Introduction to Python 3

Joe Wilkes, PhD (Weeks 1-3)
Neil Cherry (Week 4)

Week 1

The Class

- This class provides the basics of Python
- It is not meant to replace a complete college course
- It will give students taking the class enough information to:
  - Begin writing programs for their projects
  - Be able to look up and understand concepts that are new
  - Expand their programming ability
- My examples will be simple
  - I find when too many different concepts are included in one example, it is hard to understand the example
- Later in the class I give more complicated examples

The Instructors

- Joe
  - Electrical Engineer
  - Co-author of Three books on cellular phone technology
  - Been building electronic and computer projects since teenager

- Neil
  - Electrical Engineer
  - Author of Linux Home Automation Book
  - Builds/modifies/restores Retro Computers

Installing python on your computer

- See notes for preparation for class
- A list of references is also included
  - Some are books to be purchased
  - Some are free (open source) pdfs to be downloaded (links included)

Development environment

- All code examples will be available from github
  - https://github.com/IXR
  - We will use IDLE to write and modify the code for Windows and OS X
  - We will use gedit to write and modify the code for Linux
- We will use the python 3 interpreter to run the code

- There are several differences between python 2 and python 3 formatting of statements
- Python 2 is at end of life; Python 3 is the recommended version
- Code written for one version will not necessarily run on the other version

Editor

- Use one editor and stick to it
- As we will see python depends on the indenting of the code to determine how the code works
- Most people use tabs rather than spaces for indenting
  - Python recommends using spaces
- Different editors respond to tabs in different ways resulting in changes in indenting when moving from one editor to another
- While most people uses tabs, if you editor supports it check to box to insert spaces rather than tabs
- Switching between Operating Systems can also cause indentation problems

First Example Program

first_example.py - /Volumes/raid_blk/data/python_class/code_examples/first_example.py (3.6.3)

```python
"""This is a multi-line comment
You put some text here explaining the purpose the program.
"""
print ("This is an example program")
i=2
print(i)
print("i equals: ", i)
# For loop in the example (example comment)
for i in range (0, 12):
        i_squared = i*i
        print("i is ", i, "\tand i squared is ",i_squared)
# end of for loop
print("end of loop")
```

Python 3.6.3 Shell

```
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct  3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: /Volumes/raid_blk/data/python_class/code_examples/first_example.py
This is an example program
2
i equals:  2
i is  0         and i squared is  0
i is  1         and i squared is  1
i is  2         and i squared is  4
i is  3         and i squared is  9
i is  4         and i squared is  16
i is  5         and i squared is  25
i is  6         and i squared is  36
i is  7         and i squared is  49
i is  8         and i squared is  64
i is  9         and i squared is  81
i is  10        and i squared is  100
i is  11        and i squared is  121
end of loop
>>>
```

Ln: 22 Col: 4

Note:
Check to make sure
we are running Python 3
and
not Python 2

Running programs

- Python is an interpreted language
- We will build the programs in a text editor
- We will save the programs as program.py
- We will then open a command window to run the program
  - python program.py
- Some programs require elevated access to the operating system
  - In Linux and OS X we will run them by
    - sudo /usr/bin/python program.py
  - In Windows we will open the command window with Administrator Privileges
    - python program.py
- With the right header as the first line in the program we can run by typing:
  - ./program.py
  - (First we need to set program access by chmod a+rx program.py or chmod u+rx program.py)

Formatting python programs

- Many languages use special symbols to determine the end of a statement, the end of a set of statements, etc.
- For example C uses ; { } ( ) and other symbols
- Python does not use ; to indicate the end of line
  - Each statement begins on a new line
- While C uses { and } to indicate a set of code that goes together, python uses indents
- As we look at examples, we will see how the indentation is used to control program flow
- We will also discuss global and nonglobal variables

Comments

- In a line anything after a # is interpreted as a comment
- The comment ends at the end of a line
- If # is inside a set of ' ' then it is not a comment but part of the string

- Examples
  - # this is a comment
  - """ This is a
    multi-line comment
    """
  - ''' This is another form of a
    multi-line comment
    '''
  - i=3 # this is a line of code followed by a comment
  - print('This is what a tic-tac-toe symbol looks like #') # the first '#' does not start a comment

Variables

- Variables in python are defined implicitly when they are assigned
- example_int = 43 is an integer number
- example_float = 43.2000 is a floating point number
- example_string = “My name is Joe” is a string
- example_bin = 0b10011100100 is a binary number
- example_hex = 0x8e is a hexadecimal number
- example_oct = 0o73 is an octal number
- example_complex = i +3j is a complex number

Notes on variables

- In python 3:
  - A= 1
  - B = 2
  - both integers
  - A/B =1.5, a float
- type (variable) returns its type
- Printing the value of a number does not change its type

- In most other languages and Python 2
  - A= 1
  - B = 2
  - both integers
  - A/B =0, an integer

Reuse of Variables

- A variable can be reused as a different type within the same program without having to declare the type
- Type declarations are implicit
- So, the following is allowed:
  - x = 23 # an integer
  - …
  - x = ‘Bob’ # a string
  - …
  - x = 43.2134 # a float
- **Reusing variable names is allowed**
- **But will generate code that is difficult/impossible to debug**

Naming Convention

- Python uses its own unique name convention for variables
- It uses all lower case letters and underscores
- Example: this_is_my_variable = 23
- This is a recommendation but is not required by the interpreter
- If you plan to share your code with others use the ”standard name” convention
- Do not use (the Microsoft convention)
  - ThisIsMyVariable = 23,
  - though the code will work

Boolean Expressions

- and
- or
- not
- != (not equal)
- == (equal)
- >= (greater-than-equal)
- <= (less-than-equal)
- True
- False

Boolian Expression (con't)

- | bitwise OR
- ^ bitwise XOR
- & bitwise AND
- << shift left
- >> shift right
- -x, +x subtract, add 1 to x
- ~ bitwise NOT

Arithmetic Expressions

- + plus
- - minus
- * multiplication
- / divide
- ** power (ex. 2**5 is 32)
- % modulus, return remainder of division
- // Floor Division
  - round division towards zero for positive number
  - round division towards –infinity for negative number

Common constructs for loops

- If/elif/else
- for loop
- while/else
- for/else
- break
- continue
- pass

if elif else

```
statement              # not part of if elif else
if (test_statement):                  # start of if(test)
        statement      # statement is indented
        statement
elif
        statement
        statement
else:
        statement
        statement      # end of start of if(test)
 statement             #not part of if elif else
```

Example if/elif/else

if_elif_else_example.py - /Volumes/raid_blk/data/python_class/code_examples/if_elif_else_example.py (3.6.3)

```python
# example for if elif else
temperature = 0.0
scale = "C"
if (scale == "F"):
    print("Temperature is ", "%10.2f"% temperature, "degrees Farenheit")
    print("Temperature is ", "%10.2f"% ((temperature-32.0)*(5.0/9.0)), "Degrees Centigrade")
    print("Temperature is ", "%10.2f"% ((temperature-32.0)*(5.0/9.0)+273.15), "Kelvin")
elif(scale == "C"):
    print("Temperature is ", "%10.2f"% ((temperature*9.0/5.0)+32.0), "degrees Farenheit")
    print("Temperature is ", "%10.2f"% temperature, "Degrees Centigrade")
    print("Temperature is ", "%10.2f"% (temperature+273.15), "Kelvin")
elif (scale == "K"):
    print("Temperature is ", "%10.2f"% ((temperature-273.15)*(9.0/5.0)+32.0), "degrees Farenheit")
    print("Temperature is ", "%10.2f"% (temperature-273.15), "Degrees Centigrade")
    print("Temperature is ", "%10.2f"% temperature, "Kelvin")
else:
    print("Wrong scale specified")
print("Conversion Complete")
```

Ln: 3 Col: 10

Results for Example if/elif/else

```
Python 3.6.3 Shell
 RESTART: /Volumes/raid_blk/data/python_class/code_examp
les/if_elif_else_example.py
Temperature is       32.00 degrees Farenheit
Temperature is        0.00 Degrees Centigrade
Temperature is      273.15 Kelvin
Conversion Complete
>>>
 RESTART: /Volumes/raid_blk/data/python_class/code_examp
les/if_elif_else_example.py
Temperature is     -951.34 degrees Farenheit
Temperature is     -546.30 Degrees Centigrade
Temperature is     -273.15 Kelvin
Conversion Complete
>>>
 RESTART: /Volumes/raid_blk/data/python_class/code_examp
les/if_elif_else_example.py
Temperature is     -459.67 degrees Farenheit
Temperature is     -273.15 Degrees Centigrade
Temperature is        0.00 Kelvin
Conversion Complete
>>>
                                   Ln: 430  Col: 4
```

This answer is not correct

for Loop

```
statement                    # not part of for loop
for i in range (A,B,C):      # starts at A and ends at B-1, with increment C
    statement                # this and all statements in the for loop
    statement                # are indented
statement                    # not part of for loop
```

- range increments by 1 if not specified
- example of range() step option:
  - range(1, 11) # return 1 through 10 in steps of 1
  - range(1, 11, 2) # returns 1 thought 10 in steps of 2 (only odd digits)
  - range(2, 11, 2) # returns 2 thought 10 in steps of 2 (only even digits)

Example for loop

for_loop_example.py - /Volumes/raid_blk/data/python_class/code_examples/for_lo...

```
for i in range(12,20):
    sq_root_i = i**(1.0/2.0)
    print ("for i of",i,"square root of i is", "%5.3f"% sq_root_i)
```

Ln: 4 Col: 4

Python 3.6.3 Shell

```
 RESTART: /Volumes/raid_blk/data/python_class/code_examples/for_loop_example.py
for i of 12 square root of i is 3.464
for i of 13 square root of i is 3.606
for i of 14 square root of i is 3.742
for i of 15 square root of i is 3.873
for i of 16 square root of i is 4.000
for i of 17 square root of i is 4.123
for i of 18 square root of i is 4.243
for i of 19 square root of i is 4.359
>>>
```

Ln: 338 Col: 4

While Loop

```
statement                # not part of while loop
while(test):             # start of while(test)
        statement        # part of while loop
        statement        # end of while (test)
statement                # not part of while loop
```

- A while loop executes while the test is true
- while(1) will execute forever or until something stops it

Example of While Loop

while_loop_example.py - /Volumes/raid_blk/data/python_class/code_examples/while_loop_example.py (3.6.3)

```
i=10
while i<=120:
    sq_root_i = i**(1.0/2.0)
    print ("for i of","%4.0d"% i,"square root of i is", "%5.3f"% sq_root_i)
    i=i+10
```

Ln: 4 Col: 31

Python 3.6.3 Shell

```
>>>
 RESTART: /Volumes/raid_blk/data/python_class/code_examples/while_loop_example.py
for i of   10 square root of i is 3.162
for i of   20 square root of i is 4.472
for i of   30 square root of i is 5.477
for i of   40 square root of i is 6.325
for i of   50 square root of i is 7.071
for i of   60 square root of i is 7.746
for i of   70 square root of i is 8.367
for i of   80 square root of i is 8.944
for i of   90 square root of i is 9.487
for i of  100 square root of i is 10.000
for i of  110 square root of i is 10.488
for i of  120 square root of i is 10.954
>>>
```

Ln: 366 Col: 4

for/else

```
• statement                    #not part of for/else
  for i = range(A,B):          # start of for i=range(A,B)
        statement
        statement
  else:
        statement
        statement              # end of for i=range(A,B)
  statement                    #not part of for/else
```

Example of for/else

for_else_example.py - /Volumes/raid_blk/data/python_class/code_examples/for_else_example.py (3.6.3)

```
# find all prime numbers between 1 and 100
# if the number is not prime, list its factors
for number in range(1, 101):
    for factor in range(2, number):
        if number % factor == 0:
            print("This number is not prime", "%4d"% number, '=', factor, '*', "%2d"% (number/factor) )
            break
    else:
        # loop fell through without finding a factor
        print('This is a prime number  ', "%4d"% number)
```

Ln: 6 Col: 82

Example output for/else

```
Python 3.6.3 Shell
 RESTART: /Volumes/raid_blk/data/python_
class/code_examples/for_else_example.py
This is a prime number      1
This is a prime number      2
This is a prime number      3
This number is not prime    4 = 2 *  2
This is a prime number      5
This number is not prime    6 = 2 *  3
This is a prime number      7
This number is not prime    8 = 2 *  4
This number is not prime    9 = 3 *  3
This number is not prime   10 = 2 *  5
This is a prime number     11
This number is not prime   12 = 2 *  6
This is a prime number     13
This number is not prime   14 = 2 *  7
This number is not prime   15 = 3 *  5
This number is not prime   16 = 2 *  8
This is a prime number     17
This number is not prime   18 = 2 *  9
This is a prime number     19
This number is not prime   20 = 2 * 10
This number is not prime   21 = 3 *  7
This number is not prime   22 = 2 * 11
This is a prime number     23
This number is not prime   24 = 2 * 12
This number is not prime   25 = 5 *  5
                                Ln: 386  Col: 4
```

...

```
Python 3.6.3 Shell
This number is not prime   75 = 3 * 25
This number is not prime   76 = 2 * 38
This number is not prime   77 = 7 * 11
This number is not prime   78 = 2 * 39
This is a prime number     79
This number is not prime   80 = 2 * 40
This number is not prime   81 = 3 * 27
This number is not prime   82 = 2 * 41
This is a prime number     83
This number is not prime   84 = 2 * 42
This number is not prime   85 = 5 * 17
This number is not prime   86 = 2 * 43
This number is not prime   87 = 3 * 29
This number is not prime   88 = 2 * 44
This is a prime number     89
This number is not prime   90 = 2 * 45
This number is not prime   91 = 7 * 13
This number is not prime   92 = 2 * 46
This number is not prime   93 = 3 * 31
This number is not prime   94 = 2 * 47
This number is not prime   95 = 5 * 19
This number is not prime   96 = 2 * 48
This is a prime number     97
This number is not prime   98 = 2 * 49
This number is not prime   99 = 3 * 33
This number is not prime  100 = 2 * 50
>>>
                                Ln: 386  Col: 4
```

Quick and dirty print

- for now just use
- print (variable) to get printed results of variable
- We will discuss formatted printing next week

Questions?

Home Work

- If you haven't installed python 3 and your favorite text editor on your laptop, please do so
- Add a "for loop" around the if_elif_else_example to find the results for a range of Fahrenheit temperatures from -200 to +400 with an increment of 50
- Add a fix for temperatures below absolute zero to print out an error message when the temperature is below absolute zero. (Google the value of absolute zero if you do not know it).