Programming Basics using Jupyter Notebooks

Get set up

Open your terminal, navigate to where you saved the notebook and start jupyter:

```
$ cd where/my/notebook/is
$ jupyter notebook
```

Open browser and go to http://localhost:8888/

Click on Programming basics.ipynb

Get set up / Galaxy

Go to usegalaxy.org & login

Go to:

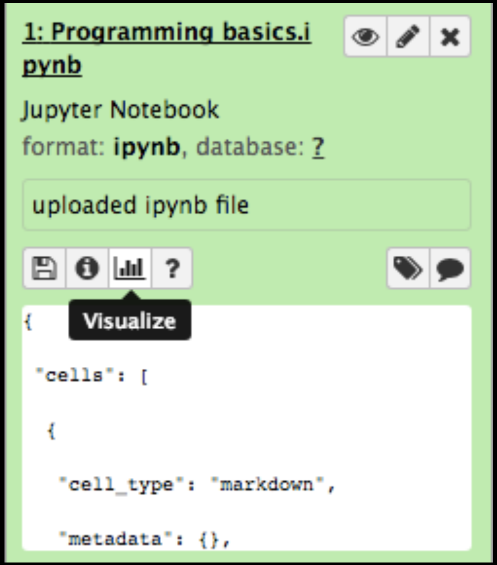
https://usegalaxy.org/u/drcris/h/immport-workshop

Import the history to your account by clicking on ‘Import history’, top right corner.

Click on Programming basics.ipynb

Click on the graph icon

Click on Jupyter


1: Programming basics.ipynb

Jupyter Notebook

format: **ipynb**, database: ?

uploaded ipynb file

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
```

Programming syntax

- Crucial for accurate interpretation
- Depends on programming language

  refer to your favorite language manual for specifics.
- Python: pep8

https://www.python.org/dev/peps/pep-0008/

Programming syntax - Comments

```
# this is a comment
green_worms = []  # this is also a comment
```

- Annotate your code (*please* do)
- Comment out line(s) for troubleshooting or development

Programming syntax - Indentation

```
green_worms = []
for worm in all_worms_on_the_plate:
    if all_worms_on_the_plate[worm]['color'] == 'green':
        green_worms.append(worm)
```

Python:
  4 spaces for one indent – *NOT* tabs

Special characters break I

White spaces are critical, esp. in Python

TAB	- tab key	*(avoid in Python)*
SPACE	- space key	
NEWLINE	- enter/return key	

Programming syntax – End of line

End of line indicator

PERL

```
my @green_worms;
for my $i (0..$#all_worms_on_the_plate) {
    if ($worm_color[$i] eq 'green'){
        push(@green_worms, $all_worms_on_the_plate[$i]);
    }
}
```

(NOT IN PYTHON)

Programming concepts

- Variables
- List of things
- Dictionary of things
- Functions
- Modules
- Objects

Programming concepts

- Variables

```
Reserved memory location
for a given construct
```

Variables - Declaration vs. assignment

Declaration	Reserving memory slot to store value
Assignment	Value of the variable is defined

```python
green_worms = []
```

Python: assignment *is* declaration

Programming concepts - variables

```
worm_species = "C. remanei"
worm_dev_stage = 3
green_worms = []
all_worms_on_plate = {}
dev_stages = ("E", "L1", "L2", "L3", "L4", "A", "D")
```

Python main data types

```
worm_species = "C. remanei"
worm_dev_stage = 3
green_worms = []
all_worms_on_plate = {}
dev_stages = ("E", "L1", "L2", "]
```

String
Number
List
Dictionary
Tuple

```
", "D")
```

Programming concepts

- Variables
- List of things
- Dictionary of things
- Functions
- Modules
- Objects

Programming concepts

- Variables
- List of things

Indexed list of constructs

Python: Lists and Tuples

Programming concepts - lists

```
empty_list = []
empty_tuple = ()
prime_numbers = [1, 3, 5, 7, 9, 11, 13, 17]
primary_colors = ("red", "green", "blue")
list_of_variables = [var1, var2, var3]

prime_numbers[2]
>> 5

primary_colors[0]
>> 'red'
```

Indexing break

Counting starts at 0 *(except in R)*

```
primary_colors = ("red", "green", "blue")
          Index       0        1        2
```

Programming concepts - lists

```
list_of_lists = [[1,2], [3,4], [5,6], [7,8]]
misc_list = [1, [2,3], {4:5, 6:7}, (8,[9,10])]
misc_tuple = (1, [2,3])

list_of_lists[2][0]
>> 5


misc_list[2][4]
>> 5
```

Programming concepts

- Variables
- List of things
- Dictionary of things
- Functions
- Modules
- Objects

Programming concepts

- Variables
- List of things
- Dictionary of things

Unordered list of key:value pairs

Python: Dictionaries

Programming concepts - dictionaries

SNP position	SNP
1013340	A
1298347	C
2348893	A
2458789	-
2798876	G

Semantics breaks

SNP position	SNP
1013340	A
1298347	C

KEYS

VALUES

```
snp_per_position = {
                      1013340 : "A",
                      1298347 : "C",
                      2348893 : "A"
               }
```

Programming concepts – dictionaries

```
empty_dict = {}
snp_per_position = {
      1013340 : "A",
      1298347 : "C",
      2348893 : "A"
}

snp_per_position[1298347]
>> 'C'
```

NO index,
UNIQUE keys

Programming concepts – dictionaries

```
dict_of_lists = {
      "wormA" : ["dpy","unc","fem"],
      "wormB" : ["gfp","dpy"],
      "wormC" : ["dpy","unc"]
}


dict_of_lists["wormB"][0]
>> 'gfp'
```

Programming concepts – dictionaries

Dataframes

Worm ID	Worm Sex	Developmental Stage	GFP
1	ND	L1	no
2	ND	L2	yes
3	Male	Adult	no
4	Female	L4	no
5	Female	Late L4	yes

Programming concepts – dictionaries

```
dict_of_dicts = {
        "worm1": {
                "sex" : "ND",
                "dev_stage" : 1,
                "GFP" : False
        },
        "worm2": {
                "sex" : "ND",
                "dev_stage" : 2,
                "GFP" : True
        }
}
```

Programming concepts – dictionaries

```
dict_of_dicts = {
        "worm1": {
                "sex" : "ND",
                "dev_stage" : 1,
                "GFP" : False,
                "conditions" : ("20C", "OP50")
        },
        "worm2": {
                "sex" : "ND",
                "dev_stage" : 2,
                "GFP" : True,
                "conditions" : ("20C", "OP50")
        }
}
```

Best practice break

Reserved words:

Words that already mean something in programming language speak and cannot be used as variables

Don’t call your list ‘`list`’ or your dictionary ‘`dict`’

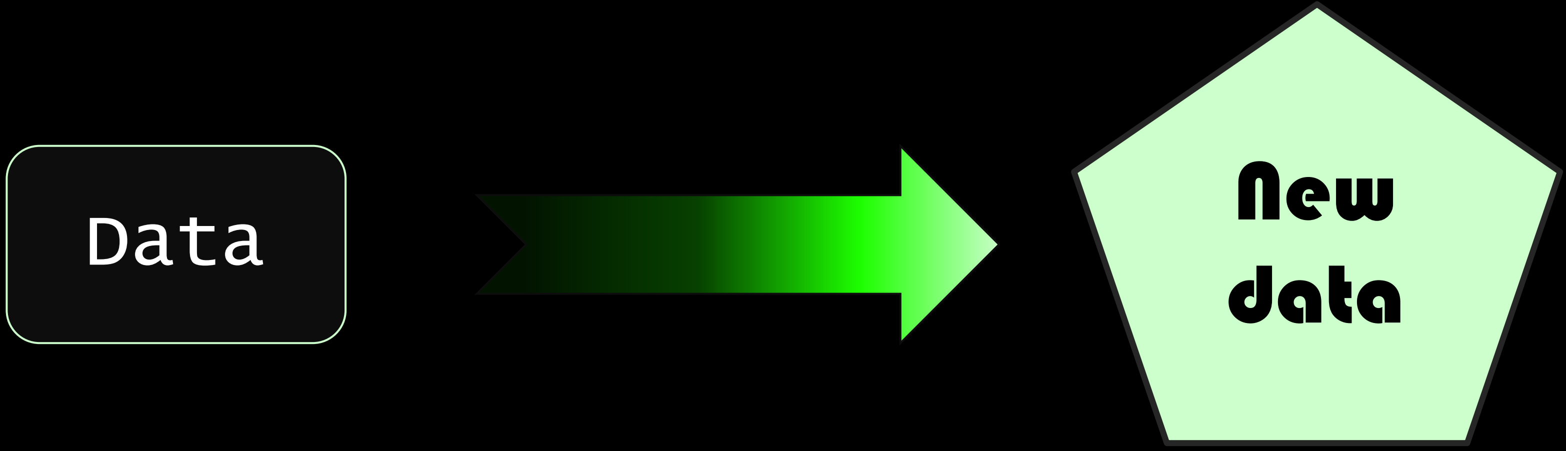
https://docs.python.org/3/reference/lexical_analysis.html

Programming concepts

- Variables
- List of things
- Dictionary of things
- Functions
- Modules
- Objects

Programming concepts

- Functions



Programming concepts - functions

Built-in functions

```
worm_species = "C. remanei"

print(worm_species)
>> C. remanei
```

Python list of built-in functions:
https://docs.python.org/3/library/functions.html

Programming concepts - functions

User-defined functions

```
def new_function(data):
      # do something to data
      # and call it new_data
      new_data = data + 1
      return new_data

my_data = 3
my_new_data = new_function(my_data)
my_new_data
>> 4
```

Variable scope

```
a = 0

def do_stuff(b):
    c = b + 3
    return c

print(a)
print(b)
print(c)
```

http://python-textbok.readthedocs.io/en/1.0/Variables_and_Scope.html#variable-scope-and-lifetime

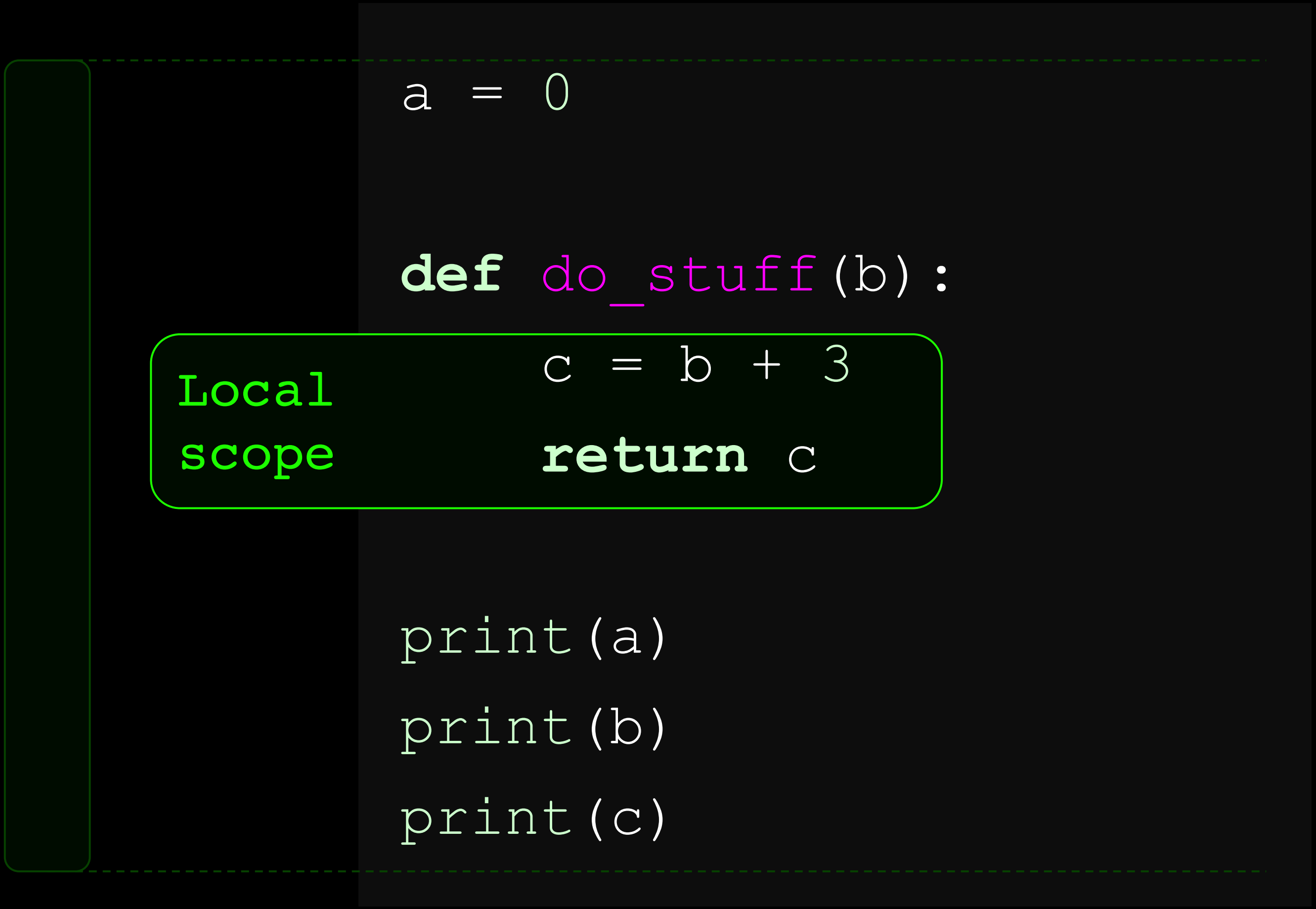
Variable scope

Global scope

```
a = 0

def do_stuff(b):
    c = b + 3
    return c

print(a)
print(b)
print(c)
```

http://python-textbok.readthedocs.io/en/1.0/Variables_and_Scope.html#variable-scope-and-lifetime

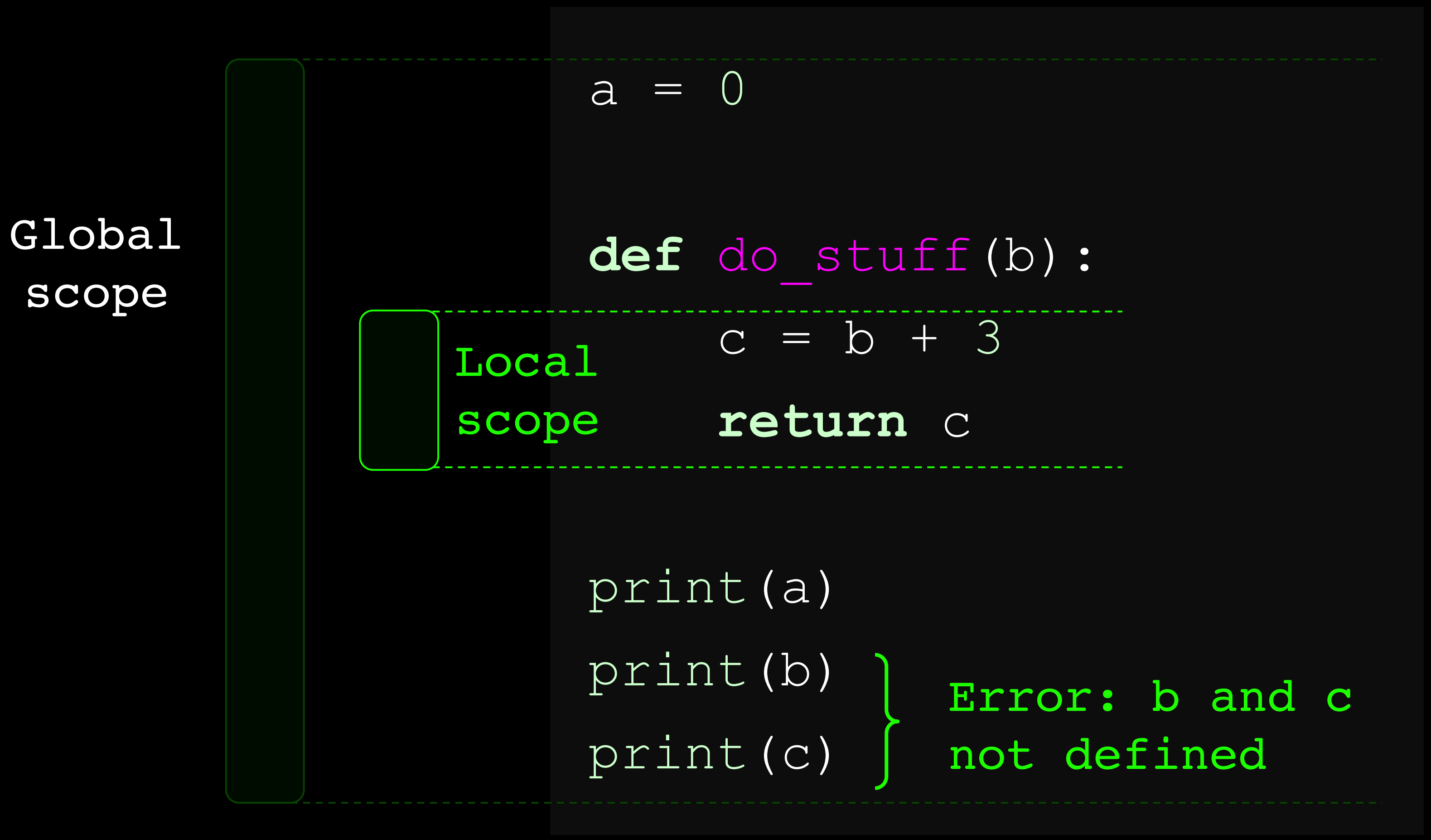
Variable scope



```
a = 0

def do_stuff(b):
    c = b + 3
    return c

print(a)
print(b)
print(c)
```

http://python-textbok.readthedocs.io/en/1.0/Variables_and_Scope.html#variable-scope-and-lifetime

Variable scope



```
a = 0


def do_stuff(b):
    c = b + 3
    return c


print(a)
print(b)
print(c)
```

http://python-textbok.readthedocs.io/en/1.0/Variables_and_Scope.html#variable-scope-and-lifetime

Programming concepts

- Variables
- List of things
- Dictionary of things
- Functions
- Modules
- Objects

Programming concepts

- Modules / Libraries / Packages

File containing code that you can use and re-use

Programming concepts - modules

- Existing modules or packages?
- Make yours re-usable by yourself or others

Programming concepts - modules

```
# tell your computer which library you want
import package_name

# use a function from the library
package_name.some_function()

# other valid import statements:
import package_name as pn
pn.some_function()

from package_name import some_function
some_function()
```

Programming concepts - modules

```
# avoid this:
from package_name import *
```

$\downarrow$

everything

Programming concepts

- Variables
- List of things
- Dictionary of things
- Functions
- Modules / Libraries
- Objects

Programming concepts

- Classes >> Objects

Programming concepts - classes

Blueprint

Programming concepts - classes

Blueprint

Programming concepts - classes

Blueprint -- States

Worms

- **Common name:** Nematodes
- **Main studied species:** *C. elegans*
- **Sister species:** *C. remanei, C. briggsae,*
  C.brenneri, C. japonica
- **Larval stages:** 4
- **Mating system:** Androdioceous
- **Chromosomes:** 5 + X

Programming concepts - classes

Blueprint -- Behaviors

Worms

- **Common name:** Nematodes
- **Main studied species:** *C. elegans*
- **Sister species:** *C. remanei, C. briggsae,*
  C.brenneri, C. japonica
- **Larval stages:** 4
- **Mating system:** Androdioceous
- **Chromosomes:** 5 + X

- Crawl around
- Eat
- Mate
- Dauer formation

Programming concepts - classes

Blueprint

Worms

- **Common name:** Nematodes
- **Main studied species:** *C. elegans*
- **Sister species:** *C. remanei*, *C. briggsae*, *C.brenneri*, *C. japonica*
- **Larval stages:** 4
- **Mating system:** Androdioceous
- **Chromosomes:** 5 + X

Attributes

- Crawl around
- Eat
- Mate
- Dauer formation

Methods

Programming concepts – classes & objects

An object is an
instance of a class

Programming concepts – classes & objects

Every Worm:

- **Common name:** Nematodes
- **Main studied species:** *C. elegans*
- **Sister species:** *C. remanei, C. briggsae, C.brenneri, C. japonica*
- **Larval stages:** 4
- **Mating system:** Androdioceous
- **Chromosomes:** 5 + X

Attributes

- Crawl around
- Eat
- Mate
- Dauer formation

Methods

Programming concepts - objects

```
class Worm:
    def __init__(self, name):
        self.species_name = name
        self.sister_sp = []
        self.state = "hungry"

    def eat(self, food):
        if food == 'OP-50':
            self.state = "happy”
        else:
            self.state = "grumpy"

    def enter_dauer(self):
        self.state = "staaaaaarving"

    def assessment(self):
        print("I’m " + self.state)
```

Programming concepts - objects

```
class Worm:
    def __init__(self, name):
        self.species_name = name
        self.sister_sp = []
        self.state = "hungry"

    def eat(self, food):
        if food == 'OP-50':
            self.state = "happy”
        else:
            self.state = "grumpy"

    def enter_dauer(self):
        self.state = "staaaaaarving"

    def assessment(self):
        print("I’m " + self.state)
```

Attributes

Methods

Programming concepts - objects

```
harry = Worm("C. elegans")

print(harry.species_name)
>> C. elegans

harry.assessment()
>> I’m hungry

harry.eat("salad")
harry.assessment()
>> I’m grumpy

harry.enter_dauer()
harry.assessment()
>> I’m staaaaaarving
```

Programming concepts - objects

```
harry.eat("OP-50")
harry.assessment()
>> I’m happy

print(harry.sister_sp)
>> []

harry.sister_sp.append("C. briggsae")
print(harry.sister_sp)
>> ['C. briggsae']

sally = Worm("C. elegans")
print(sally.sister_sp)
>> []
```

Programming concepts – Python & objects

Everything is an object

Python main data types

```
worm_species = "C. remanei"
worm_dev_stage = 3
green_worms = []
all_worms_on_plate = {}
dev_stages = ("E", "L1", "L2", "I
```

String

Number

List

Dictionary

Tuple

```
", "D")
```

Standard Data Analysis

1. Open File
2. Look at data
3. Curate data
4. Make pretty graphs
5. Publish cool paper
6. Take over the world

Standard Data Analysis

1. Open File
2. Look at data
3. Curate data
4. Make pretty graphs
5. Publish cool paper
6. Take over the world

I/O operations

1. Double-click on file
2. Open in excel
3. Create new data in (new) file
4. Save file
5. Close file

1. Give path to file
2. Read in file
3. Create new data in memory
4. Write to (new) file
5. Close file

I/O operations

1. Double-click on file	1. Give path to file
2. Open in excel	2. Read in file
3. Create new data in (new) file	3. Create new data in memory
4. Save file	4. Write to (new) file
5. Close file	5. Close file

I/O operations

For this tutorial, let’s look at an ImmPort file.

Locate the folder in which you saved the material for this course.
We’ll use the file called *subject.txt*

I/O operations

```
my_file = 'path/to/my/file/subject.txt'
```

Or:

```
my_file = input('File to open for analysis: ')
```

Reads input and returns a string

Pro: no hard-coding

Con: need to type your file path every time

Best practice break

Hard coding:

Using a value instead of a variable

Avoid this:

```
f = open('path/to/my/file/subject.txt', 'r')
```

Instead, do this:

```
my_file = 'path/to/my/file/subject.txt'


f = open(my_file, 'r')
```

Good discussion about this here:
http://softwareengineering.stackexchange.com/questions/67982/is-it-ever-a-good-idea-to-hardcode-values-into-our-applications

I/O operations

```python
my_file = 'path/to/my/file/subject.txt'

f = open(my_file, 'r')
```

Path to file

w	write
r	read (default)
r+	read + write
a	append

File object

Assumption

```
my_file = 'path/to/my/file/subject.txt'

f = open(my_file, 'r')
```

$\downarrow$

```
This is a
text file
```

I/O operations

```
my_file = 'path/to/my/file/subject.txt'


f = open(my_file, 'r')
```

Do stuff

```
f.close()
```

I/O operations

Better practice:

```
my_file = 'path/to/my/file/subject.txt'


with open(my_file, 'r') as f:
```

Do stuff

Closes the file handle after block

Standard Data Analysis

1. Open File
2. Look at data
3. Curate data
4. Make pretty graphs
5. Publish cool paper
6. Take over the world

I/O operations

```
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline()
```

Reads file up until first NEWLINE

First line in file as a string INCLUDING newline character

Useful tidbits - Strings

```
line.strip([chars])
```

$\Longrightarrow$ String

Removes leading and trailing characters specified by *chars* in *line* (default white space)

```
my_line = " a simple example "
my_line.strip()
>> 'a simple example'
```

Special characters break II

Some non-alphanumeric characters need to be coded or 'escaped':

```
TAB     - tab key          - '\t'
SPACE   - space key        - ' '
NEWLINE - enter/return key - '\n'
```

I/O operations

```
my_file = 'path/to/my/file/subject.txt'


with open(my_file, 'r') as f:
    first_line = f.readline().strip()
```

First line in file as a string WITHOUT newline character

Useful tidbits - Strings

line.split(*sep*) $\Rightarrow$ List

Splits the given *line* at every *sep*
(default white space)

```
my_line = " a simple example "
my_line.split()
>> ['a', 'simple', 'example']
```

I/O operations

```
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")
```

$\xrightarrow{}$ List of file’s headings

File content

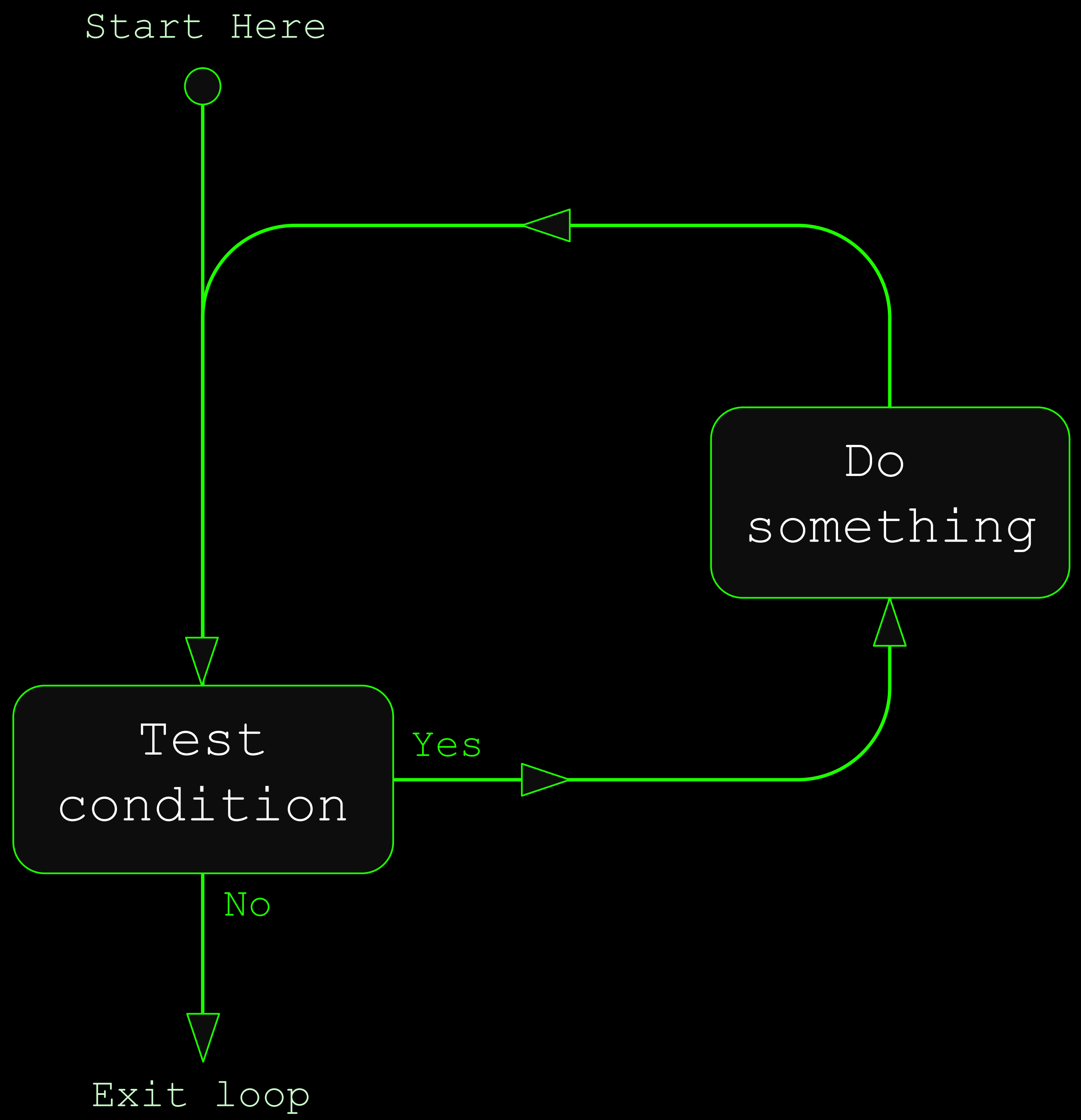
```
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")
    second_line = f.readline().strip().split("\t")
    # do something
```

File content

```
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")
    second_line = f.readline().strip().split("\t")
    # do something
```

Repeat this
for all lines

Loops



Loops in Python

```
for ITEMS in LIST:
    # do something
```

```
while condition:
    # do something
```

https://docs.python.org/3/reference/compound_stmts.html

Loops in Python

```
for ITEMS in LIST:
    # do something

for i in range(0, 10):
    print(i)

for letters in 'bananas':
    print(letters)

for colors in primary_colors:
    print(colors)
```

```
while condition:
    # do something
```

Loops in Python

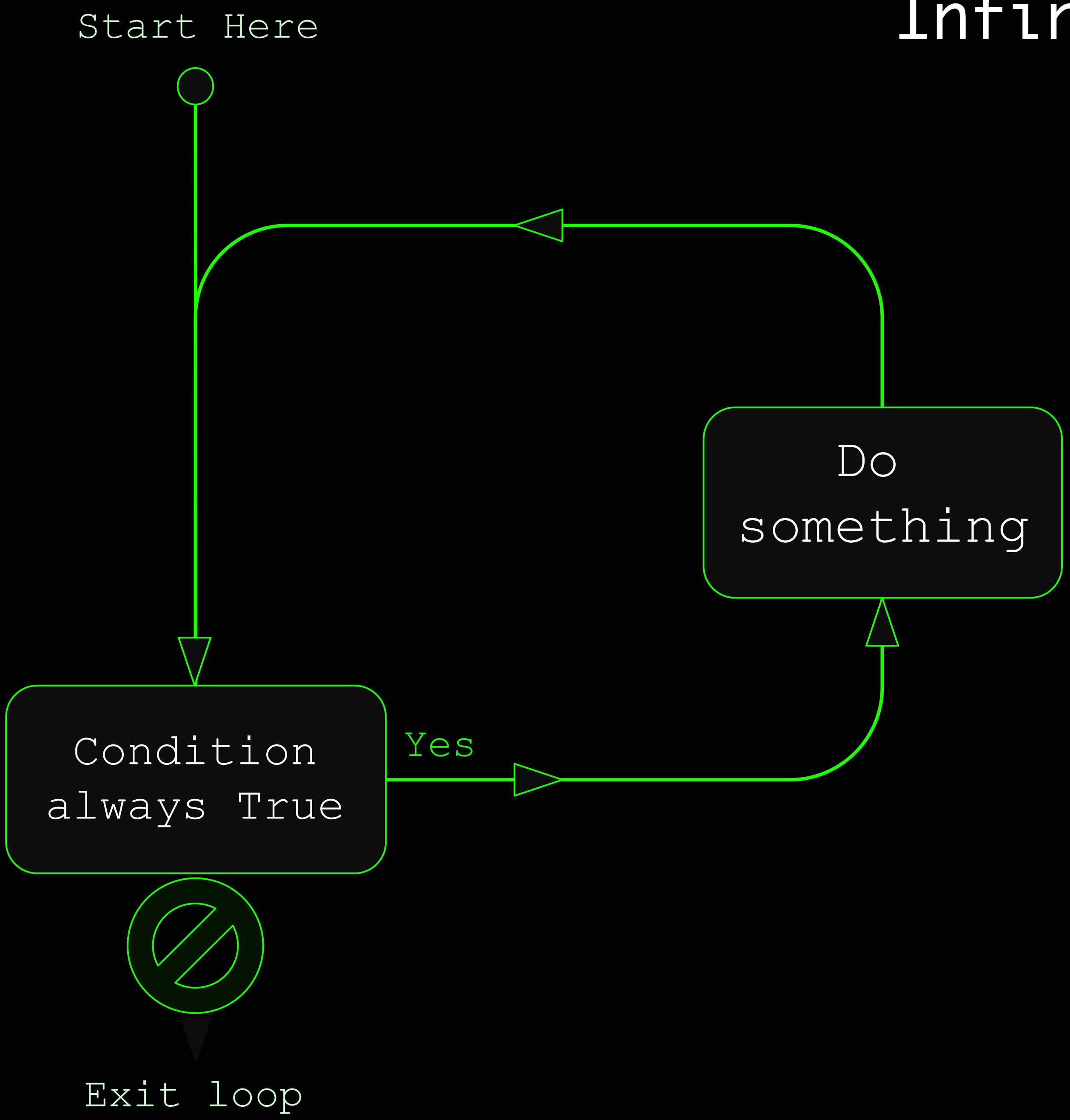
```
for i in range(0, len(some_list)):
    print(i)

for index, element in enumerate(primary_colors):
    print(index, element)

for words in some_dictionary:
    print(words + " " + some_dictionary[words])
```

Loops in Python

```
for ITEMS in LIST:
    # do something
```

```
while condition:
    # do something

i = 0
while i < 10:
    print(i)
```

Infinite Loops



Loops in Python

```
for ITEMS in LIST:
    # do something
```

```
while condition:
    # do something

i = 0
while i < 10:
    print(i)
    i += 1
```

File content

```python
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    for line in f:
        line_content = line.strip().split("\t")
```

File content

```
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, print race
```

Conditional statements

```
if condition:
    # do something
elif condition:
    # do something
elif condition:
    # do something
else:
    # do something
```

Conditional statements

```python
if i == 0:
    print("i is null")
elif i == 1:
    print("i is 1")
elif i > 1:
    print("i is positive")
else:
    print("i is negative. Or between 0 and 1")
```

Conditional statements

```
if carrots:
    print("let’s make a salad")
else:
    print("where is the rabbit?")
```

Useful tidbits - Boolean

Will evaluate as:

true	false
True	**False**
	None
`1, 1.3, 4J`	`0, 0.0, 0j`
`'hi', [1,2], (2,3)`	`'', [], ()`
`{fr: "non", en: "no"}`	`{}`

Conditional statements

```
color = "red"
if color in primary_color:
    print(color + " is a primary color")


if color not in primary_color:
    print(color + " is not a primary color")
```

File content

```python
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, print race:
        if line_content[4] == 'Male':
            print(line_content[5])
```

String matching

```
str.startswith(prefix)
str.endswith(suffix)
```

$\Longrightarrow$ True/False

Checks if *str* starts or ends with *prefix*/*suffix*

```
new_line1 = 'a simple example'
if new_line1.startswith('a'):
    print('yay!')
>> yay!
```

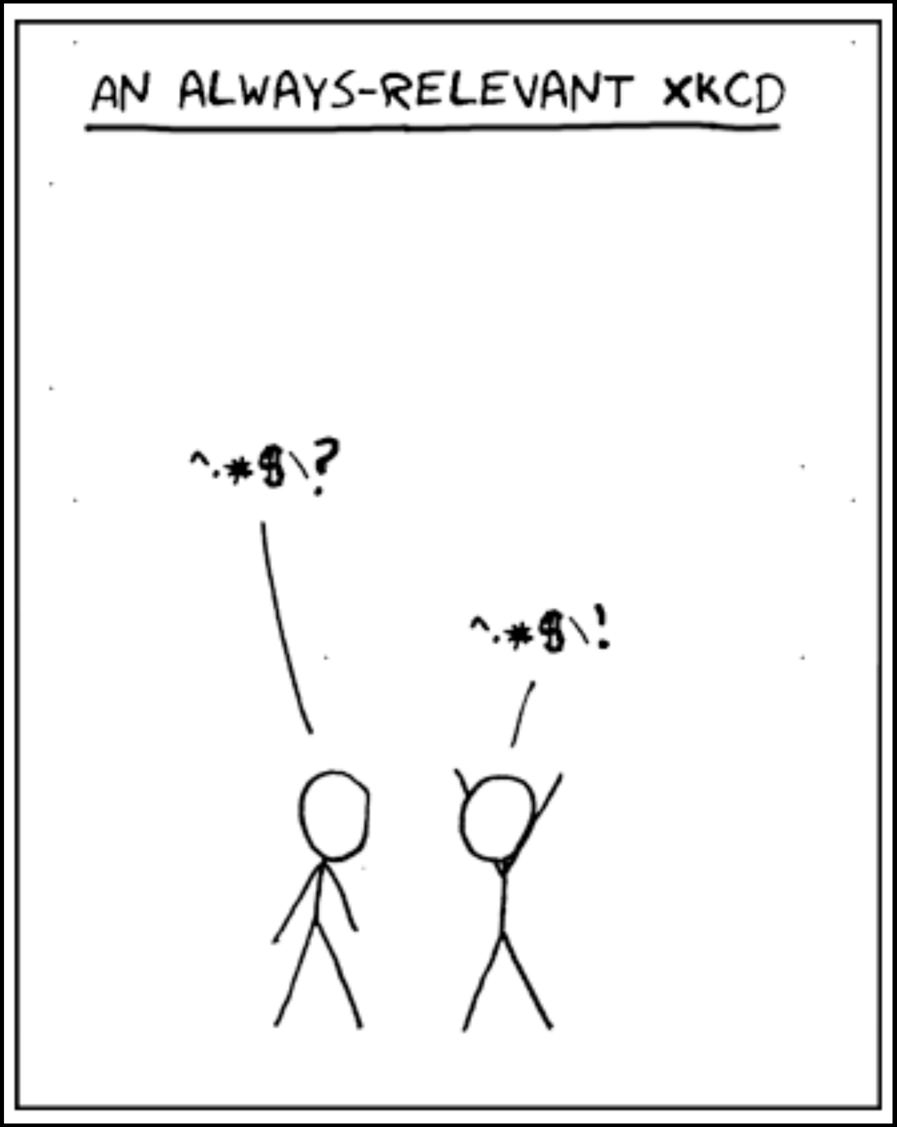
For more complex string matching:

re module:
  https://docs.python.org/3/library/re.html

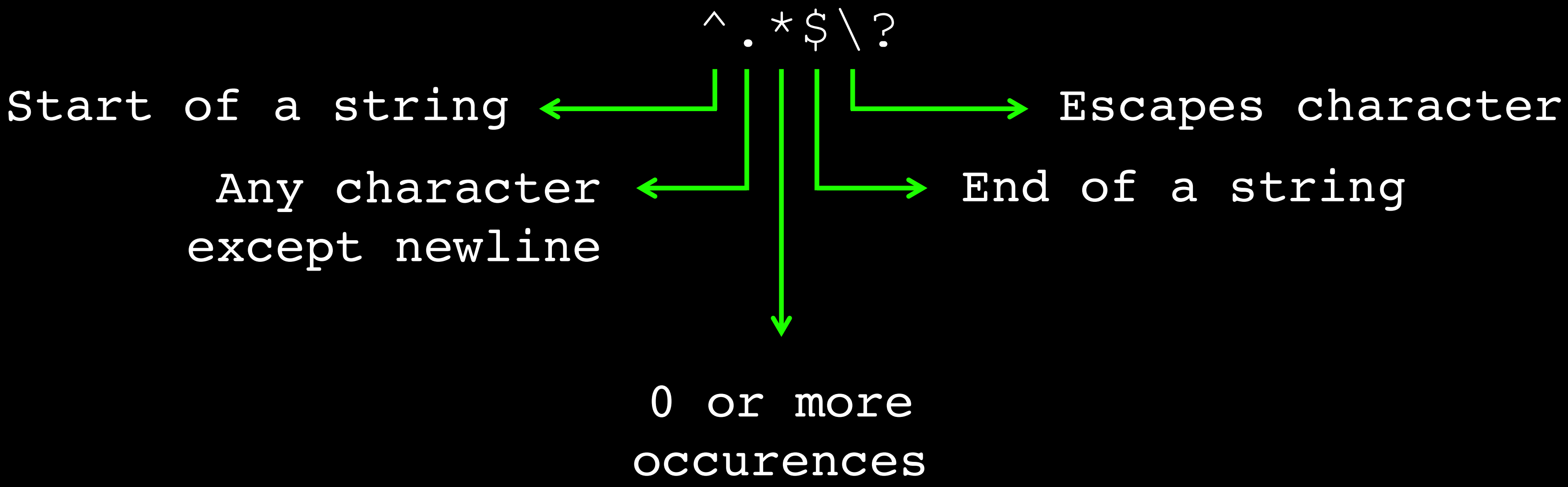
Regular expression – matching a pattern

Sequence of characters that defines a pattern

Regular expression – matching a pattern



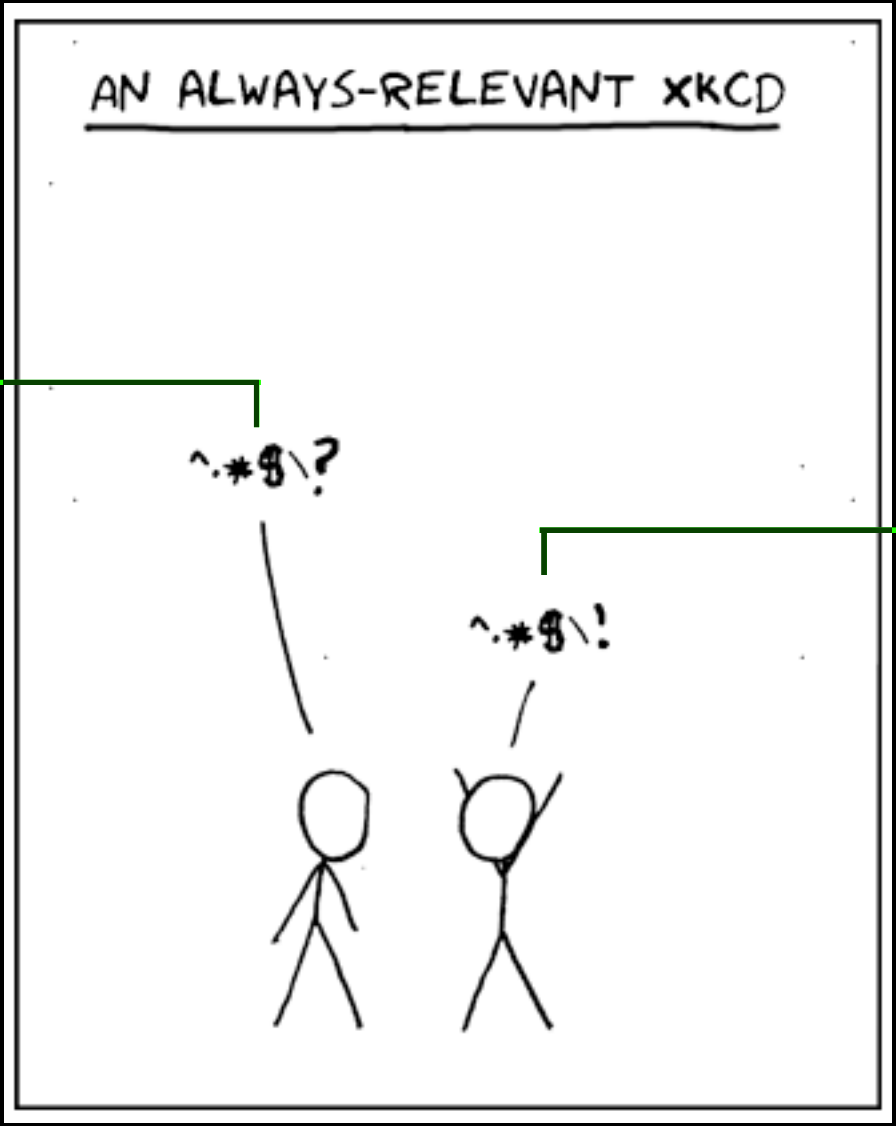
Regular expression – matching a pattern



https://www.debuggex.com/cheatsheet/regex/python
http://www.pyregex.com/

Regular expression – matching a pattern

Any question?



Any answer!

File content

```python
my_file = 'path/to/my/file/subject.txt'

with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, print race:
        if line_content[4].startswith('M'):
            print(line_content[5])
```

File content

```
with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    male_div = []
    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, remember race:
        if line_content[4].startswith('M'):
            male_div.append(line_content[5])
```

Add element to list

File content

```
with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    male_div = []
    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, remember race:
        if line_content[4].startswith('M'):
            male_div.append(line_content[5])
```

File content

```python
with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    male_div = []
    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, remember race:
        if line_content[4].startswith('M'):
            male_div.append(line_content[5])
```

File content

```
with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    male_div = []
    sex_ratio = {'M':0, 'F':0}
    for line in f:
        line_content = line.strip().split("\t")
        # if subject is male, remember race:
        if line_content[4].startswith('M'):
            male_div.append(line_content[5])
            sex_ratio['M'] += 1
```

File content

```
with open(my_file, 'r') as f:
    first_line = f.readline().strip()
    headings = first_line.split("\t")

    male_div = []
    sex_ratio = {'M':0, 'F':0}
    for line in f:
        ...
```

Get data back?

I/O operations

```
with open(output_file, 'w') as outf:
    outf.write('stuff')
```

Opens file for writing.

If the file already exists, erases it

Writes to file

Special characters break III

White space are not included automagically.

```
TAB     - tab key          - '\t'
SPACE   - space key        - ' '
NEWLINE - enter/return key - '\n'
```

I/O operations

```
with open(output_file, 'w') as outf:
    outf.write('stuff\n')
```

$\downarrow$

Adds a newline character

I/O operations

```
with open(output_file, 'w') as outf:
    outf.write('GENDER\tDIVERSITY\tCOUNT\n')
```

I/O operations

```python
with open(output_file, 'w') as outf:
    outf.write('GENDER\tDIVERSITY\tCOUNT\n')
    for gender in sex_ratio:
        line_to_write = gender + "\t" + \
                        div[gender] + "\t" + \
                        sex_ratio[gender] + "\n"
```

Useful tidbits - Strings

sep.join(*list*) $\Rightarrow$ String

Creates a single string from *list* elements joined by *sep*

```
new_line2 = ['a', 'simple', 'example']
"-".join(new_line2)
>> 'a-simple-example'
```

I/O operations

```
with open(output_file, 'w') as outf:
    outf.write('GENDER\tDIVERSITY\tCOUNT\n')
    for gender in sex_ratio:
        line_to_write = "\t".join([
                                  gender,
                                  div[gender],
                                  sex_ratio[gender]
                              ])
```

Useful tidbits – Unique set

```
set(list)
```

$\Rightarrow$ List

Gets unique set of elements in *list*

```
rep_list = ['a', 'b', 'c', 'a', 'c', 'b']
unique_list = set(rep_list)
print(unique_list)
>> ['a', 'b', 'c']
```

I/O operations

```python
with open(output_file, 'w') as outf:
    outf.write('GENDER\tDIVERSITY\tCOUNT\n')
    for gender in sex_ratio:
        diversity = set(div[gender])
        line_to_write = "\t".join([
                                    gender,
                                    diversity,
                                    sex_ratio[gender]
                               ])
```

Useful tidbits – Type casting

`str(object)` $\Rightarrow$ String

Changes *object* to string

```
not_a_string = 3
str(not_a_string)
>> '3'
```

Useful tidbits – Type casting

```
an_integer = 2
a_float = 2.0

int(3.8)
>> 3

float(5)
>> 5.0
```

Pro-tip: be aware that with Python 2.x, 1/2 = 0

I/O operations

```
with open(output_file, 'w') as outf:
    outf.write('GENDER\tDIVERSITY\tCOUNT\n')
    for gender in sex_ratio:
        diversity = set(div[gender])
        line_to_write = "\t".join([
                                  gender,
                                  str(diversity),
                                  str(sex_ratio[gender])
                              ])
        outf.write(line_to_write + "\n")
```

Reality check break

There are many many valid ways to do things

Standard Data Analysis

1. Open File
2. Look at data *the easier way*
3. Curate data
4. Make pretty graphs
5. Publish cool paper
6. Take over the world

Exploratory Data Analysis

```python
import pandas as pd
my_file = 'path/to/my/file/subject.txt'
df = pd.read_table(my_file)
```

→ Reads in a general delimited file

→ DataFrame object (eq. to R's dataframes)

Standard Data Analysis

1. Open File
2. Look at data
3. Curate data
4. Make pretty graphs
5. Publish cool paper
6. Take over the world

Pre-analysis Data Curation

- Familiarize yourself with the file structure:
  How many columns, how are they separated, headings or not, strings or numbers…

- Expect inconsistent data:
  Empty columns / Nas / 0s, inconsistent data types, missing columns, ‘hi’ vs. ‘Hi’, Excel misconception of dates…

- Strategize your code to minimize I/O operations

Standard Data Analysis

1. Open File
2. Look at data
3. Curate data
4. Make pretty graphs
5. Publish cool paper
6. Take over the world

EDA - Resources

Coursera class R. Peng, J. Leek and B. Caffo (**in R**)
https://www.coursera.org/learn/exploratory-data-analysis

Analyzing and Manipulating Data with Pandas by J. Rocher
https://www.youtube.com/watch?v=0CFFTJUZ2dc

Awesome Data Science. 2.0 Introduction to Pandas and Exploratory Data Analysis
https://www.youtube.com/watch?v=ZrRpN_IrcBA

More advanced Jupyter / Pandas tutorial by J. Vanderplas
https://www.youtube.com/playlist?list=PLYCpMb24GpOC704uO9svUrihl-HY1tTJJ

Going back to functions

If you are going to be using this snippet of code you just wrote a lot, make it into a function.

Modules / libraries

If this code you just wrote would be useful to more than just you, make it into a library.

Consider sharing it with the community.

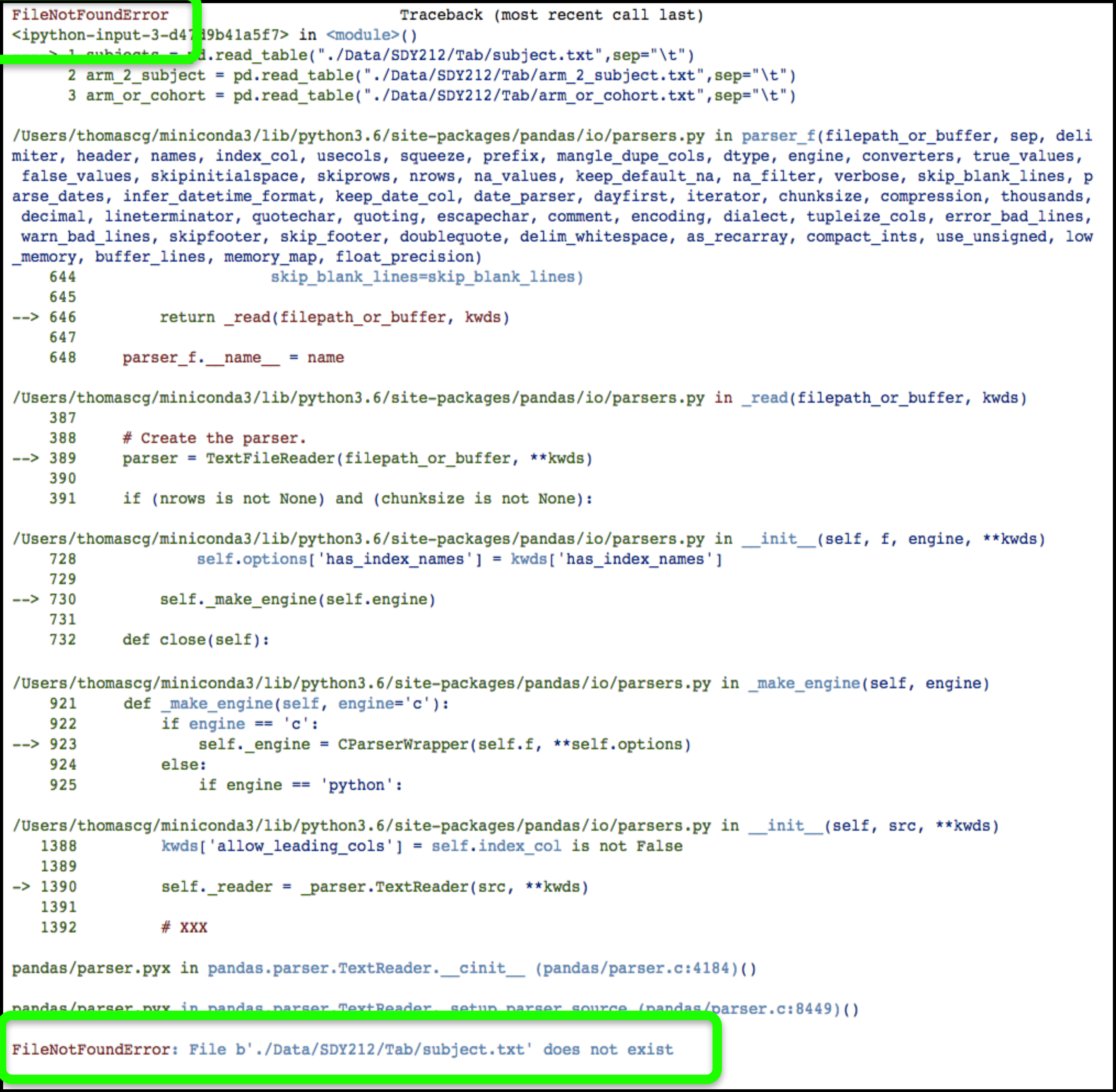
Consider making it a Galaxy tool.

Troubleshooting

```
print p[k]

  File "<ipython-input-31-25a740847cb6>", line 1
    print p[k]
          ^
SyntaxError: Missing parentheses in call to 'print'
```

Troubleshooting

```
print(p[k])

NameError
<ipython-input-32-cfed065d0920> in <module>()
----> 1 print(p[k])

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



```
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-3-d47d9b41a5f7> in <module>()
----> 1 subjects = pd.read_table("./Data/SDY212/Tab/subject.txt",sep="\t")
      2 arm_2_subject = pd.read_table("./Data/SDY212/Tab/arm_2_subject.txt",sep="\t")
      3 arm_or_cohort = pd.read_table("./Data/SDY212/Tab/arm_or_cohort.txt",sep="\t")

/Users/thomascg/miniconda3/lib/python3.6/site-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep, deli
miter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values,
 false_values, skipinitialspace, skiprows, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, p
arse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, iterator, chunksize, compression, thousands,
 decimal, lineterminator, quotechar, quoting, escapechar, comment, encoding, dialect, tupleize_cols, error_bad_lines,
 warn_bad_lines, skipfooter, skip_footer, doublequote, delim_whitespace, as_recarray, compact_ints, use_unsigned, low
_memory, buffer_lines, memory_map, float_precision)
    644                     skip_blank_lines=skip_blank_lines)
    645 
--> 646         return _read(filepath_or_buffer, kwds)
    647 
    648     parser_f.__name__ = name

/Users/thomascg/miniconda3/lib/python3.6/site-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    387 
    388     # Create the parser.
--> 389     parser = TextFileReader(filepath_or_buffer, **kwds)
    390 
    391     if (nrows is not None) and (chunksize is not None):

/Users/thomascg/miniconda3/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    728             self.options['has_index_names'] = kwds['has_index_names']
    729 
--> 730         self._make_engine(self.engine)
    731 
    732     def close(self):

/Users/thomascg/miniconda3/lib/python3.6/site-packages/pandas/io/parsers.py in _make_engine(self, engine)
    921     def _make_engine(self, engine='c'):
    922         if engine == 'c':
--> 923             self._engine = CParserWrapper(self.f, **self.options)
    924         else:
    925             if engine == 'python':

/Users/thomascg/miniconda3/lib/python3.6/site-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1388         kwds['allow_leading_cols'] = self.index_col is not False
   1389 
-> 1390         self._reader = _parser.TextReader(src, **kwds)
   1391 
   1392         # XXX

pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:4184)()

pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:8449)()

FileNotFoundError: File b'./Data/SDY212/Tab/subject.txt' does not exist
```

Troubleshooting

- Read the traceback to see where the error comes from
- Google
- Stack overflow

http://www.blog.pythonlibrary.org/2012/09/12/python-101-exception-handling/
https://doughellmann.com/blog/2009/06/19/python-exception-handling-techniques/

Any more questions?

Immport.org

Material and more tutorials:

immport.org/resources/tutorials

immport@immport.org

Cristel.Thomas@nih.gov

Bisc NG YouTube

@ImmPortDB

@crstlthms