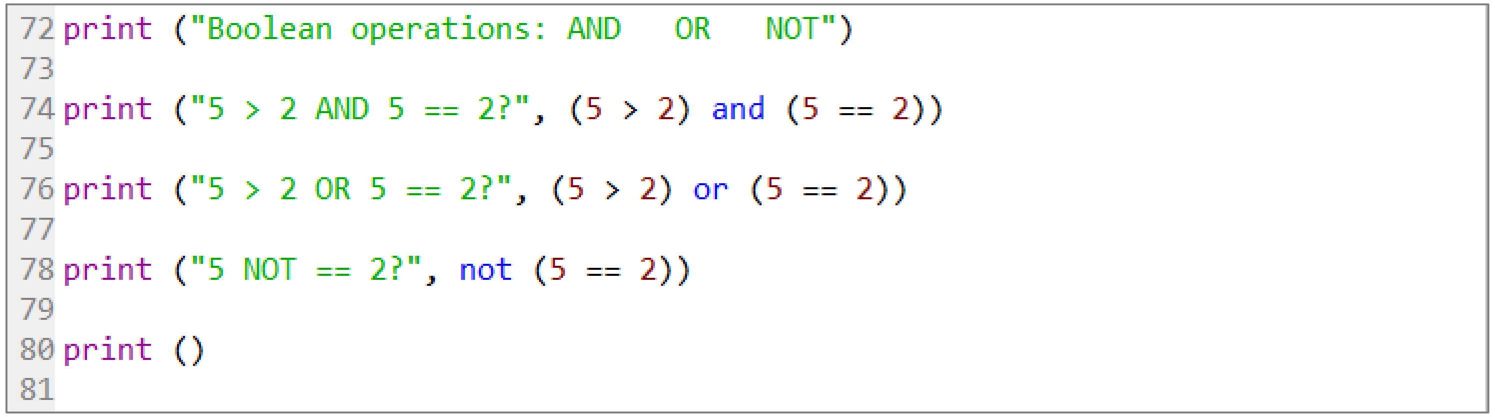
```python
print ("Shifting operators:   >>   <<")

print ("100 >> 3 = ", 100 >> 3)

print ("    same as: 100 // (2**3) = ", 100 // (2**3))

print ("100 << 3 = ", 100 << 3)

print ("    same as: 100 * (2**3) = ", 100 * (2**3))

print ()

```



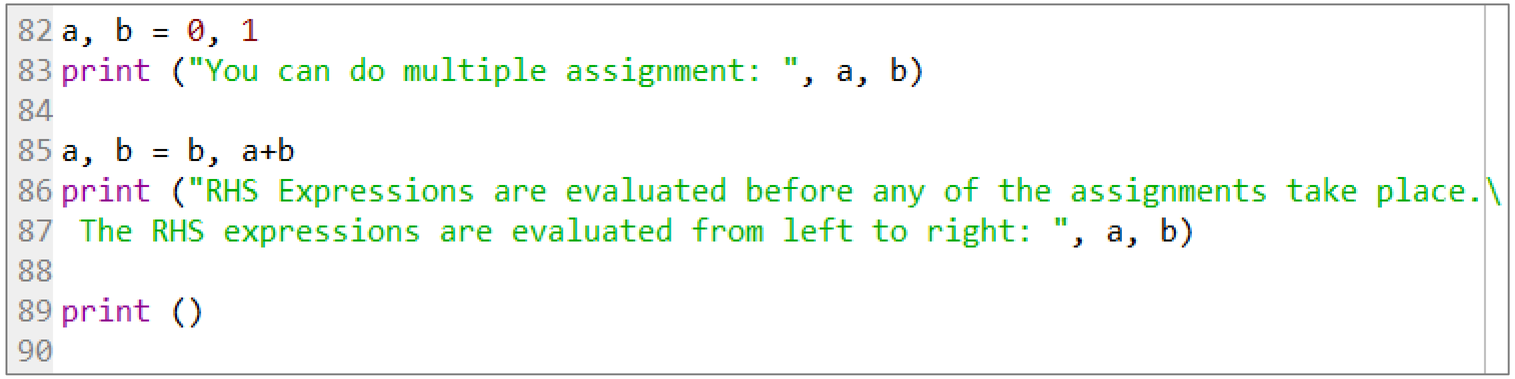
```python
print ("Binary bitwise operators:   |   &   ^")

print ("3 | 0 = ", 3 | 0)

print ("3 ^ 3 = ", 3 ^ 3)

print ("3 & 0 = ", 3 & 0)

print ()

```



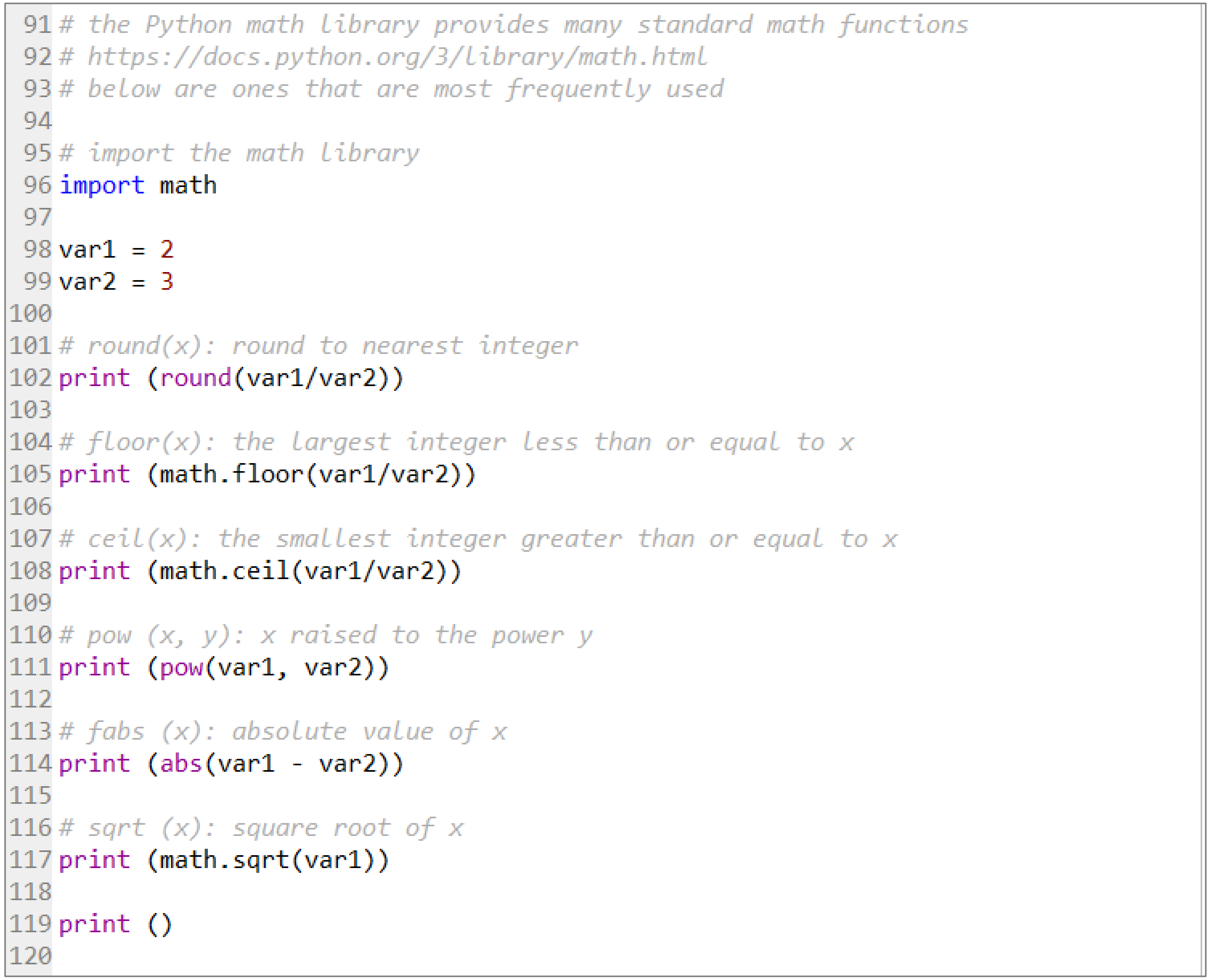
```python
print ("Comparisons:   >   >=   <   <=   ==   in   not in")

print ("5 > 2?", 5 > 2)

print ("5 == 3?", 5 == 3)

print ("5 <= 4??", 5 <= 4)

print ("2 in (1, 2, 3)?", 2 in (1, 2, 3))

print ("5 not in (1, 2, 3)?", 5 not in (1, 2, 3))

print ()

```



```python
print ("Boolean operations: AND   OR   NOT")

print ("5 > 2 AND 5 == 2?", (5 > 2) and (5 == 2))

print ("5 > 2 OR 5 == 2?", (5 > 2) or (5 == 2))

print ("5 NOT == 2?", not (5 == 2))

print ()

```

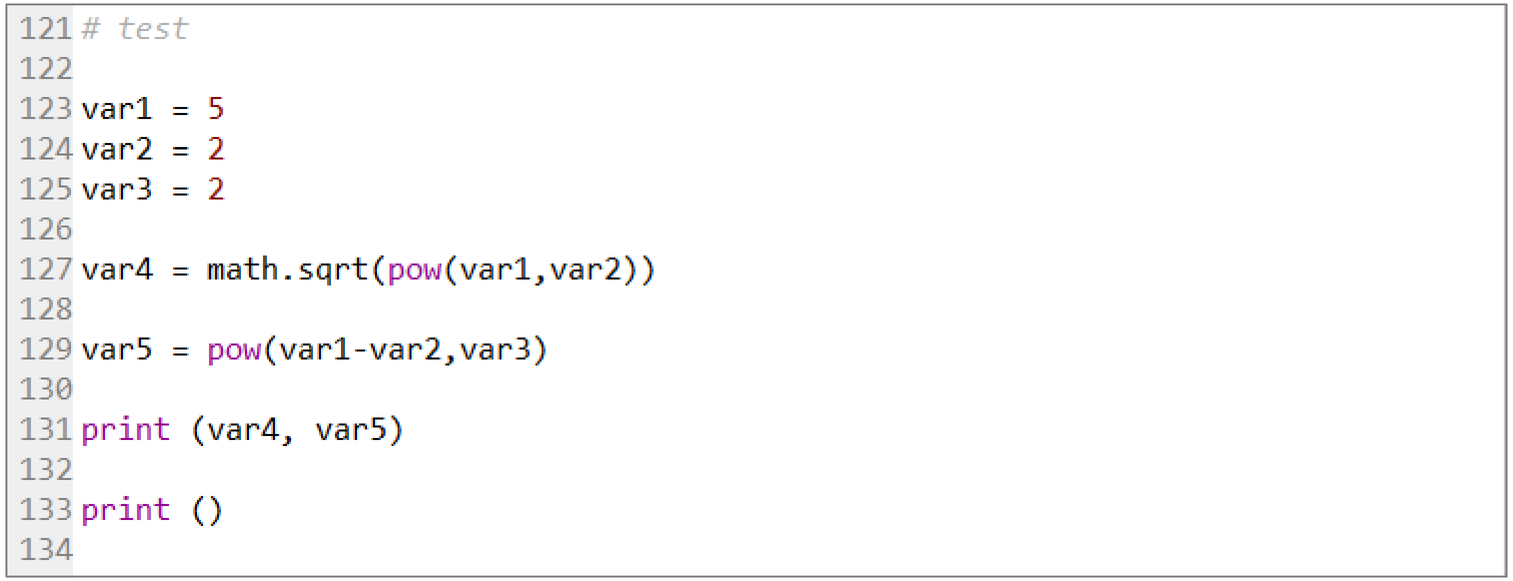


```python
a, b = 0, 1
print ("You can do multiple assignment: ", a, b)

a, b = b, a+b
print ("RHS Expressions are evaluated before any of the assignments take place.\
 The RHS expressions are evaluated from left to right: ", a, b)

print ()

```



```python
# the Python math library provides many standard math functions
# https://docs.python.org/3/library/math.html
# below are ones that are most frequently used

# import the math library
import math

var1 = 2
var2 = 3

# round(x): round to nearest integer
print (round(var1/var2))

# floor(x): the largest integer less than or equal to x
print (math.floor(var1/var2))

# ceil(x): the smallest integer greater than or equal to x
print (math.ceil(var1/var2))

# pow (x, y): x raised to the power y
print (pow(var1, var2))

# fabs (x): absolute value of x
print (abs(var1 - var2))

# sqrt (x): square root of x
print (math.sqrt(var1))

print ()

```

Will these statements work?



```python
# test

var1 = 5
var2 = 2
var3 = 2

var4 = math.sqrt(pow(var1,var2))

var5 = pow(var1-var2,var3)

print (var4, var5)

print ()

```