Python Workshop - ANU (CBE)

Jupyter Notebooks and Python Programming Fundamentals

Matthew McKay [matthew.mckay@anu.edu.au]

June 2017

Agenda ...

1. Jupyter Notebooks
2. Intro to Python Programming
3. Resources

Jupyter

Quick Review

1. Notebook Basics
2. Modal Editing
3. Running Code
4. Text Editor Features (Syntax Highlighting etc.)
5. Tab Completion
6. Object Introspection
7. Working with the shell
8. Working with Files
9. First Python Program

Everyone is able to run Jupyter?

Any Questions

Discuss Order of Cell Execution (DEMO)

Intro to Python Topics

1. Introductory Example
2. Basic Structure of a Python Program
3. Variables and Assignment
4. Data Types
    - Booleans
    - Numbers (integers, floats, fractions and complex numbers)
    - Strings
    - Bytes (and byte arrays)
    - Lists
    - Dictionaries
    - Sets
    - Tuples
5. Mutable and Immutable

Python Fundamentals

See notebook **intro-to-python.ipynb**

Order of Operations

Python uses math conventions to determine the order of operations

1. Parentheses
2. Exponentiation
3. Multiplication and Division
4. Addition and Subtraction

Note: Operators that share precedence are then evaluated from left to right.

Using parentheses is good programming practice to improve clarity.

How are numbers stored by a computer?

Binary Number System

Is a base 2 number system with digits 0, and 1

Example: 1011_2

Very useful when using Boolean Logic (True and False).

Decimal Number System

Is a base 10 number system with digits: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9

Conversion:

$$1011_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 8 + 0 + 2 + 1 = 11_{10}$$

Others: Hexa-Decimal, Octal

Numbers and Precision ...

The way computers store numbers is important when using math

Computers have finite resources and can only represent ranges of values.

Example:

Signed 8-bit Integer can represent values up to $2^7 - 1$

Unsigned 8-bit Integer can represent values up to $2^8 - 1$

Python provides a number of conveniences when working with numbers.

e.g. Integers are limited by memory

Floating Point Numbers

Floating point numbers are often approximate values with varying degrees of precision

Example:

```
In [8]: 1/3
Out[8]: 0.3333333333333333

In [9]: import math

In [10]: math.pi
Out[10]: 3.141592653589793

In [11]: format(math.pi, '0.20g')
Out[11]: '3.1415926535897931'

In [12]: format(math.pi, '0.2g')
Out[12]: '3.1'

In [13]: format(math.pi, '0.2f')
Out[13]: '3.14'
```

Floating Point Numbers

Comparison can be a bit tricky ...

Example:

```
In [15]: 1/3 == 1/3
Out[15]: True

In [16]: 0.3 == 0.1 + 0.1 + 0.1
Out[16]: False

In [17]: 0.1 + 0.1 + 0.1
Out[17]: 0.30000000000000004

In [18]: import math

In [19]: math.isclose(0.1+0.1+0.1, 0.3)
Out[19]: True
```

Python Floating Point Limitations

https://docs.python.org/3.5/tutorial/floatingpoint.html

Additional References

The main reference is:
`http://quant-econ.net/py/learning_python.html`

Additional References:

1. “Think Python”, Allen B. Downey, Oreilly Media
2. “Data Science from Scratch”, Joel Grus, Oreilly Media
3. “Python for Data Analysis”, Wes McKinney, Oreilly Media