Standard Libraries

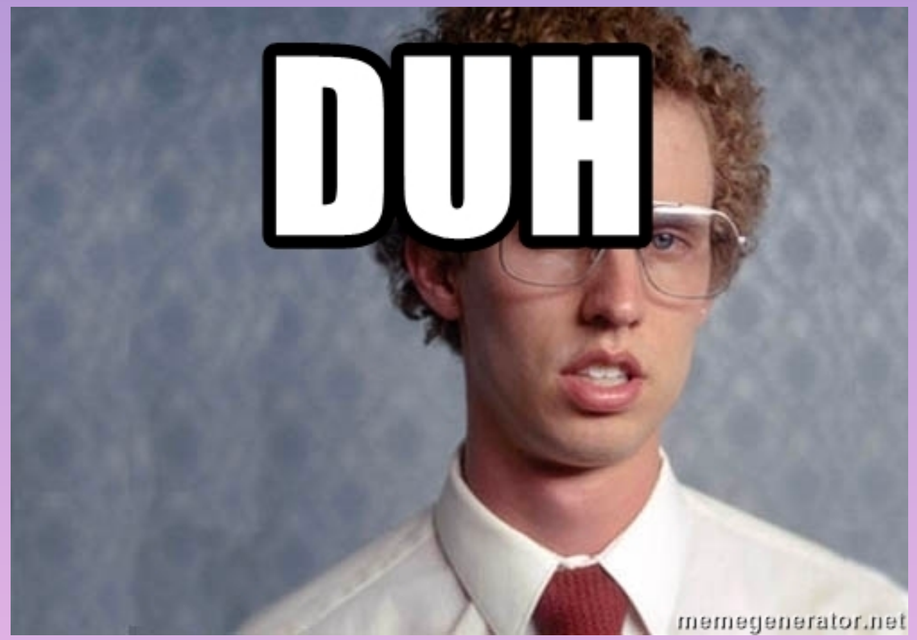
May 23th

Final Project

- Due in 14 days
- Feel Free to meet with us anytime in next few weeks

But first….

A little more functional programming


DUH
memegenerator.net

Lambda: Anonymous Inline Functions

```
lambda params : expression
```

Let’s see how these are used

```
def square_add_two(x,y):
    return  x**2 + y**2
```

Let’s see how these are used

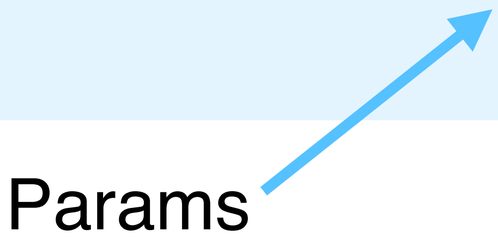
```
def square_add_two(x,y):
    return  x**2 + y**2
```

```
lambda x, y : x**2 + y**2
```

Let’s see how these are used

```
def square_add_two(x,y):
    return  x**2 + y**2
```

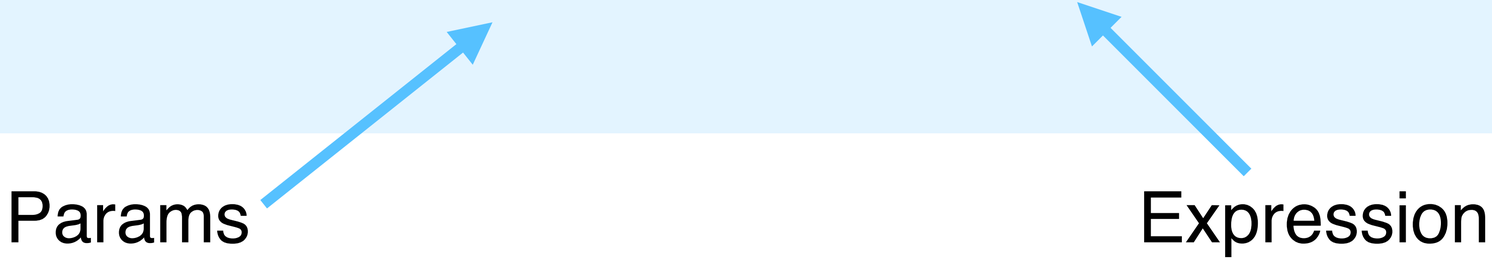
```
lambda x, y : x**2 + y**2
```



Let’s see how these are used

```
def square_add_two(x,y):
    return  x**2 + y**2
```

```
lambda x, y : x**2 + y**2
```



Map

- Takes in a function, and an iterable
- Applies that function to the iterable

Map

```
numbers = [1, 2, 3, 4, 5]

# Using map with a lambda function to square each number
squared_numbers = list(map(lambda x: x**2, numbers))

print(squared_numbers)
```

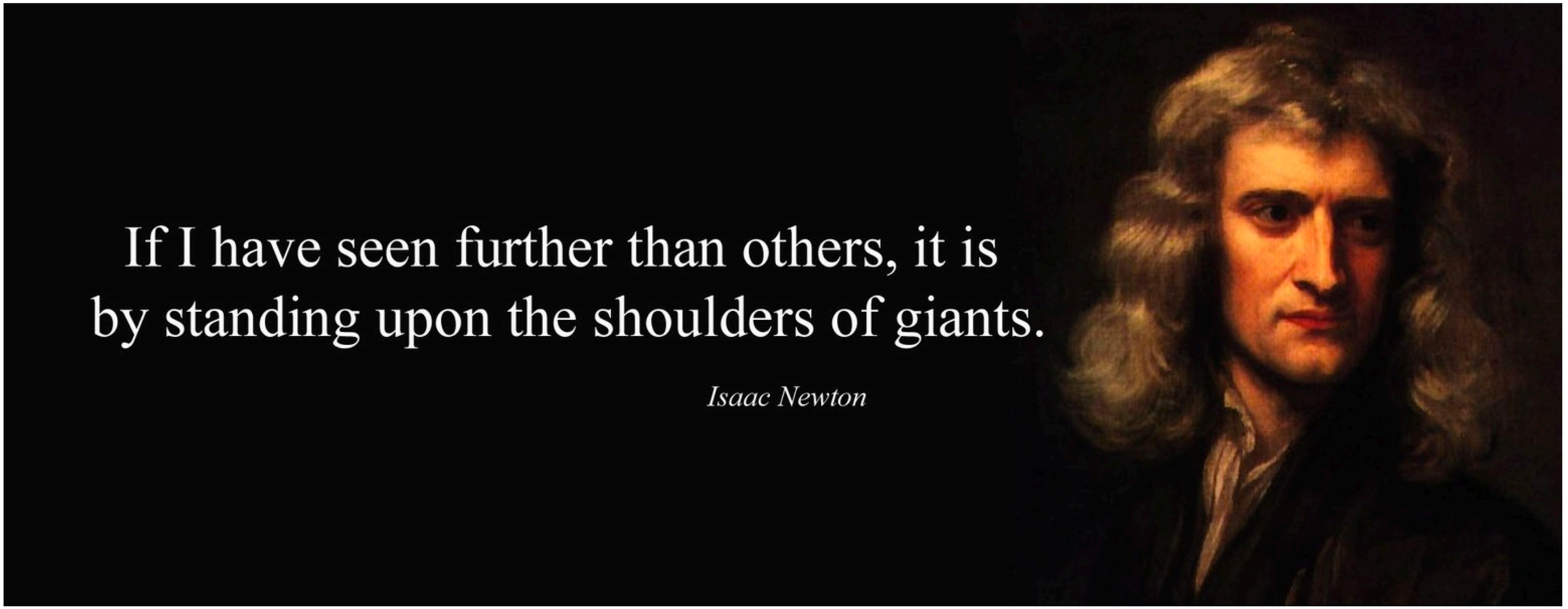
To call a lambda, you can put the arguments in parenthesis after the definition (also in paraenthesis)

```
(lambda x, y : x**2 + y**2)(8,5)
```

Let’s Try Some Out

Okay back to today’s topic

Standard Library

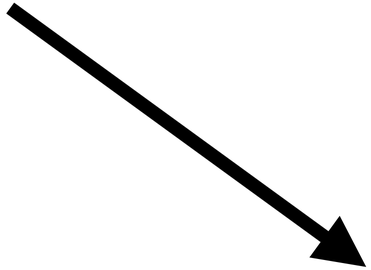


Standard Library

- The packages and modules that come with python
- Viewed as important/fundamental to the developers
- To access them, just need to import!

Modules

- Smallest unit of reusable python code.
- Basically a file with functions and statements
- We write modules allll the time



Package

- A logical collection of modules
- E.g. numpy, random, ect

Where can I find these?

- https://docs.python.org/3/library/

Import Conventions

```
sound/
      __init__.py
      formats/
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
sound/
        __init__.py
        formats/
                __init__.py
                wavread.py
                wavwrite.py
                aiffread.py
                aiffwrite.py
                auread.py
                auwrite.py
                ...
        effects/
                __init__.py
                echo.py
                surround.py
                reverse.py
                ...
        filters/
                __init__.py
                equalizer.py
                vocoder.py
                karaoke.py
                ...
```

```
import sound

sound.effects.echo.echofilter(a, b)
```

```
sound/
        __init__.py
        formats/
                __init__.py
                wavread.py
                wavwrite.py
                aiffread.py
                aiffwrite.py
                auread.py
                auwrite.py
                ...
        effects/
                __init__.py
                echo.py
                surround.py
                reverse.py
                ...
        filters/
                __init__.py
                equalizer.py
                vocoder.py
                karaoke.py
                ...
```

```
import sound

sound.effects.echo.echofilter(a, b)
```

```
from sound.effects import echo

echo.echofilter(a, b)
```

```
sound/
        __init__.py
        formats/
                __init__.py
                wavread.py
                wavwrite.py
                aiffread.py
                aiffwrite.py
                auread.py
                auwrite.py
                ...
        effects/
                __init__.py
                echo.py
                surround.py
                reverse.py
                ...
        filters/
                __init__.py
                equalizer.py
                vocoder.py
                karaoke.py
                ...
```

```
import sound

sound.effects.echo.echofilter(a, b)
```

```
from sound.effects import echo

echo.echofilter(a, b)
```

```
from sound.effects.echo import echofilter

echofilter(a, b)
```

Let’s explore some Standard Libraries

pickle

`pickle`

- Pickle lets you serialize/unserialize python objects
- Turns python objects into binary representations

pickle

Method	Use
`pickle.dump(obj, file)`	Write pickled representation of arbitrary object obj into the file object file.
`pickle.dumps(obj)`	Return pickled representation of arbitrary object obj as a bytes string.
`pickle.load(file)`	Read the object which has been pickled into `file` into memory, return the object.
`pickle.loads()`	Execute the current statement, proceed to the next one.

When would you want to use this?

- Want to store something efficiently
- Sending information between Processes

An example

Safety around pickle

PDB

Two Ways

- In-file
- In Terminal

pdb

Command	Use
`pdb.set_trace()`	Sets breakpoint; launches `pdb` when execution encounters the breakpoint.
`p` *`expression`* or `print` *`expression`*	Prints the value of *`expression`*.
`c` or `continue`	Continue execution from the current breakpoint.
`n` or `next`	Execute the current statement, proceed to the next one.
`s` or `step`	Execute the current statement, step into the function.
`b` *`line`* or `b` *`function`*	Sets a breakpoint at line number *`line`*, or at the beginning of *`function`*.

Two Ways

- In-file
- In Terminal
  - python3 -m pdb file.py

collections

collections

Command	Use
`collections.namedtuple`	Object that supports name-binding to elements of a tuple.
`collections.defaultdict`	Object that supports default dictionary values (to reduce need for error catching when working with dictionaries).
`collections.Counter`	Simplify counting of elements in a collection.

functools

functools

Command	Use
`functools.cache`	Caches the output of a function (like the decorator we wrote in the FP lecture).
`functools.wraps`	Updates a wrapper function to look like the wrapped function. (`__name__`, `__doc__`, etc.)
`functools.partial`	Returns a function object with some positional and keyword arguments pre-filled.

itertools

itertools

Command	Use
`itertools.cycle(iterable)`	Returns an infinite iterable, cycling through the elements of `iterable`.
`itertools.count(start [,step])`	Returns an infinite iterable, counting up from `start` in increments of `step`.

Re

Character	Brief Description	Simple Example
.	any character (wildcard)	w..kly
^	Begins with	^where
[ ]	character set	[0-9]
|	either or	hi|bye
+	Once or more	me+
*	zero occurances or more	me*
{}	exact number of times	I{4}
\	special character operations	\w
$	Ends with	$bar

Command	Use
`re.findall(pattern, string)`	Returns a list of all non-overlapping substrings of `string` that match `pattern`.
`re.match(pattern, string)`	Returns `MatchObject` (allows for easy match processing) if string matches pattern, `None` otherwise.

"\(\d{3}\)[-]\d{3}-\d{4}"

(123) 456-7890

Non-SL Packages

- (Packages not in S.L. must be downloaded with something like pip)
- Some popular ones you might recognize
    - Flask
    - Numpy
    - Pandas
    - Requests

These all have their own documentation

oject Links

ate
l Releases
rce Code
e Tracker
t

ntents

come to Flask
ser's Guide
API Reference
Additional Notes

uick search

Go

Flask

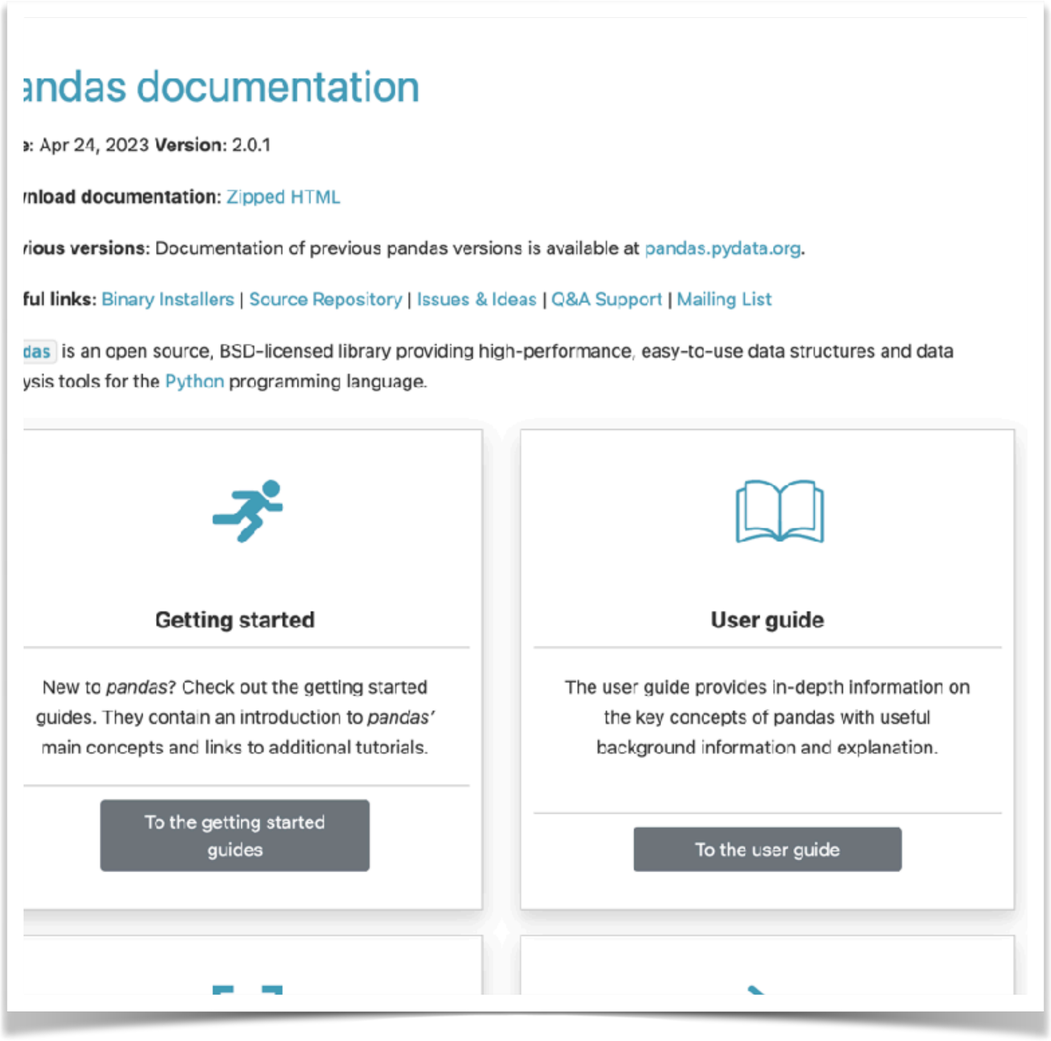
Welcome to Flask's documentation. Get started with Installation and then get an overview with the Quickstart. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Common patterns are described in the Patterns for Flask section. The rest of the docs describe each component of Flask in detail, with a full reference in the API section.

Flask depends on the Werkzeug WSGI toolkit, the Jinja template engine, and the Click CLI toolkit. Be sure to check their documentation as well as Flask's when looking for information.

User's Guide

Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

- Installation
  - Python Version
  - Dependencies
  - Virtual environments
  - Install Flask
- Quickstart
  - A Minimal Application
  - Debug Mode



andas documentation

: Apr 24, 2023 **Version:** 2.0.1

nload documentation: Zipped HTML

ious versions: Documentation of previous pandas versions is available at pandas.pydata.org.

ful links: Binary Installers | Source Repository | Issues & Ideas | Q&A Support | Mailing List

das is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data ysis tools for the Python programming language.

Getting started

New to *pandas*? Check out the getting started guides. They contain an introduction to *pandas'* main concepts and links to additional tutorials.

To the getting started guides

User guide

The user guide provides in-depth information on the key concepts of pandas with useful background information and explanation.

To the user guide

Be careful with pip install!

- In 2018, a package called "acqusition" was uploaded to PyPI, imitating the well-known package "acquisition."
- The malicious package contained code to collect user SSH and GPG keys and send them to a remote server.
- The package was removed, but it highlighted the importance of double-checking package names and sources.