An Introduction to Programming with Python

Carolin Villforth

Why you should learn programming:

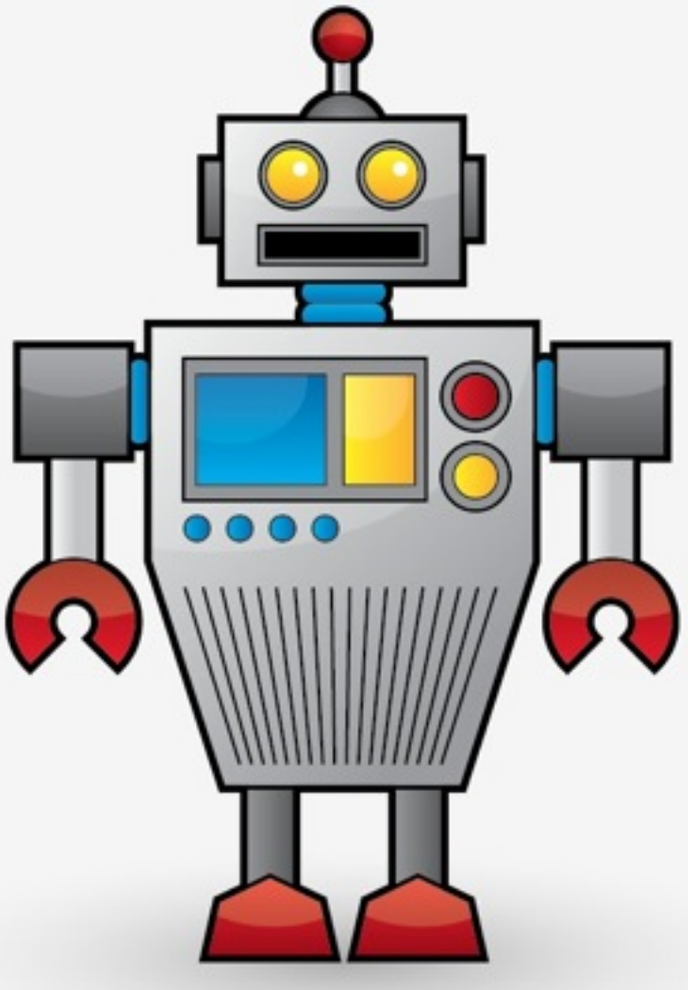
- to perform complex calculations
- for repetitive tasks
- if you want to work in research, you'll have to program.
- if you want to work for any company where you apply statistics, you'll have to program.
- and it's actually kind of fun.

So, what is programming?

- programming means writing down a list of instructions for the computer to execute

- this can reach from executing a few simple commands to performing complex calculations

- the instructions have to follow the syntax of a programming language, which interprets them and passes them on to the computer



http://blog.spoongraphics.co.uk/

Different programming languages

- Compiled languages
  - C, C++, Fortran: good when speed is an issue
- Interpreted languages
  - R, S: popular amongst statisticians
  - Python: see next slide

Why Python?

- Open Source & Free
- intuitive syntax
- easy to learn
- interpreted (no compiling necessary)
- many many packages available for statistics, plotting and computing
- popularity among academics (NASA, LNBL…) and companies (including Yahoo, Google, Nokia, Banks, Gaming Companies) and the CIA uses Python too
- good interaction with C, C++, R, Fortran, MySQL…

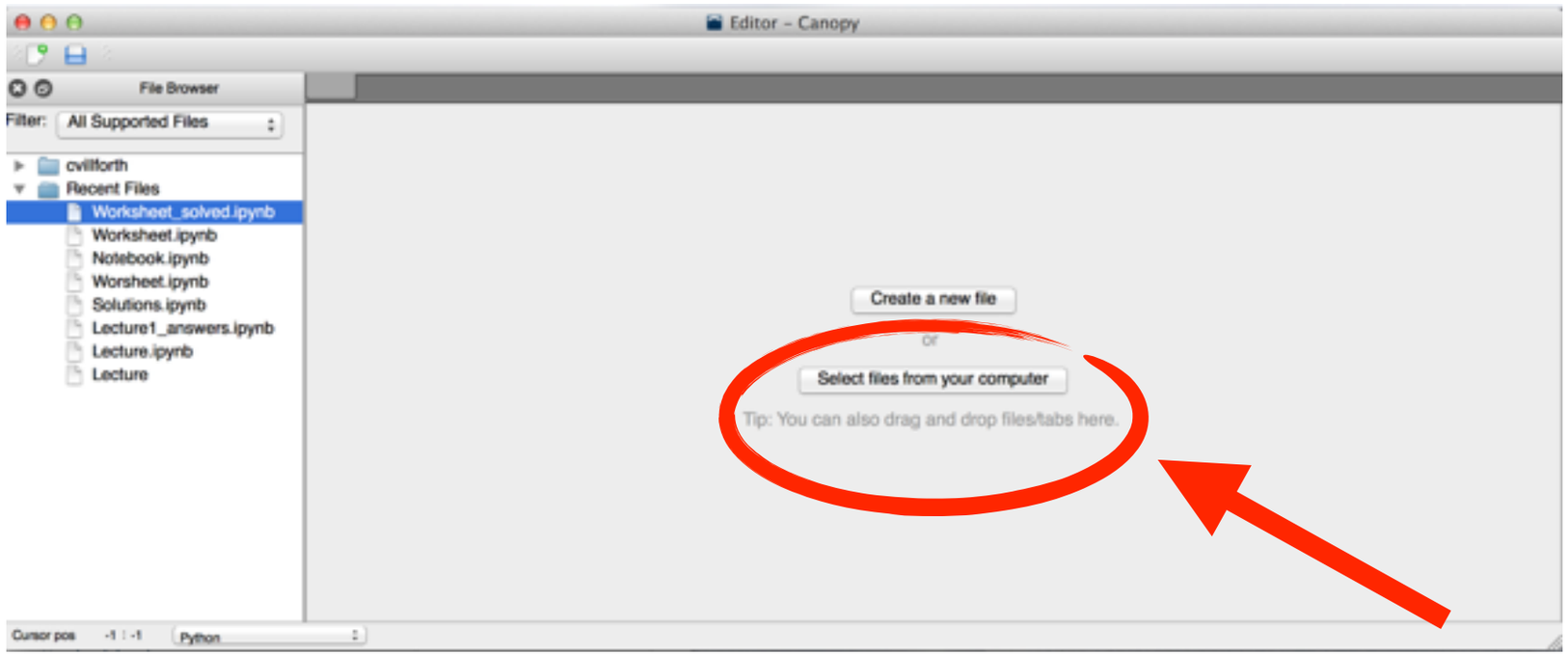
Getting started: opening Python!

- locate the Canopy software on the desktop, it should look something like this:

- double click and wait until you see a window looking like this:

- click on the ‘Editor’ button, which is located here

- you’ll now see a window that looks like this

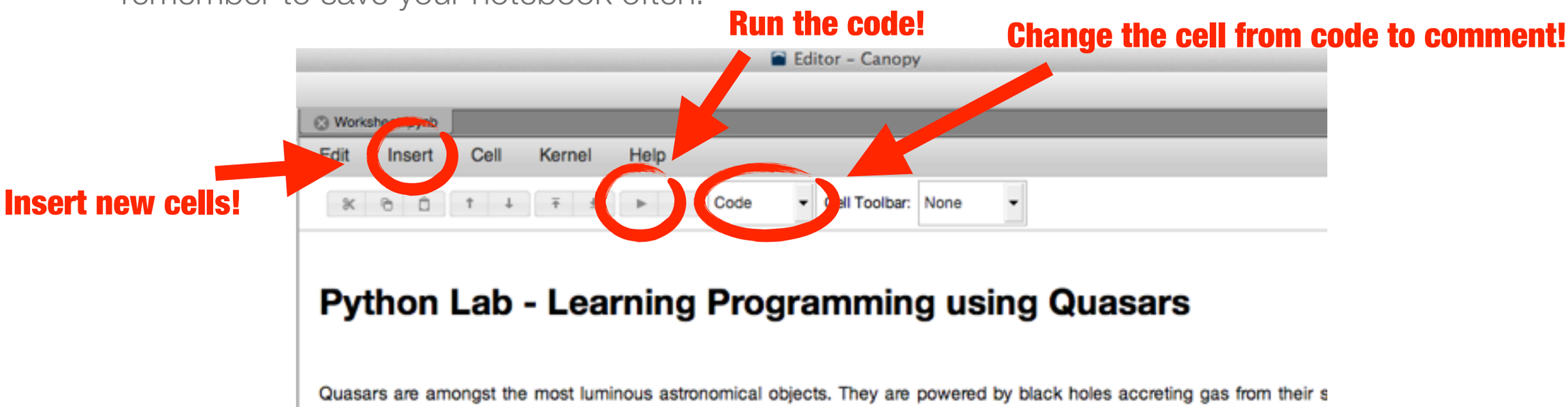


- click here to open the file Worksheet.ipynb

Using the worksheet

The worksheet is an ipython notebook. Here some tips for using it:

- type the code in the cells
- to run the code, press the play button in the toolbar (see below)
- to add more cells, choose ‘insert cell below/above’ (see below)
- if you want to add comments, change the type of the cell in the toolbar (see below)
- remember to save your notebook often!



Cheat Sheet: Column Numbers

---

(0) MI_Z2: Absolute magnitude

(1) BAL_FLAG: flag indicating if signs of ultra-fast outflow from the black hole are detected. 0 means no outflows. >=1 means outflows are detected

(2) FIRST_FR_TYPE: morphology of large scale jets: -1: no radio data, 0: too faint, 1: dominated by emission from close to the black hole, 2: large radio jets

(3) R_6CM_2500A: strength of radio (jet emission) compared to optical emission

(4) LOGL3000: log luminosity at 3000 Angstroem in log(erg/s)

(5) LOGL5100: log luminosity at 5100 Angstroem in log(erg/s)

(6) FWHM_MGII: velocity dispersion of gas near the black hole in km/s, measured using MgII line

(7) redshift: redshift

(8) mag_g: apparent magnitude in the optical

Cheat Sheet - Block 1

- to load data
    - **numpy.loadtxt('My Data File')**
- to access an array line
    - **array[line number]**
- to access an array column
    - **array[:, column number]**
- to access the nth entry in column m
    - **array[n,m]**
- Remember: python starts counting at 0, not 1!
- to get the minimum value in an array:
    - **numpy.min(array)**
- to get the maximum value in an array:
    - **numpy.max(array)**
- to calculate the mean of an array:
    - **numpy.mean(array)**

Cheat Sheet - Block 2

- to plot data
    - **pylab.plot(x, y, ls=*line style*, c=*colour*, marker=*marker style, label=label for data*)**
    - **line styles:** *None: no line, ’-’ : solid line, ‘—’ dashed line, ‘:’ dotted line….*
    - **colours:** *‘r’: red, ‘b’: blue, ‘k’: black, ‘g’: green….*
    - **marker styles:** *‘o’: dot, ‘.’: smaller dot, ‘*’: star, ‘p’: pentagon, ‘^’: triangle….*
- to add a legend
    - **pylab.legend()**
- to label the x-axis and y-axis or add a title
    - **pylab.xlabel(“This is the x-axis label”)**
    - **pylab.ylabel(“This is the y-axis label”)**
    - **pylab.title(“This is the figure title”)**
- to plot a histogram:
    - **pylab.hist(data) or**
    - **pylab.hist(data, bins) where bins can be an array with the bin edges or the number of bins**
- to create arrays:
    - **numpy.arange(start, end, stepsize)**
    - **numpy.linspace(start, end, number of steps)**

Cheat Sheet - Block 3

- calculations with arrays
  - **array + 5: adds 5 to each entry of the array**
  - **array1 + array2: adds the ith entry of array 1 to the ith entry of array 2 (must have the same size)**
  - **other types of calculations: *: times, **: to the power of, /: division, -: subtraction**
  - **useful commands: numpy.log10(), numpy.sqrt(81), numpy.mean(), numpy.min(), numpy.max()**

- function definition basics:
  - **remember to add indents**
  - **remember to “return” your result in the last line of the function**

```
In [85]: def addnumbers(a, b):
             output = a + b
             return output
```

Cheat Sheet - Block 4

- fitting a line to data:
    - **numpy.polyfit(x, y, order)**
    - **order=1: line, order=2: quadratic function……**
    - **result is an array where the first entry is the slope and the second entry is the intercept**
- masking arrays:
    - **array[mask] masks an array**
    - **to create a mask, use expressions such as “array > 14”, “array == 5”, “array <= 25”…..**
    - **example: array[array > 5] creates an array that contains all entries from the array that are greater than 5**
- to calculate the correlation coefficient:
    - **import scipy.stats**
    - **scipy.stats.pearsonr(x, y), returns the correlation coefficient and the p-value**

Welcome to the world of programming!

from www.helloworldprogram.com