Introduction to Python

In this exercise, we will go through some of the basics of Python using Jupyter Notebook, the Python interpreter, and Python scripts.

Jupyter Notebook

We will use Jupyter Notebook to demonstrate some of the basics of Python programming. Jupyter Notebook provides an interactive user friendly way of writing and testing snippets of code, but for more complex programs, scripts (described below) are much more efficient.

The `print()` function

For our first code, we will have Python output a message using the **`print()`** function (to execute code in the code boxes, type shift + return):

In [1]:
```
print("Hello, world!")
```

```
Hello, world!
```

Python is very particular about syntax. Try removing the parentheses:

In [2]:
```
print "Hello, world!"
```

```
  File "<ipython-input-2-d07e2790dcc1>", line 1
    print "Hello, world!"
                        ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean prin
t("Hello, world!")?
```

Try removing the quotes:

In [3]:
```
print(Hello, world!)
```

```
  File "<ipython-input-3-d14e6f3d0ea2>", line 1
    print(Hello, world!)
                       ^
SyntaxError: invalid syntax
```

Try changing the double quotes to single quotes:

In [6]:
```
print("How's it going")
```

```
How's it going
```

If we want the message to span multiple lines, we can use triple quotes:

In [7]:
```
print('''Hello,
world!''')
```

```
Hello,
world!
```

The Python interpreter

Open a terminal and type `python --version` at the prompt (`$`). If your default version of python is 2.x, and you've installed python version 3.x, you can invoke it by typing `python3` .

At the prompt (>>>), use the print function to print the phrase "I love Python!".

Python scripts

The Jupyter notebook and the Python intepretor are great for writing and testing snippets of code, but when we want to actually create a Python program we write the instructions in a file, often called a script, using text editing software (gedit, TextWrangler, Notepad++, emacs, etc).

Building blocks of a script

Examine the `building_blocks.py` script available in the scripts link on the course website. Note some of the common components of a Python script.

Exercise 1a

Open a text editor and create a new document. Save the document as **`ex1a.py`** in a new folder called `Scripts` or some other descriptive name. The `.py` extension is the conventional way of designating a file as a python script but it is not necessary to execute the script for Unix-based operating systems.

Everything in the file will be intepreted as code, unless its commented out with a `#`.

Write a Python script that prints the message "My first Python script!". The text editing software you use should color the code to make it easier to read. Adding the `.py` extension should be sufficient to trigger syntax coloring within the file.

To execute the script from the command line, type `python` (or `python3`, if Python v2 is the default) followed by the name of the script (e.g. `python3 ex1a.py`).

Math

Python has 7 arithmetic operators: `+`, `-`, `*`, `/`, `**` (to the power of), `%` (modulus), `//` (floor division).

Python does math in a fairly intuitive way. For example, what is 7 minus 3?

```
In [8]: 7-3
```

```
Out[8]: 4
```

What is 1 divided by 2?

```
In [9]: 1/2
```

```
Out[9]: 0.5
```

What is 2 times 3?

```
In [10]: 2*3
```

```
Out[10]: 6
```

What is 2 cubed ('to the power of' uses the syntax `**`)?

```
In [11]: 2**3
```

```
Out[11]: 8
```

Try some more complex math. Does Python follow conventional rules for presedence of mathamatical operations?

In [12]:
```
3-1*5
```

Out[12]:
```
-2
```

In math, we often work with variables, such as x and y. Try assigning a value to a variable `x`, just like you would if you were doing an algebra problem.:

In [13]:
```
x = 5
```

Now use the variable in an equation:

In [17]:
```
x+3
```

Out[17]:
```
8
```

Try returning the value of the variable using the **`print()`** function:

In [18]:
```
print('x')
```

```
x
```

If you followed the syntax used for the the **`print()`** function in the earlier example, you probably got as output exactly what you entered. What happens if you remove the quotes?

In [19]:
```
print(x)
```

```
5
```

Try returning the value of a math operation using the **`print()`** function. Test what happens when you include or exclude quotes:

In [20]:
```
print(5*3)
```

```
15
```

In [21]:
```
print('5*3')
```

```
5*3
```

Exercise 1b

Write a script called `square_root.py` that does the following:

1. Stores a number to a variable.
2. Calculates the square root of the variable and stores it as a new variable.
3. Prints the result to the terminal.

The script above should have three lines of code. How can you condense it down to two lines of code while obtaining the same output? How can you condense it down to one line of code while obtaining the same output? Which is best?

Questions

Q. When are quotation marks needed and when should they be excluded in the `print()` function?

A.

Q. When are parentheses needed in the `print()` function?

A.

Q. What are the differences between single, double, and triple quotes?

A.

Q. What does a `#` signify when written directly before code on the same line? What if the `#` is on the line preceeding the code? If unsure, try modifying the `ex1a.py` by adding a `#` at the beginning of a line of code.

A.

Q. In the Python interpreter or notebook, the output of a mathematical operation is returned without using the `print()` function. What happens to the output of mathematical operations within a script?

A.

Q. Where does white space matter in a Python script? Between lines of code? Within commands? Test it in the code cell below.

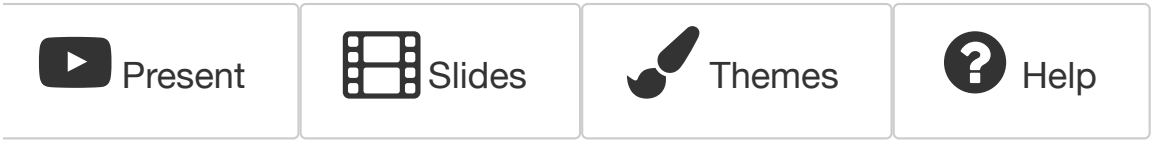
A.

In [ ]:

Additional Exercises

1. Determine the default version of Python on your computer using `python --version`. Is the default Python 3? If not, how can you invoke Python 3?
2. Launch the Python interpreter. In the python interpreter, assign the value `5` to the variable `num1` and then print the variable using the `print` function. Keep the intepreter open for the next problem.
3. In the python interpreter, assign the value `2` to the variable `num2`. Assign the product of `num1` and `num2` to the variable `num3` and print the value of `num3` to the teriminal.
4. Exit the python interpreter using either `ctrl-d` or `quit()`.
5. Relaunch the python interpreter. Are the variables you assigned above still defined?
6. Write a python script that calculates the area of a circle with radius `n`, where `n` is a variable containing any value and prints the value to the terminal. pi is roughly 3.14 but if add the following line of code to the begining of your script, you can use `pi` for pi: `from math import pi`
7. If you haven't already done so, modify the script from above with a comment embedded in the code describing what is being done.
8. Write a script that calculates the remainder of a division between two numbers, assigned to the variables `num1` and `num2`. Assign the remainder to a third variable, `rem`, and then print the remainder to the terminal.
9. Write a script that calculates the division of two numbers and returns an integer value (no digits after the decimal point; use the floor operator - `//`). What happens if you use negative numbers?
10. Using the code cell below, print some lyrics of song you know by heart split between three separate `print()` functions. Notice that the lyrics in each `print()` function are printed on separate lines. By default, the `print()` function appends a new line (`\n`). You can specify something other than `\n`. For example, if you didn't want any additional characters included in each `print()` statement, you could use `print ("string", end='')`. What happens if you edit your code with `end=''` within the `print` functions? What about `end=','` or `end='\t'`?

In [ ]:


Present
Slides
Themes
Help