Week 1: Course Outline and Introduction to Python

Computational Tools and Techniques in STEM

Jan 29-31, 2019

Outline

What is programming?

Step-by-step explanation of how to achieve a task on a computer.

Why computational sciences?

- Computational science is the third pillar of science besides theory and experiment.
- Cross-section of many disciplines and skill sets.
- Can test various theories and augment the existing experiments.
- Some phenomena too complex or unrealistic for experiments.
- Supports reproducible research and open science.

Course Topics

1. Data processing and visualization
2. Version control using Git and Github
3. Navigation in unix systems
4. Fundamentals of object oriented programming
5. Basics of memeory hierarchy and architecture
6. Navigation in supercomputing clusters
7. Code readability and documentation
8. Reading and writing data
9. Data visualization using VisIt software
10. Using external libraries in your code
11. Unit and integrated testing
12. Error catching and debugging
13. Profiling
14. Parallel programming using MPI

Logistics

Class Structure

- Most of the work done during class.
- Live coding demos.
- Hands-on exercises.
- Discussion.

Evaluation

Project based. Last 30 mins of the class every Thu will be dedicated to including concepts learned in the class in your project.

Learning Goals

- **L1**: Installation and setup.
- **L2**: Introduction to Python.
- **L3**: Data types and some commands in Python.
- **L4**: Using Python interactively via conda/python/ipython shell.
- **L5**: Creating and running simple Python scripts.
- **L6**: Python data structures and manipulation.
- **L7**: Using Python modules, such as Numpy.
- **L8**: Getting started with Jupyter notebooks.
- **L9**: Getting started with Spyder.

Installation

- Install Anaconda package from the website `https://www.anaconda.com/download` by clicking on your operating system and following the instructions.
- Test installation.
  - Open the Ananconda prompt by searching for it (Windows users).
  - Or by typing conda list in your terminal.
- Add Anaconda to your path.
  - Unix users: Add this to your .bash_profile or .bashrc file

```
export PATH="/home/.../anaconda/bin:$PATH"
```

  - Mac users: same as Unix
  - Windows users:

`https://docs.alfresco.com/4.2/tasks/fot-addpath.html`

What is Python?

Python is an **interpreted** language (no compilation needed!).

- Clean and simle syntax
- Faster to write programs
- Large number of modules (numpy, scipy, pandas)
- Interface with other languages

Compilation: Conversion to machine language before running the program.

Data Types

- Numbers
  - a
  - integers (`var = 10`)
  - real (`var = 10.0`)
  - complex (`var = 2+3j`)
- Strings (`var = "pooja"`)
- Boolean (`var = True`)

Commands

- `type (var)`
- `print (var)`

Data Structures

- Lists

```
grocery = ["mango", "milk", "bread"]
```

- Tuples

```
grocery = ("mango", "milk", "dairy")
```

- Dictionaries

```
generic_dict = {key : value}
```

- Sets

```
fruitset = {"apple", "cherry", "kiwi"}
```

Exercise 1

- Can you add two lists?
- How to access elements of a list?
- How to access a slice of list?
- Can you have lists of containing different data types?
- Can you create a list that is copy of another?
- Can you convert a tuple into a list and vice-versa?

Other Operations on a List

- `<listname>.append()`
- `<listname>.sort()`
- `<listname>.remove()`
- `<listname>.delete()`
- `<listname>.count()`

Some Writing Style Guidelines (PEP)

- Adding comments in Python.

```
# This is a comment.
"""This is also a comment. But it is a really
large comment with lots of words and spanning
multiple lines."""
movie ="alien"#Demonstrating inline comment
```

- Python convention is to use 4 spaces for indentation.
- For throw-away variables either _ or __ is used.
- Limit all lines to a maximum of 79 characters.
- Line continuation is done by using parantheses (preferred) or backward slash.
- Variable names (`joined_lower`), `camelCase` also used.
- Readability and consistency above everything else.

Exercise 2

```
# Create two lists
list_first = [1, 2, 3]
list_second = [10, 20, 30]

# Copy the first list onto the second
list_second = list_first

# Print the two lists
print "list_first", list_first
print "list_second", list_second

# Change the first list, does the second list change?
list_first[0] = 1000

# Change the second list, does the first list change?
list_second[0] = 9999
```