Introduction to Base Camp

Intro & Data Types

September 6, 2021
Samantha Breslin

Overview

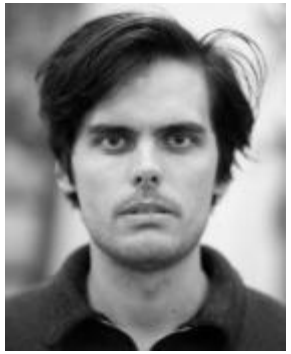
- Overview of course
- Tools & landscape of programming in the Base Camp
- Social Data Science Programming Community
- Variables & Operations
- Groups

People

Core teachers


Samantha


Friedolin


Asger


Greg

TAs


Vero


Mònica


Pepa

WELCOME

- Fundamentals of Programming
- Fundamentals of Data Collection (largely from a programming standpoint)
- Basic Quantitative Data Analysis
- Some Basic Qualitative Data Analysis (2 lectures)

Base Camp

- Course outcomes:
  - To climb the mountain and all reach the base together!
  - To prepare to scale the summit (tackle advanced Python in later courses)
- Learn programming from a social data science perspective
- You will be put in "coding groups" (end of the class today)
  - Help each other, learn from one another!
  - Take initiative and explore (if you have time)

Schedule

Mondays	Wednesdays
Lecture 8:15-10:00 With 15 min break (1.1.18)	Lecture 8:15-10:00 With 15 min break (1.1.18)
Exercises 10-13 (1.1.12 1.0.10 2.1.02)	Exercises 10-13 (1.1.12 1.0.10 2.1.02)

Official

Mondays	Wednesdays
Lecture 8:30 – 10:00 With 5 min break (1.1.18)	Lecture 8:30 -10:00 With 5 min break (1.1.18)
Exercises 10-13 (1.1.12 1.0.10 2.1.02)	Exercises 10-13 (1.1.12 1.0.10 2.1.02)

Practice

Two Friday Classes

- Friday Oct 8: Netnography
- Friday Nov 5: Review

See the syllabus!

Schedule: Part I – Fundamentals of Programming & Data Collection

- Week 36: Beginnings
- Week 37: Programming Structures
- Week 38: More Structures
- Week 39: Review and Pandas
- Week 40: Getting Data
- Week 41: Getting Data II
- Week 42 & 43: NO CLASS
- Week 44: Visualization, Functions, & Review
- Week 45: Basic Data Analysis

Schedule: Part II – Basic Data Analysis

- Details to be posted on Absalon

Exercises & Exam

- The exam will consist of submitting code to collect and process data in order to produce a dataset of the student's choosing, along with a description and reflection on how they constructed the dataset. The code must be in the form of a Jupyter Notebook. Within the Notebook, students will also be required to conduct a basic analysis on that dataset in accordance with the Learning Outcomes.
- To be eligible for the exam in Social Data Science Base Camp, it is a requirement that students have completed and submitted all of the exercise assignments via Absalon prior to the exam start date. Each class-day will have an associated exercise assignment (max. 28 Jupyter Notebooks).
- https://kurser.ku.dk/course/asdk20001u/2021-2022

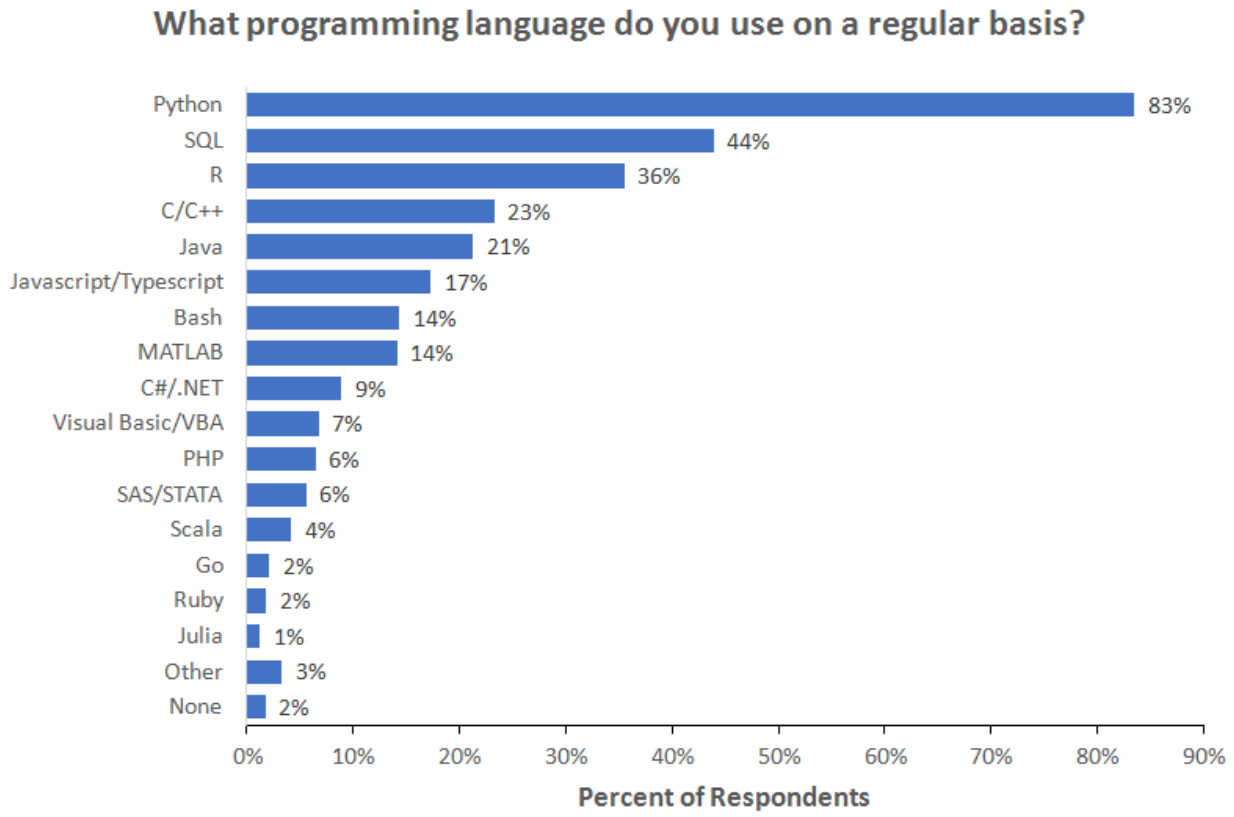
Exam cont'd

- Pass/Fail: the goal is practice and support in learning to program
- Submitted individually
- Detailed guidelines will be posted on Absalon
- Final deadline for exercise assignments: December 20, 2021
- Exam Submission Deadline: January 17, 2022
  - Re-exam deadline: February 7, 2022

Why Python

- Relatively straightforward language to learn
- General purpose, can do most things
- Good resources for data analysis, machine learning, visualization, etc (all things we want to learn and do!)

Languages and Data Science



Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018. A total of 18827 respondents answered the question.

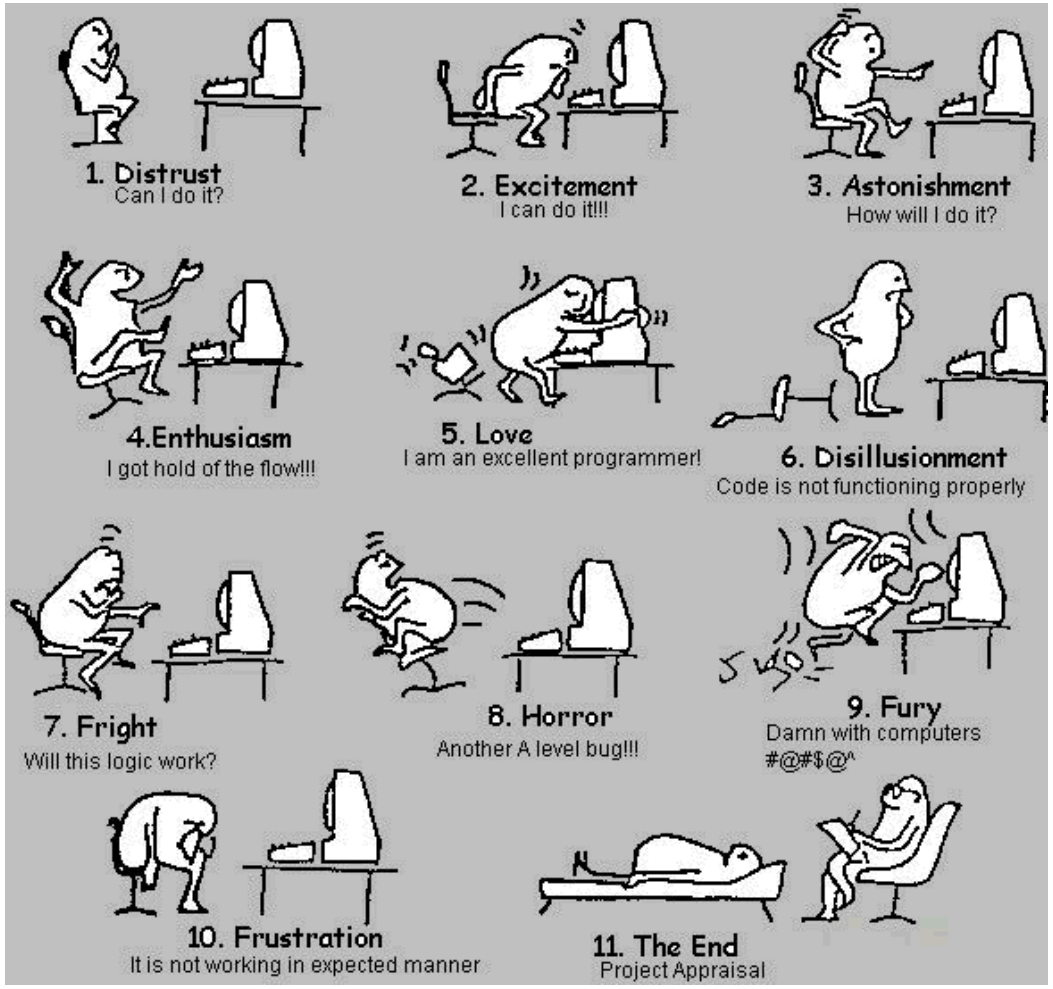
BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

Landscape of tools

- Your computer!
- Python Interpreter (the translator)
- Jupyter Notebook
- Shell

Learning to code I

- Is a lot of work
- Possible for everyone
- Can cause a lot of emotions, including frustration, anger, etc.



Programming as social science

- We are opening the “black box”
- We are programming for exploration, method, problem solving, and more
- We are creating a programming as social science community
- We consider the history, context, and politics of our methods (i.e. gendered practices)
- We are both learning specialized programming (e.g. for data collection and analysis) and general programming practices

Learning to code II

- Coding is a form of communication
  - Between you and the computer
  - Between you and other people
  - Between you and your discipline
- Errors are a form of communication
  - It is the computer (actually the past programmer) trying to tell you something is wrong
  - The computer and past programmer trying to work with you to fix the code

Learning to code III

- Where to get help:
  - The international programming community (Google, Stack Overflow)
  - Your classmates (Your groupmates, your core groups, other classmates)
  - The TAs
  - Us (Samantha, Friedo, Asger, Greg, by email, Absalon discussion group)

Programming community norms

Online (e.g. Stack Overflow)

- May differ from this class
- Consider anthropologically, not personally

This class ("Grilled cheese programming")

- Supportive not judgmental
  - Help each other
  - Practice and contribute
  - Learn by teaching
- Reflexive about methods
- Focus on readability

Learning to code IV

- Reuse code
  - Type out everything - do not copy and paste
- Think then write
- Practice
- If you’re helping someone else, don’t take over the keyboard

Variables & Operations

Jupyter I

- To start Jupyter notebook:
  - Type: `jupyter notebook` in the shell
  - Open Anaconda and then Jupyter Notebook
- To make a new notebook:
  - Navigate to where you want your notebook to be stored
  - Click the `New` button in the right corner
  - Click on `Python 3`

Try making a new notebook

Jupyter II

- Jupyter works with cells where you can put code
  - A cell with a **green** bar to the left is in edit mode
  - A cell with a **blue** bar to the left is in command mode

Jupyter III

- To add code
  - Click on a cell, or press enter, to go into edit mode
- To run your code:
  - Click the ► button
  - Or, press SHIFT + ENTER

Jupyter IV

- Add cells:
  - Click the + symbol
  - In command mode, type “a” to add a cell above
  - In command mode, type “b” to add a cell below

Try adding a cell using each of the methods above.

In your new cell type `print(“Hello Base Camp!”)` *and run your cell.*

Jupyter V

- More resources:
  - This blogpost from Social Data Science 2019: https://abjer.github.io/sds2019/post/jupyter/
  - General resources and documentation here: https://jupyter.readthedocs.io/en/latest/

Jupyter Markdown

- You can change a cell from Code format to Markdown format
  - In the dropdown menu above choose “Markdown”
- This is one way of adding comments and information about your code

Jupyter Markdown

Try adding a cell and changing it to Markdown mode. Type:

```
# Heading 1
## Heading 2
### Heading 3

I have added comments! :)
```

Run your cell.

- See here for more, including numbered lists, images, links, etc.: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Commenting (in a code cell)

- # for a single line comment
- """ works for multiline comment """
  - Easier (less typing) but not recommended → defines a text constant and produces output
  - Could cause unexpected errors
  - Solution: # for every line

Data Types

- There are four fundamental data types: `int`, `float`, `bool`, and `str`
- Assigning a value to a name (like using x & y in math)
  - `A = 5`
  - `B = 'I am a string'`
  - `C = 4.3`
  - `D = True`
- Names should be meaningful (not just A, B, C, D) – more on this later
  - Helps with readability of your program
  - May help make explicit assumptions in your code
- Variables persist throughout your notebook (usually – also more next week)

Changing Data Types

- Data types can (sometimes) be converted into one another
  - `int(1.6)` will convert the `float 1.6` to the integer `1` (it will always round down)

What will be the value of `float(int(1.6))`? Why?

Operators

- Numeric operators: take numeric input (i.e. int or float), does a mathematical calculation, and outputs a numeric value
  - +; * (multiplication); -; /
  - % (modulus); ** (exponent); // (floor division)
  - abs(x); round(x)
- Comparison operators: Outputs a boolean value (True or False)
  - == ; != (equal or not equal, can accept most types as input)
  - < ; > (less than, greater than, accepts numeric input)
  - <= ; >= (less than or equal to; greater than or equal to, accepts numeric input)
- Logical operators: Takes a boolean input and outputs a boolean value.
  - and ; &
  - or ; |
  - not ; !
- More here: https://www.tutorialspoint.com/python/python_basic_operators.htm

Objects and Methods

- Everything in Python is an object
  - Related to what is known as 'object oriented programming'
  - Like human languages, programming languages (re)produce certain worldviews
- Objects have methods
  - Methods are things that the object can do
  - Looks like: `variable_name.method(x)`

In your notebook, create a float my_float:

```
my_float = 3.0
```

You can check and see if your float is an integral value (i.e. essentially an integer) using the is_integer() method.
Try and see what the answer is, then try a different value for my_float.

Documentation

- The Python programming language has documentation explaining all of the built in types and functions
  - See here for basic types: https://docs.python.org/3/library/stdtypes.html
  - See here for overall documentation: https://docs.python.org/3/contents.html

Strings I

- Strings are containers of characters (more on this next week too)
- A way of storing text
- Can do many things with strings

Strings II

- Combining strings (concatenation)
  - Add them!

```
s1 = 'social'
s2 = 'science'

print (s1 + ' ' + s2)
```

What happens if you don't include the ' '?

Strings III

- the string type comes with a variety of methods
  - `upper()`
  - `lower()`
  - `capitalize()`

Substrings

- A substring is a part of a string
  - A string is always a substring of itself
  - `in; not in`

Is 'soc' a substring of s1? How do you check?

Substrings II

- You can replace a specific substring
  - `replace(x,y)`

Replace 'ial' in s1 with 'iology' and then print s1. What happened?

Debugging

- Reminder: errors are normal, and a way for the computer and past programmers to communicate with you

In your notebook, add a cell, go into edit mode and add the following code:

```
my_string = 'I am a string'
int(my_string)
print(my_string)
```

Run your code. What happens?

Debugging II

- Inspect the error message
  - Where is the error?
    - what line number?
  - What is the error?
    - Reread several times
    - Look at the documentation if the error seems to refer to basic function
    - Search on Google if you don't understand what it is
- Common errors that the interpreter helps you with:
  - SyntaxError: spelling
  - ValueError: datatype mismatch
  - TypeError: information given to a method is not the right type or missing
- Common errors that the interpreter might not spot:
  - Logic errors: the program doesn't do what you want/expect
  - Undetected errors: When the program hasn't been thoroughly tested

Groups

- Found on Absalon under Course Information
  - F21 – Basecamp – Programming Groups
- Meet your group members
- Sit with your group members in your exercise sessions
- Matching of members by programming level is not perfect

Exercise Sessions

- Support members in your group and your class
- Ask the TA for help – that's why they're there!
- Can work in pairs within your group
  - Discuss what your group prefers
  - Switch up pairs across classes

Exercise Notebooks

- Work through exercises
- Submit exercises for the week by Friday at noon
- TAs will review exercises the following Monday
- The goal is practice!