PPOL564: Data Science I

Unit 01: Intro, Setup, and Python Basics

Goal for today's session

- Course goals
- Intros
- Nut and bolts: course website and specific course components
- Python basics: lists; basic list comprehension; numpy arrays

Where we are

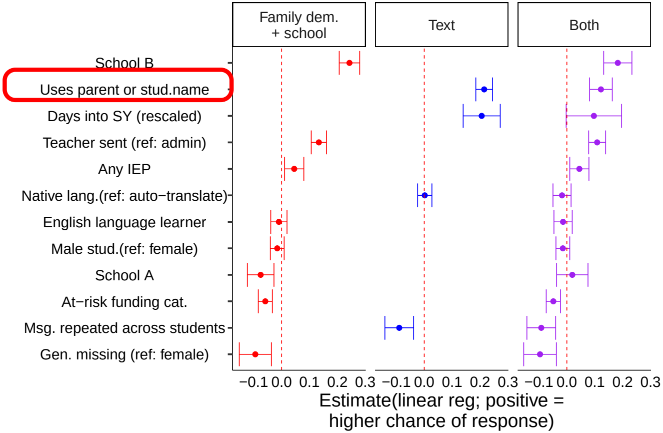
- **Course goals**
- Intros
- Nut and bolts: course website and specific course components
- Python basics: lists; basic list comprehension; numpy arrays

Broad goals

- Get you started on Python and SQL for applied data science; lay the foundations for the remainder of the core sequence
- Two components
  1. **Workflow tools:** Git/GitHub; LaTeX; basic use of command line
  2. **Programming in messy contexts:** applied tasks in Python (data wrangling; basic text analysis); some SQL

An example

Graph from a recent talk; box in red shows that parents are more likely to respond to text messages from teachers when the teacher uses the parent or their child's name:

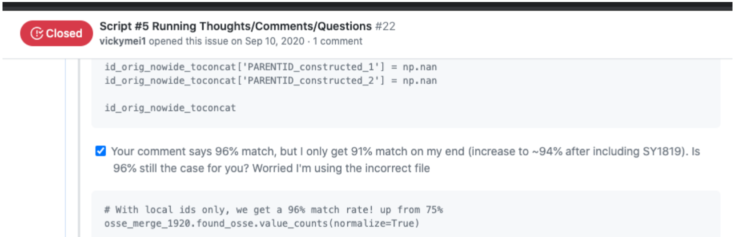


Beyond the statistics, series of workflow and programming tasks before running regression

1. **Acquire the data:**
   - **Ideal:** csv or database
   - **Real:** excel file w/ variable number of tabs and spaces in column names; pdfs containing text; website
2. **Clean the data:**

p_name	s_name	msg_content
Rebecca Johnson	Jennifer Johnson	Hi Ms. Johnson! Jenny did great on her math test.
Rebecca Johnson	Jennifer Johnson	Hello Rebecca- I'm concerned about Jennifer's grades.

3. **Reconcile different decisions in data cleaning:**



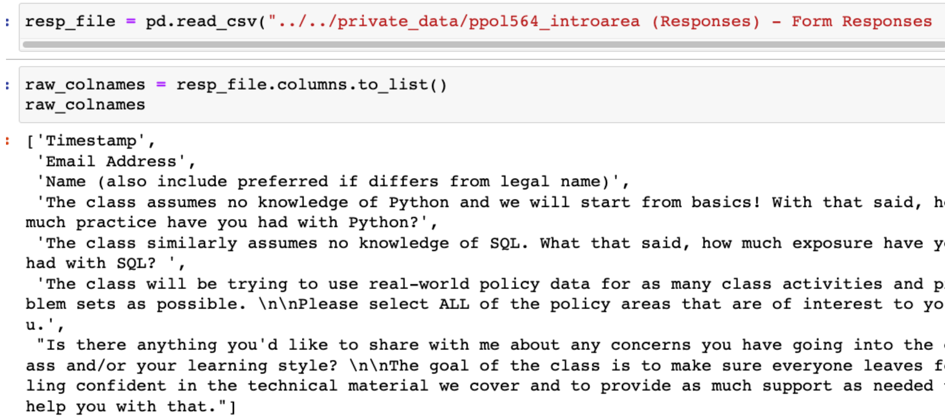
Script #5 Running Thoughts/Comments/Questions #22
vickymei1 opened this issue on Sep 10, 2020 · 1 comment

```
id_orig_nowide_toconcat['PARENTID_constructed_1'] = np.nan
id_orig_nowide_toconcat['PARENTID_constructed_2'] = np.nan

id_orig_nowide_toconcat
```

- [x] Your comment says 96% match, but I only get 91% match on my end (increase to ~94% after including SY1819). Is 96% still the case for you? Worried I'm using the incorrect file

```
# With local ids only, we get a 96% match rate! up from 75%
osse_merge_1920.found_osse.value_counts(normalize=True)
```

To reiterate the workflow before data are usable...



```
: resp_file = pd.read_csv("../../private_data/ppol564_introarea (Responses) - Form Responses
```

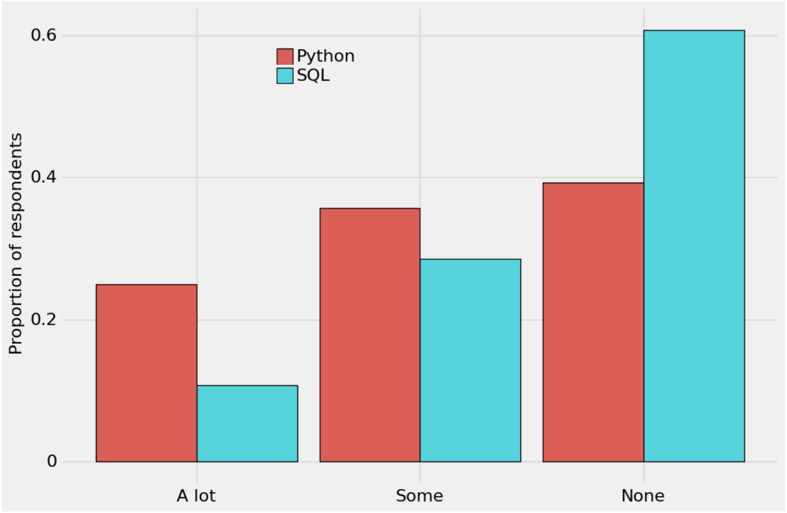
```
: raw_colnames = resp_file.columns.to_list()
  raw_colnames
```

```
: ['Timestamp',
   'Email Address',
   'Name (also include preferred if differs from legal name)',
   'The class assumes no knowledge of Python and we will start from basics! With that said, h
  much practice have you had with Python?',
   'The class similarly assumes no knowledge of SQL. What that said, how much exposure have y
  had with SQL? ',
   'The class will be trying to use real-world policy data for as many class activities and p
  blem sets as possible. \n\nPlease select ALL of the policy areas that are of interest to yo
  u.',
   "Is there anything you'd like to share with me about any concerns you have going into the 
  ass and/or your learning style? \n\nThe goal of the class is to make sure everyone leaves f
  ling confident in the technical material we cover and to provide as much support as needed 
  help you with that."]
```

To reiterate the workflow before data are usable...

```python
def clean_onecol(one_col: str,
                 cutoff = 5):
    """
    Take in a messy column name and return a
    cleaned one

    @param one_col: Messy column name
    @param cutoff: number of tokens to cut the string at (default 5)

    @return:  clean column name
    """

    l = one_col.lower()
    l_nosp = re.sub(r"\s+|\,|\/|\(|\)|\?|\.", '_', l)

    ## tokenize
    l_nosp_token = l_nosp.split("_")

    ## if longer, add some remainder back in to
    ## differentiate similar q's
    if len(l_nosp_token) > cutoff+5:
        random.seed(2021)
        l_short = "_".join(l_nosp_token[:cutoff]) + \
                  "_".join(random.sample(l_nosp_token[cutoff:], 5))

    ## otherwise keep short
    else:
        l_short = "_".join(l_nosp_token[:cutoff])
    return(l_short)
```

Python and SQL exposure



Some FAQs from the intro survey

- **Q:** how much prior Python/SQL/data science knowledge is needed to do well in the course?
  - **A:** None! The course is designed to take you from no previous exposure to these languages to be able to complete and do well on all assignments (there are no exams; only project-based psets and a final project). We're also providing extra support through the bootcamp to get everyone to same place.
- **Q:** are there materials you recommend for extra self study?
  - **A:** the course is designed to be self-contained through the combination of slides + in-class activities + DataCamp + psets. But if you need extra help with certain concepts after going through the assigned materials, we can provide extra content-specific online resources.

Some FAQs from the intro survey

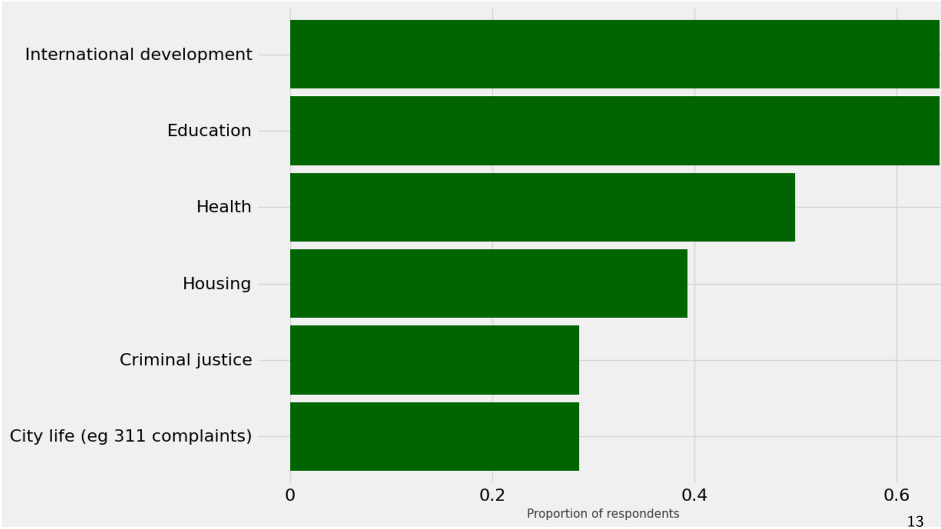
- **Q:** if I already have substantial experience w/ Python, is there a way to provide me with more advanced material?
    - **A:** (1) as noted on syllabus, you can skip all DataCamps and apply that portion of grade to psets; (2) starting with pset two or three, we'll have optional extra credit questions (that you can still get an A in course without completing); and (3) come to me and the TAs
- **Q:** will this class/can this class incorporate statistics/causal inference into the problem sets/learning?
    - **A:** because the DSPP program is covering that material concurrently with rather than before this class, no problem sets draw on specific concepts from causal inference or statistics; however, if you want to use those tools in the final project you're very welcome to

Where we are

- Course goals
- **Intros**
- Nut and bolts: course website and specific course components
- Python basics: lists; basic list comprehension; numpy arrays

Policy areas of interest

Main ones listed in other: elections; energy/the environment; technology



Going around....

- Preferred name
- (Briefly) what you were up to prior to the DSPP
- If you could have any data source at your disposal, what would it be and what's a question you would ask?

A bit about me

Where	What	Languages
	Psychology; economics; MA in ethics/philosophy; internships in consulting	stata
BIOETHICS AT THE NIH	Research fellow at NIH dept of bioethics	None
PRINCETON UNIVERSITY	PhD in sociology, demography, and social policy	R
DATA SCIENCE FOR SOCIAL GOOD THE LAB @ DC OES Office of Evaluation Sciences	Data scientist	python™

Course TAs: Sonali and Yifan

Contact info. and office hour details at this page: https://rebeccajohnson88.github.io/PPOL564_datascience1_fall22/docs/index.html

- In addition to helping with problem sets/grading, first point of contact for questions via Slack

DSPP bootcamp: additional help for the first few weeks

Leaders: Gloria Li (gl587@georgetown.edu) and Vince Egalla (ve68@georgetown.edu)

- Mix of drop in office hours and two content-focused classes reiterating course material
- All on this zoom: `https://georgetown.zoom.us/my/gloriali`
- See Canvas for specific schedule; first session Thursday 5:30-7 pm

Where we are

- Course goals
- Intros
- **Nut and bolts: course website and specific course components**
- Python basics: lists; basic list comprehension; numpy arrays

Course website: most authoritative guide

Please make sure to read the following pages most closely (can click on links in posted slides):

1. **Evaluation and grading**: `https://rebeccajohnson88.github.io/PPOL564_datascience1_fall22/docs/eval_grades_py.html`- covers four late days for problem sets and exact grade breakdown
2. **Software setup**: `https://rebeccajohnson88.github.io/PPOL564_datascience1_fall22/docs/software_setup.html`
3. **Course schedule (more subject to change):** `https://rebeccajohnson88.github.io/PPOL564_datascience1_fall22/docs/course_schedule.html`

Course components

1. **Most important - in-person class sessions:** mix of lectures and hands-on practice in small groups (after lecture 1); problem sets entail more advanced applications
2. **Slack**
3. **Office hours**
4. DataCamp for review/basic syntax
5. Six problem sets
6. Final project

Structure of typical in-class session

Time window	**What**
12:30-1:30	Slides; DataCamp questions
1:30-1:40	Break and get into small groups
1:40-2:20	Work with group on in-class activity; I'll circulate for questions
2:20-2:30	Break
2:30-2:55	Reconvene as a group and go over solutions/questions you ran into; outline any prep for next class
2:55-3:00	Anonymous sticky notes with remaining questions

- Might deviate as we have visitors (might have guest speakers working in data science and public policy if there's interest and if we have spare course time)
- Currently not taking attendance but might start if issues

Slack: course communication

- Forum for you to communicate with the TAs (I'm on in case you need to DM me about private matters or if the TAs point me to question)
- Please add an image and preferred name to your profile by next week's class
- **Expectations:**
    - If in doubt, always default to a public channel so that others can benefit from your question
    - **Order:** tag the TAs and then they can defer to me if they have problems answering
    - **DMs to me**: use rather than email; reserve only for family emergencies and other personal issues [do not need to DM me if missing class]
    - Problem sets are typically due on Friday night; TAs will respond to all questions posted before **5 pm** on due date but not questions between 5 pm and midnight when due

Office hours

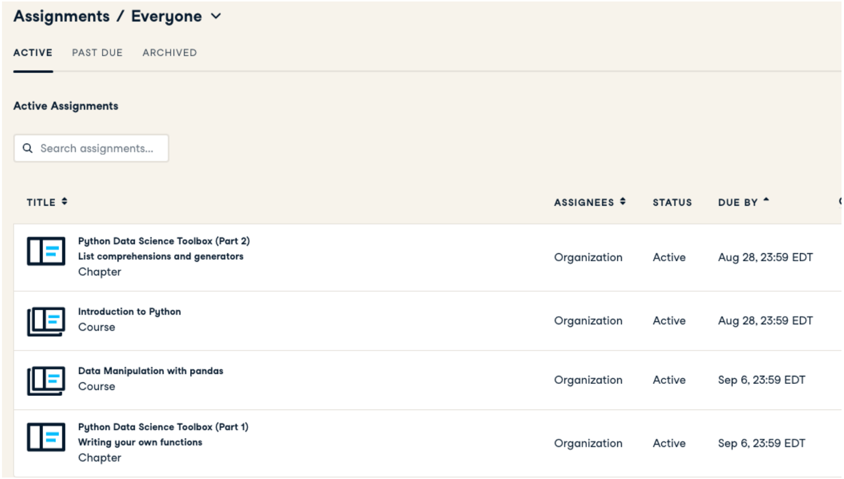
See schedule on course page for links to Calendly and Zoom where relevant; currently:

- **Sundays:** Yifan 7-8 pm
- **Mondays:** none
- **Tuesdays:**
  - **Prof:** 10:30-11:30 AM
  - **Yifan:** 4:15-5:30 PM
- **Wednesdays:** Sonali 3-4 PM
- **Thursdays:** Prof 10:30-12:00 PM
- **Fridays:** Sonali 1:30-3:00 PM

Course components

1. Most important - synchronous sessions: lab-based rather than lecture-based; hands-on practice with more advanced applications / work on problem sets
2. Slack
3. Office hours
4. **DataCamp for review/basic syntax**
5. **Six problem sets**
6. Final project

DataCamp: join group for course



Assignments / Everyone

ACTIVE PAST DUE ARCHIVED

Active Assignments

Search assignments...

TITLE	ASSIGNEES	STATUS	DUE BY
Python Data Science Toolbox (Part 2) List comprehensions and generators Chapter	Organization	Active	Aug 28, 23:59 EDT
Introduction to Python Course	Organization	Active	Aug 28, 23:59 EDT
Data Manipulation with pandas Course	Organization	Active	Sep 6, 23:59 EDT
Python Data Science Toolbox (Part 1) Writing your own functions Chapter	Organization	Active	Sep 6, 23:59 EDT

- Tag Sonali in `#datacamp_questions` if you're having trouble seeing our class page
- Graded complete/not complete; if want to skip, can reapportion the

Six problem sets

- Pset one posted on Canvas: https://georgetown.instructure.com/courses/158038/assignments/792619
- **Problem set one: due Friday 09-02 at 11:59 pm**
- **Others:** see schedule on course website
- **For these**:
  - Start well in advance (at least 3-4 days) and space out the parts
  - May devote some class time pre deadline to work on the pset/answering questions
  - For ones where relevant, will provide intermediate/cleaned data so that getting stuck on early parts doesn't impede later parts
  - For certain psets, we'll allow partners (randomly assigned for pset two)

Ways to get help on problem set

- **Office hours with me and the TAs**
- **Post on GitHub issues and TA or I will answer within 24 hours during weekdays; by sunday night if weekend**

Course components

1. Most important - synchronous sessions
2. Slack
3. Office hours
4. DataCamp for review/basic syntax
5. Six problem sets
6. **Final project:** applied data science project in teams of 3-4 students; we'll give you a choice of data sets and you'll produce: (1) a short 10-page report and (2) a complete GitHub repo that goes from raw data $\implies$ final analyses. More details in a couple weeks

Project examples from past classes

Focus one year: legal oversight of agricultural employers that employ temporary guestworkers to prevent labor abuses of those workers

- Geo-visualization of locations of job sites relative to Census tract attributes (e.g., migration rates; unemployment): https://github.com/rebeccajohnson88/qss20_s21_proj/blob/main/memos/final_papers/dol_geocoding_writeup.pdf
- Causal analysis of relationship between inspection capacity and findings of legal issues: https://github.com/rebeccajohnson88/qss20_s21_proj/blob/main/memos/final_papers/dol_opmstaffing_writeup.pdf
- Natural language processing of job contracts: https://github.com/rebeccajohnson88/qss20_s21_proj/blob/main/memos/final_papers/dol_textasdata_writeup.pdf
- Supervised machine learning predicting investigations/violations: https://github.com/rebeccajohnson88/qss20_s21_proj/blob/main/memos/final_papers/dol_predictviol_writeup.pdf

Checklist (by Friday end of day)

1. Are you on Slack and have you filled out your profile (photo or avatar; preferred name)?
2. Are you set up on DataCamp and working on the assignment due Sunday night?
   - Ping Sonali #datacamp_questions with your email if need adding
3. Do you have Python installed and a way to load/edit Jupyter notebooks for problem set one?
   - Visit office hours and ping on public Slack if issues installing (if you have installation issue others likely do too)

Where we are

- Course goals
- Intros
- Nut and bolts: course website and specific course components
- **Python basics: lists; basic list comprehension; numpy arrays**

Why Python?

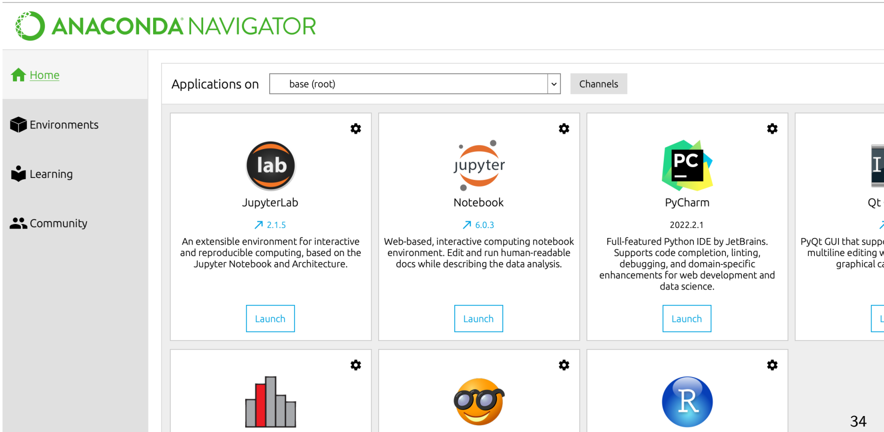
- Four programming languages used to varying degrees in data science: Python; R; STATA; and SAS
- Python: has superior libraries for machine learning, natural language processing, and other techniques previewed in this course and later in the sequence
- Open-source which means free to download and use

Terminology/ways of interacting with code in Python

- IDE (integrated development environment): place to write and test code
- Scripts: `this_is_my_script.py`
  - IDEs: Any program for working with text files (eg Sublime text; Vim; Atom); Spyder; PyCharm
  - When: larger datasets; "production-ready" code
- Notebooks: `this_is_my_script.ipynb`
  - IDEs: **Jupyter Notebooks**; Google colab
  - When: smaller datasets; testing code; integrating code, text, and figures
- **Focus at beginning:** notebooks that we edit in Jupyter

How do I open a jupyter notebook? Point and click way

1. Make sure Anaconda is installed (see course website)
2. Go to Applications and click on the Anaconda-Navigator icon
3. Something like this should pop up (be patient). Click on the **Launch** icon under Jupyter Notebook



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

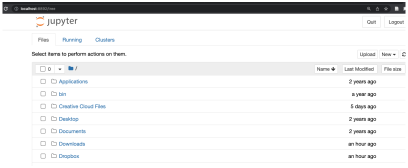
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

PyQt GUI that supp
multiline editing w
graphical ca

How do I open a jupyter notebook? Point and click way

4. That should open up a browser window or tab with your full set of directories/files


localhost:8892/tree

jupyter

Quit Logout

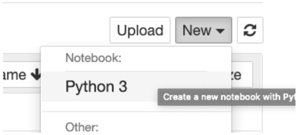
Files Running Clusters

Select items to perform actions on them.

Upload New

0 /	Name	Last Modified	File size
Applications		2 years ago	
bin		a year ago	
Creative Cloud Files		5 days ago	
Desktop		2 years ago	
Documents		2 years ago	
Downloads		an hour ago	
Dropbox		an hour ago	

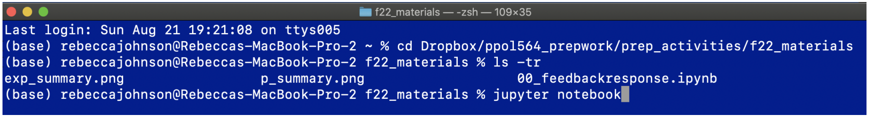
5. Then:
   - To create a new notebook, click New in right hand corner



   - If opening an existing notebook, navigate to folder where it's stored, click on the file, and notebook should open in a new browser tab

How do I open a jupyter notebook? Terminal way

1. Navigate to folder where you want to create a new or open an existing notebook
2. Type `jupyter notebook` into your terminal (Mac) or terminal emulator (Windows)



```
Last login: Sun Aug 21 19:21:08 on ttys005
(base) rebeccajohnson@Rebeccas-MacBook-Pro-2 ~ % cd Dropbox/ppol564_prepwork/prep_activities/f22_materials
(base) rebeccajohnson@Rebeccas-MacBook-Pro-2 f22_materials % ls -tr
exp_summary.png                p_summary.png                  00_feedbackresponse.ipynb
(base) rebeccajohnson@Rebeccas-MacBook-Pro-2 f22_materials % jupyter notebook
```

3. If on Windows and run into issues, see here: https://stackoverflow.com/questions/41034866/running-jupyter-via-command-line-on-windows

How do I edit, save, and compile a notebook?

Break for screensharing example

Variables/objects and types: creating variables

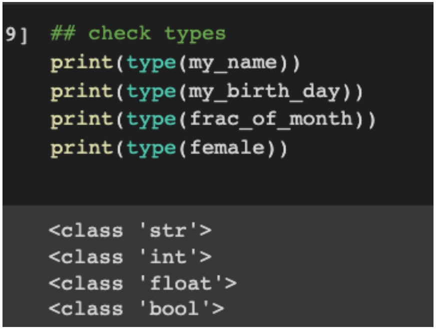
Can follow along here: `https://tinyurl.com/basicsnotebook`

```
my_name = "Rebecca"
my_birth_month = 9
my_birth_day = 19
my_birth_year = 1988
frac_of_month = 19/30
female = True
```

Things to note:

- No spaces or dots in name of object (can use underscores though)
    - Why no dots? In Python, dots are meaningful! they represent the methods and attributes that we'll discuss in later slides today
- = is what's called the *assignment operator*; when you learn `R`, you'll see that you typically use `<-` instead
- I created objects of different `types`; the quotes denote Rebecca is a string; the lack of quotes for birth month, etc. denote integer (or float)

Variables/objects and types: checking types



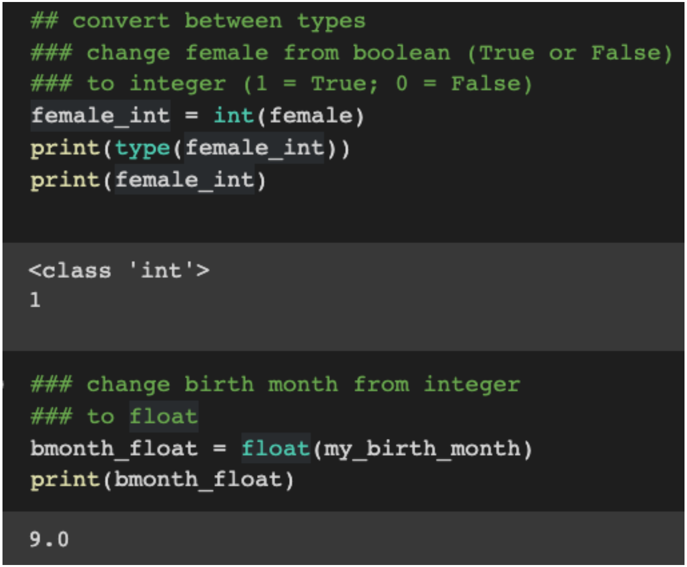
```
9]  ## check types
    print(type(my_name))
    print(type(my_birth_day))
    print(type(frac_of_month))
    print(type(female))
```

```
    <class 'str'>
    <class 'int'>
    <class 'float'>
    <class 'bool'>
```

Things to note:

- **Int versus float**: no decimals versus decimals
- **String versus boolean**: we told Python that it was boolean by setting it equal to `True` without quotes; this is encoded in python as $1 = \text{True}$; $0 = \text{False}$ and we'll see is very useful later for data aggregation/summaries

Variable/objects and types: transforming types



```
## convert between types
### change female from boolean (True or False)
### to integer (1 = True; 0 = False)
female_int = int(female)
print(type(female_int))
print(female_int)
```

```
<class 'int'>
1
```

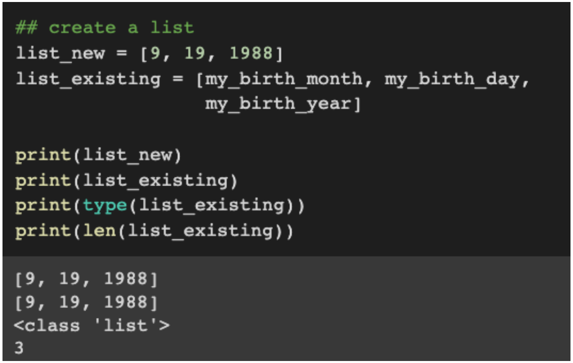
```
### change birth month from integer
### to float
bmonth_float = float(my_birth_month)
print(bmonth_float)
```

```
9.0
```

Where we are

- Course goals
- Intros
- Nut and bolts: course website and specific course components
- **Python basics: lists; basic list comprehension;** numpy arrays

Lists: how to create



```
## create a list
list_new = [9, 19, 1988]
list_existing = [my_birth_month, my_birth_day,
                 my_birth_year]

print(list_new)
print(list_existing)
print(type(list_existing))
print(len(list_existing))
```

```
[9, 19, 1988]
[9, 19, 1988]
<class 'list'>
3
```

Things to note:

- `list_new` I created from scratch; `list_existing` I combined the objects I created earlier in the code
- Either way, use `[` with commas separating list elements
- `len` is a built-in function in Python (doesn't require us to import a package) that works with lists in addition to other types of objects

Other things about lists covered in the DataCamp module due Sunday

- Methods that operate on lists, using syntax: `name_of_list.method()` like round, max, reverse, etc.
- Subsetting lists using the syntax: `name_of_list[0:3]`

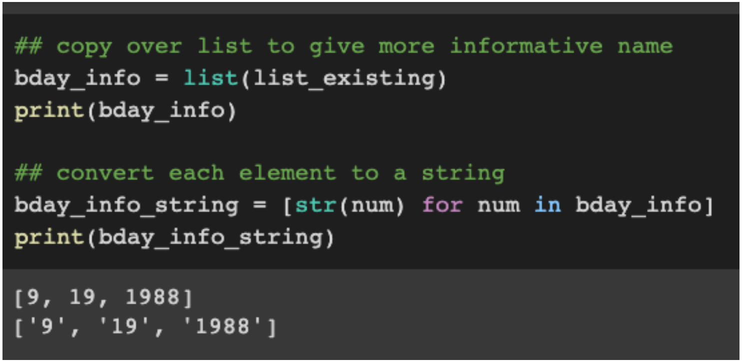
Basic list comprehension

- **Goal:** iterate over list elements and do something:
  - Filter: select a subset of list elements based on some condition
  - Transform: modify the elements of the list
    - General: modifies each element in the same way
    - Conditional: modifies some elements in some way; others in a different way
- In future week, will cover distinctions between using list comprehension for these tasks versus `for loops` (latter used commonly in R and STATA; in Python, you can use list comprehension for almost everything you'd use a `for loop` for and list comprehension has many advantages!)

Example task

Want to convert the list with the three birthday elements—`[9, 19, 1988]`—into a single string: “09-19-1988”

General transformation



```
## copy over list to give more informative name
bday_info = list(list_existing)
print(bday_info)

## convert each element to a string
bday_info_string = [str(num) for num in bday_info]
print(bday_info_string)
```

```
[9, 19, 1988]
['9', '19', '1988']
```

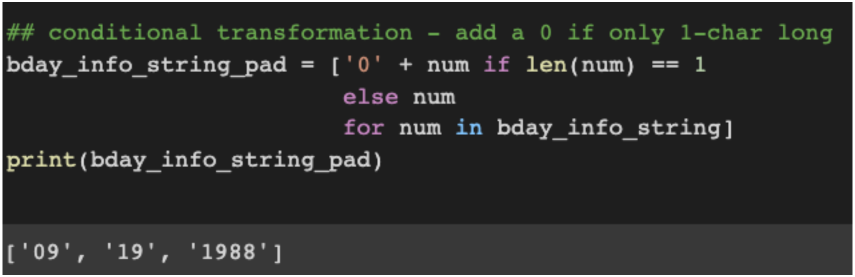
Breaking this down:

- `str(num)` is the step that's doing the transformation
- `for num in bday_info` iterates over each of the three elements in the `bday_info` list
- `num` is a totally arbitrary placeholder; we could use `i`, `el`, or whatever; key is that it's the same between the iteration and transformation

Conditional transformation

What if we want to not just convert each element to string, but add a 0 if the str is one-digit? (So pad the 9 with a 0 as it's converted to a string)?

Conditional transformation



```
## conditional transformation - add a 0 if only 1-char long
bday_info_string_pad = ['0' + num if len(num) == 1
                        else num
                        for num in bday_info_string]
print(bday_info_string_pad)
```

```
['09', '19', '1988']
```

Breaking this down:

- **What stayed the same?** The iterating through elements `for num in bday_info`
- **What changed?** We added a condition using `if` and `else`, and using the built-in `len()` function we covered earlier
  - If it's a 1-character string, it uses the + to paste the string `'0'` onto it
  - Otherwise, it keeps the string as is

Common bugs in conditional transformation: live-coding break

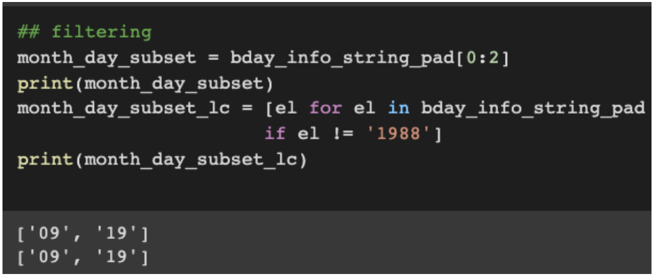
We'll see what happens if:

- We don't put the 0 in quotes when using it to pad the '9'
- We run this command on the original list where the elements are stored as `int` rather than `str`

Filtering using list comprehension

- We can use similar syntax to take a list—in this case, a length-3 list—and filter to a smaller number of elements based on some condition
- This is usually better coding practice than subsetting using the position of an element in a list since its more robust to lists ordered in certain ways

Filtering using indexing versus list comprehension



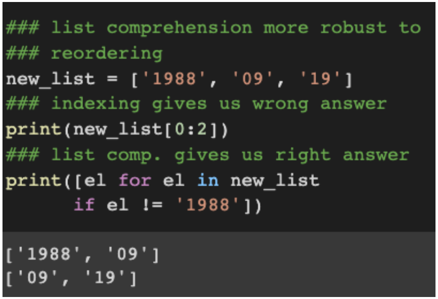
```
## filtering
month_day_subset = bday_info_string_pad[0:2]
print(month_day_subset)
month_day_subset_lc = [el for el in bday_info_string_pad
                      if el != '1988']
print(month_day_subset_lc)
```

```
['09', '19']
['09', '19']
```

Things to note:

- **Filtering using indexing:** relies on elements being in a certain order in the last; inclusive on the left hand side (so includes the first element $\implies$ index 0; second element $\implies$ index 1) but not inclusive on the right hand side
- **Filtering using list comprehension**:
  - Same `for el in list` syntax we saw earlier
  - Returns `el`: just return the list element without any transformations
  - Adding an if condition: keeps the element if it's not equal to 1988 (has drawbacks in generalizability we'll discuss in a few slides)

Robustness to reordering



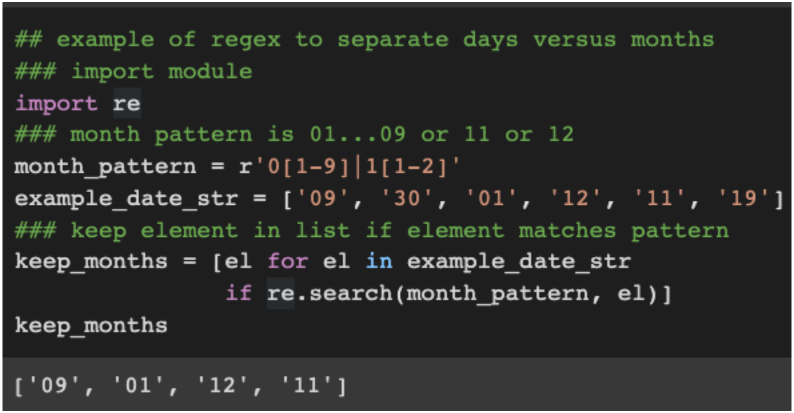
```
### list comprehension more robust to
### reordering
new_list = ['1988', '09', '19']
### indexing gives us wrong answer
print(new_list[0:2])
### list comp. gives us right answer
print([el for el in new_list
      if el != '1988'])
```

```
['1988', '09']
['09', '19']
```

Things to note:

- When the list is reordered, filtering list elements using indexing gives us the wrong answer
- If our pattern is correct, filtering using list comprehension is more robust

Especially powerful when combined with regular expressions (regex) that we'll cover later



```
## example of regex to separate days versus months
### import module
import re
### month pattern is 01...09 or 11 or 12
month_pattern = r'0[1-9]|1[1-2]'
example_date_str = ['09', '30', '01', '12', '11', '19']
### keep element in list if element matches pattern
keep_months = [el for el in example_date_str
               if re.search(month_pattern, el)]
keep_months
```

```
['09', '01', '12', '11']
```

From lists to array: background on python modules

- In previous slide, we used the `import re` statement
- This imports a module named `re` that has functions (in our case: `re.search()`) that are not automatically built into Python
- Common modules we'll use:
    - This week/pset one: `numpy`
    - Next couple weeks/pset two: `pandas` and ones for plotting

Where we are

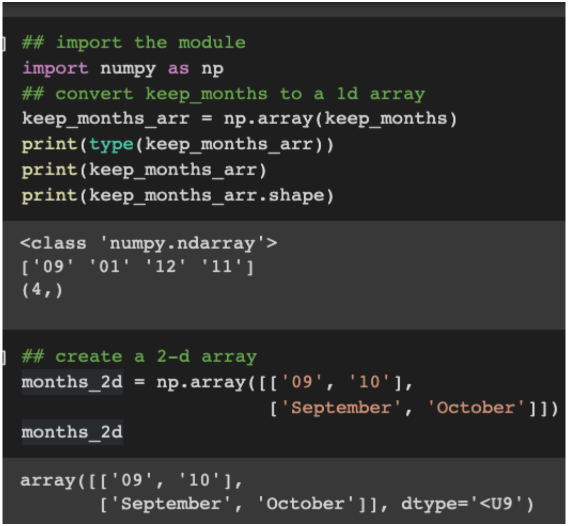
- Course goals
- Intros
- Nut and bolts: course website and specific course components
- **Python basics:** lists; basic list comprehension; **numpy arrays**

Lists versus numpy arrays

Two main (practical) differences:

1. **Lists can store heterogeneous data types; arrays cannot:** a single list can combine a string, integer, etc; an array most hold a single type of data
2. **For various reasons, arrays are more memory efficient for large computational tasks:** uses less memory to *store* the same data (explainer on this); in stats or the ML course you may learn about matrix operations and 2-dimensional numpy arrays are good for these

Numpy arrays: creating



```
## import the module
import numpy as np
## convert keep_months to a 1d array
keep_months_arr = np.array(keep_months)
print(type(keep_months_arr))
print(keep_months_arr)
print(keep_months_arr.shape)
```

```
<class 'numpy.ndarray'>
['09' '01' '12' '11']
(4,)
```

```
## create a 2-d array
months_2d = np.array([['09', '10'],
                      ['September', 'October']])
months_2d
```

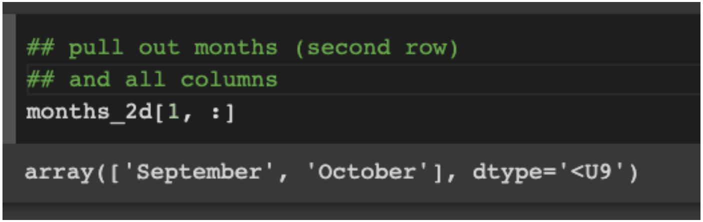
```
array([['09', '10'],
       ['September', 'October']], dtype='<U9')
```

Things to note:

- import numpy as np is called aliasing; we use an abbreviation to refer to the package name (np is arbitrary but commonly used)
- shape is an attribute of arrays; this is different than something like `len()` that's a function
- To create the 2-d array, we use a list of lists

Numpy arrays: filtering/subsetting

Task: how can we pull out the month names (second row; both columns) from the `months_2d` array?



```
## pull out months (second row)
## and all columns
months_2d[1, :]
```

```
array(['September', 'October'], dtype='<U9')
```

Wrapping up

Covered:

- Course goals
- Intros
- Nut and bolts: course website and specific course components
- Python basics: lists; basic list comprehension; numpy arrays

Order of priority:

1. Getting working version of Python installed for first problem set
2. Completing DataCamp assignment due sunday night (if needed)
3. Starting problem set one (short so more doable closer to deadline)