Python Workshop - ANU (CBE)

Python Programming and Jupyter Notebooks

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

June 2017

Agenda ...

1. Intro to Programming
2. Intro to Python
3. Tools for Writing Code
4. Jupyter Notebooks
5. Python Language Fundamentals

Nature of programming

At first it may seem like slow progress as you aquire knowledge of what seem to be simple concepts like variable assignment, data types, conditional logic ...

But over time your toolkit grows and pretty quickly you start to piece programs together that do really useful things ...

Computers ...

How do computers work?

Hardware Level

1. CPU (Logic gates, registers, memory ...)
2. RAM (Memory for running programs and data, faster than HDD)
3. HDD (Long term memory, stateful when powered down)
4. Video Card (Hardware interface to output VGA, DVI etc.)
5. USB Bus (Provide connections to peripherals - keyboards and mice)
6. Motherboard (Provides connections between all the components)
7. ...

Having a bit of knowledge of how the system works can help when writing programs that ultimately provide instructions to that underlying system.

Programming

Programming is writing a set of instructions that a machine will eventually follow to perform some work / actions.

1. High Level Languages
   - Increase portability of code
   - higher productivity - does more work in less written code
   - Example: C, C++, Python, Julia ...
2. Low Level Languages
   - Tied to specific hardware architecture (x86, ARM, PowerPC)
   - Example: Assembly language
3. Machine Code (Binary)

Interpreters and Compilation

Code gets translated from high level to low level instructions through:

1. Interpreters
2. Compilers

C is **compiled** and is **statically typed**.

Python is **interpreted** and is **dynamically typed**.

Python can be run **interactively** or in **script mode**

This downside of this translation process is that there are overheads on the conversion from high level code down to the machine level, but the benefits are portability, less code, and more productive.

Even between high level languages there are lots of reasons why they will differ in speed.

Language Comparison: Python

```
>>> print(”Hello World”)
”Hello World”
```

Language Comparison: C

```c
#include <stdio.h>

int main()
{
  printf("Hello world\n");
  return 0;
}
```

Run Program after compilation with gcc:

```
./a.out
"Hello World"
```

Language Comparison: Assembly

```
        .file       "simple.c"
        .section        .rodata
.LC0:
        .string     "Hello world"
        .text
        .globl      main
        .type       main, @function
main:
.LFB0:
        .cfi_startproc
        pushq       %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq        %rsp, %rbp
        .cfi_def_cfa_register 6
        movl        $.LC0, %edi
        call        puts
        movl        $0, %eax
        popq        %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size       main, .-main
        .ident      "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04) 4.8.4"
        .section        .note.GNU-stack,"",@progbits
```

What is Python?

Python is a general purpose programming language[1]

Python has experienced rapid adoption in the last decade and is now one of the most popular programming languages.

`http://githut.info/`

It is currently 3rd most popular on GitHub

[1] started in 1989 by Guido van Rossum

Why Python?

Python is:

1. free
2. a full programming environment
3. easier to learn than some other languages
4. highly productive
    - Large library of user contributed packages
    - high level language design
5. has a large and active community
6. cross platform
7. ...

Provides a powerful environment for scientific research and computation.

Python Usage

Scientific Community

1. Machine Learning
2. Astronomy
3. Artificial Intelligence
4. Chemistry
5. Biology

But also used extensively to manage servers, computing clusters, websites etc. by many companies such as Amazon, Google, ...

Python Features

1. A high level programming language
2. Expansive library support
3. a multiparadigm language (procedural, object-oriented, scripting etc.)
4. Interpreted rather than compiled (Good and Bad)
5. Elegant Syntax (Easy to Read and Understand)
6. Lots of language features (iterators, generators) that allow the language to be highly expressive.
7. Concise

Python 2.7 or 3.5?

Python 2.7

- Pro
  - More packages are available in Python 2.7
  - A lot of examples are written in Python 2.7 syntax.
- Con
  - In maintenance mode - not getting new features as the language develops over time.

Python 3.5 (Best **default** selection)

- Pro
  - Newest version which is the long term future of Python
  - Most of the scientific stack has been ported to Python 3
- Con
  - Sometimes want to use a library which has not been migrated to Python 3 yet. (but can make use of conda environments if needed)

Start python in your terminal: `python`

XKCD Cartoon:

```
>>> import antigravity
```

The language is named python in part because Guido's a big fan of Monty Python's Flying Circus.

The "Zen" of Python

```
>>> import this
```

Ways to use Python

The main ways are:

1. python REPL[2]
2. ipython REPL
3. jupyter

Jupyter notebooks (formerly ipython notebooks) is a really good place to start and will be used extensively in this course.

[2] Read-Eval-Print Loop

IDE's

There are a couple of interesting IDE environments that can also make working with Python a little easier. The best is probably ...

Spyder IDE

You may wish to use it to start with as it provides a TextEditor, a python REPL, and some Data and Documentation Panes that are "integrated" together.

Spyder IDE also comes as part of the Anaconda distribution.

Demo

Ways to write Python Code

Down the track - you may wish to start writing your python code in a full text editor.

1. Sublime Text [My Favourite]
2. Atom
3. Emacs
4. Vim

Many others ...

Why use a Text Editor

Demo

1. Syntax highlighting
2. More productive (tab completion, auto-indentation)
3. Regex and pattern matching
4. ...

Warning: Choosing your “favourite” text editor can become time consuming and endless ... :)

Jupyter

Jupyter is an excellent interactive environment that is used extensively in the Data Science community

Learn more **here**

Supports the notion of executable documents ...

Live Jupyter Demo

Start Jupyter Notebook from the terminal,

```
bash \$ jupyter notebook
```

Jupyter Notebook Topics

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

Swap to “intro-to-jupyter-notebooks.ipynb” notebook

Intro to Python Topics

1. Introductory Example
2. Basic Structure of a Python Program
3. Assignment
4. Data Types

Swap to “intro-to-python.ipynb” notebook

Additional References

The main reference is:
`https://lectures.quantecon.org/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