Introduction to Python basics and Jupyter Notebooks

Valère Lambert and Travis Alongi

UC Santa Cruz GEODES

Goals for today:

- Get a feeling for what a computer program is, how to start writing one and how to run it
- Practice putting some code together in Python and running it
- Feel comfortable knowing how to search for more information

What does a computer do?

Two basic tasks:

- Performs **calculations** – potentially billions of calculations per second (compute power)
- **Remembers** results – potentially 100s of gigabytes of storage (memory)

What kinds of calculations?

- **Built-in** to a programming language
- Those that **you define** as a programmer

Computers do what you tell them to do, or more specifically what your instructions (code) tells them to do

What is a program?

A detailed plan or procedure for solving a problem or performing a task with a computer

Specifically, it is an **ordered sequence of computational instructions** needed to achieve a solution/outcome

What do computers actually understand?

Computers understand data in the form of 0s and 1s – very basic machine language (native language)

They do simple manipulations with those 0s and 1s:

- Move values to different positions in a chain
- Add, multiply, subtract, divide these values
- Compare these values, and if one is less than the other they can follow one step rather than the other

What are programming languages?

A vocabulary and set of grammatical rules (like human language) that can be translated directly to computer language to communicate instructions for specific tasks

In other words, programming languages are a set of symbols and rules that form a bridge between human language and machine language

Types of programming languages

Programming languages can be classified as lower-level versus higher-level

- ➢ Lower-level languages are closer to machine language vs. higher-level are closer to human language, like English

Examples of higher-level languages: **Python**, Matlab, R, HTML, Javascript, C, C++, Fortran

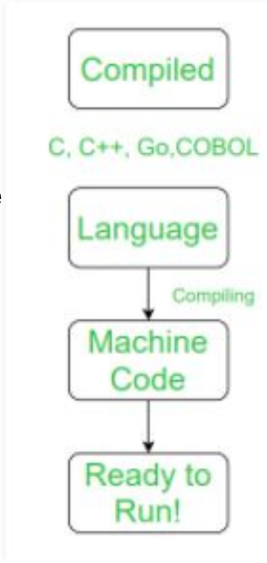
Compiled language:
Code is compiled/translated into machine language and expressed as instructions in machine language

Compilation produces an executable file that is undecipherable by humans

e.g. C, C++, Fortran

Generally faster to run
(already in machine language)

Errors may appear during compilation that prevent code from being translated



geeksforgeeks.org

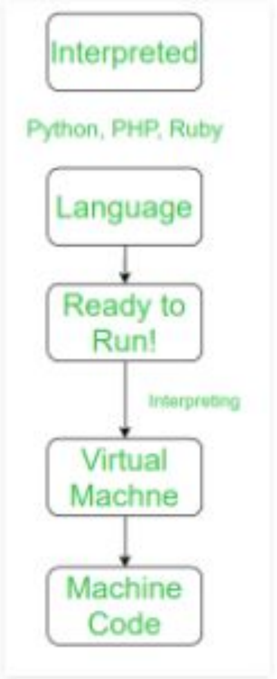
Interpreted language:
Code is interpreted without compiling into machine language instructions.

Instructions are not directly executed by machine but instead read and executed by some other program

e.g. Python, Matlab, JavaScript, Perl

One step between code and execution

Interpreted programs can be modified and debugged while they are running



What is a computing environment?

Many problems are solved by computers making use of multiple computational devices, networks and software to perform different tasks, transfer and storage information.

This **collection of software and hardware used to solve a problem** is the computing environment

What are package and environment managers?

Package managers keep track of what software is installed on your computer

- ➢ facilitate installing new software, upgrading software to newer versions and removing software

Environment managers keep track of collections of software packages used for specific projects

But why?

Imagine you have two different Python projects that use a specific software package. One requires updating the package, however if the package is updated then the other project may not function properly.

An environment manager allows on to set up multiple environments with different version of the same programs installed with different specifics

Examples: Anaconda, Homebrew, Pip

https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

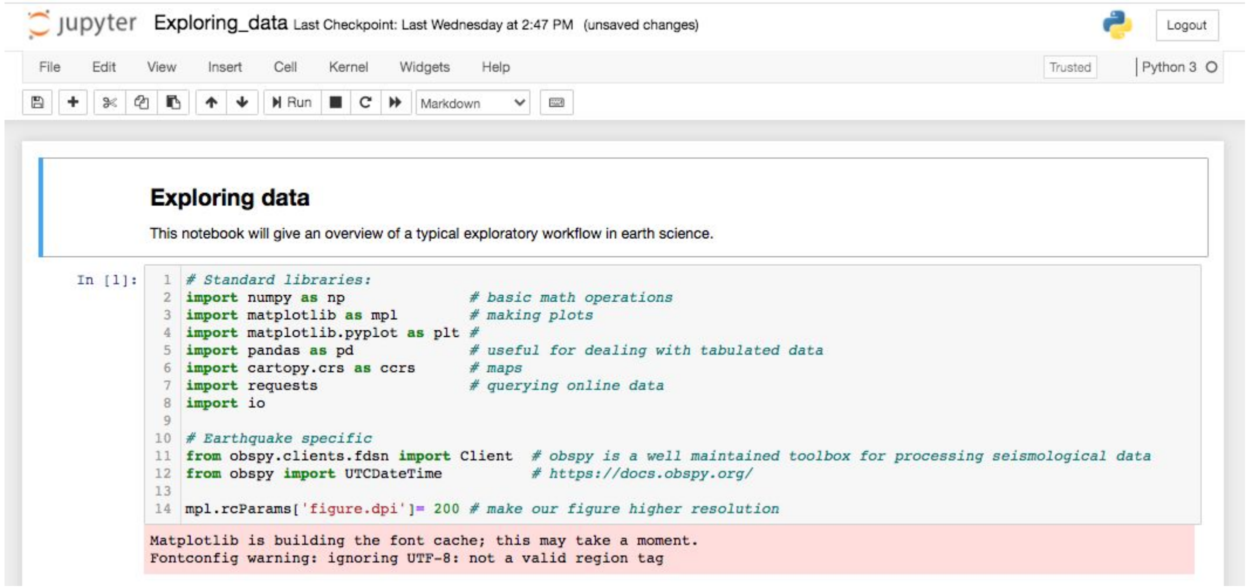
Integrated Development Environment (IDE)

IDEs are software applications that facilitate writing a computer program

- **Editing source code** – create a blank file, write some lines of code
- **Building executables** - contain necessary compilers and interpreters
- **Debugging tools**

May also have special features like syntax highlighting or autocomplete if the IDE knows language syntax

Common IDEs:
Spyder, PyCharm, Jupyterlab, Visual Studio Code, Eclipse, NetBeans, Komodo, XCode



Exploring_data Last Checkpoint: Last Wednesday at 2:47 PM (unsaved changes)

Exploring data

This notebook will give an overview of a typical exploratory workflow in earth science.

In [1]:

```
# Standard libraries:
import numpy as np                # basic math operations
import matplotlib as mpl          # making plots
import matplotlib.pyplot as plt   #
import pandas as pd               # useful for dealing with tabulated data
import cartopy.crs as ccrs        # maps
import requests                   # querying online data
import io

# Earthquake specific
from obspy.clients.fdsn import Client  # obspy is a well maintained toolbox for processing seismological data
from obspy import UTCDateTime          # https://docs.obspy.org/

mpl.rcParams['figure.dpi']= 200 # make our figure higher resolution
```

```
Matplotlib is building the font cache; this may take a moment.
Fontconfig warning: ignoring UTF-8: not a valid region tag
```

Python versus MATLAB

Python is a general purpose, **open-source** language meant to be **easy to read** and simple to implement

- Free and open-source, anyone can download, look at and modify source code – anyone can contribute!
- Generally need to add packages (e.g. Numpy, Scipy, Matplotlib) and want an IDE to write/modify code

Matlab is similar as a high-level interpreted language meant to be relatively easy to use and fast to write code in.

Key differences revolve around Matlab being a proprietary, closed-source software:

- Full package including the language, IDE to write and run code, and many different functions
- Professionally maintained, easy to develop, run and debug code
- **Expensive**, need to pay for additional toolboxes, not everyone can afford MATLAB and thus use your code

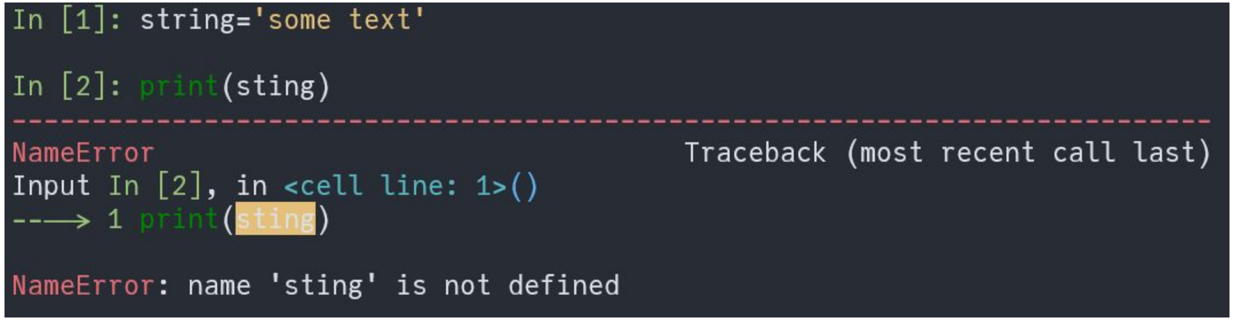
Many of concepts you have learned for Python are relevant to coding in Matlab (and in general), just with basic differences in syntax (vocabulary), significance of white space, calling functions and indexing sequences
e.g. comments in python start with “#” where as in MATLAB they use “%”

Things to keep in mind as a beginner

Computers are **good at some things** and **very bad at others**

- Syntax is very important
- Computer languages are explicit; they're interpreting text with **no** knowledge of language
    - Ie. string='some text'; print(sting) will give you an error

Read the error message, they'll likely tell you the line number in your code that it doesn't like. It might even explain what was expected.



```
In [1]: string='some text'

In [2]: print(sting)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [2], in <cell line: 1>()
----> 1 print(sting)

NameError: name 'sting' is not defined
```

Comment your codes! You'll thank yourself later!

Writing Readable Code

1. **Give objects (variables, arrays, functions, etc) meaningful names**:

Example:

```
Mass      = 25.0; % kg          OR     M   = 25.0;  % kg          RATHER THAN       Var1 = 25.0;
Velocity  = 1.5;  % m/s                 V   = 1.5;   % m/s                          Var2 = 1.5;
```

2. **Define functions for tasks that are regularly performed** and call them throughout your program
   - Make your code more compact rather than copy-pasting the same lines of code (Python example)

3. **Comment you code** – it is challenging to comment code too much!

- Define objects (variables, functions, etc.),
- Define units for objects,
- Describe methods used to solve a problem,
- Motivation for parameter values,
- Create sections of your code with different headings,
- Draw diagrams, coordinate systems, etc.

Comment code as your write it!

Makes it easier for other people (and you!) to:

- Read your code,
- Understand what you intended your code to do,
- Debug your code

Resources

Stack Exchange – online forums where many questions have already been asked and answered

Anaconda cheat sheet
https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

Common IDEs and text editors: Spyder, PyCharm, Jupyterlab, atom, sublime, vim, emacs

Useful python packages for science:
- NumPy (fundamental scientific computing)
- SciPy (data science)
- Matplotlib (2D plotting)
- Pandas (time series analysis),
- Scikit-learn (machine learning),
- Torch/pytorch (machine learning),
- Tensorflow (machine learning - deep learning, neural nets),
- Seaborn (pretty plots),
- Cartopy (maps)