Rock Solid PYTHON For Beginners

Jason Barhorst
www.JBarCode.com

Pre-Reqs (on YouTube)

- Is Python Installed?
  - python --version
  - python3 --version
- Install Python:
  - https://www.python.org
  - Windows Install:
    - YouTube Link
  - Linux Install:
    - YouTube Link
  - Mac Install:
    - ...when I revive my old Mac
- Install Git (optional):
  - https://git-scm.com/download
- Install VS Code (optional):
  - https://code.visualstudio.com/

About Me

- Started programming in 2005 w/ C++
- B.S. Electrical Engineering:
  - Minor in Math and Physics
  - Track in Electricity and Magnetism
  - Matlab TA for 3 years
- M.S. Electrical Engineering:
  - Focus in Guidance, Navigation & Control Systems
- First tried Python in 2015, but community stuck on 2.7
  - As libraries started supporting Python 3.X, I came back
- Favorite Projects:
  - MTGO Online Trading Bots (C++, SQL)
  - Attitude Determination and Control for a CubeSat
  - Auto Pilot & Sensor Integration for Autonomous Vehicles (C, C++, Matlab, Python)
  - Various ML and AI projects -> Going deep on Reinforcement Learning
  - Getting Started in Algorithmic Trading (Python & C++)
  - Graph Databases (Neo4j and Python)

Day 1:

0. Mindset & Course Setup
1. Introduction to Python
   a. Variable Assignments!
2. Working with Data
   a. Start Functions ASAP

Day 2:

3. Program Organization
   a. What is "__main__"
4. Classes and Objects
   a. Objects Everywhere

Day 3:

5. Inner Workings of Python Objects
6. Generators
7. A Few Advanced Topics

Day 4:

8. Testing, Logging, and Debugging
9. Packages
10. What’s Next

Who is this Course For?

- New Programmers Learning Their First Language
- People Looking to Master the Basics of Python
- People ready to write the code (just watching the videos won’t go far)

Course Availability

- Playlist or One-Video Marathon on YouTube:
  - Gain Familiarity with Python
- Comprehensive Examples:
  - Gain Working Knowledge
- Complete the Budget Project on Your Own
  - Learning how to build projects on your own
- For Mastery Consider the Online Course:
  - Coming soon
  - Additional Benefits of the Course:
    - Quizzes
    - Interactive Exercises
    - Mastery Tests
- Follow for Live Courses:
  - https://www.facebook.com/JBarCode37
  - https://www.meetup.com/Rock-Solid-Computer-Programming

0. Mindset & Course Setup

Section 0: Mindset

Nobody is self-taught:

- Even if you learn from documentation someone or many people wrote that for you.

Everyone is self-taught:

- Even highly educated people need to spend a ton to time reviewing, learning, and teaching themselves.
  - Just attending a class is never enough
- It’s critical that you take responsibility for your learning and education
  - In this course and everywhere else!

Section 0: Mindset

- Pygmalion (Rosenthal) Effect
  - Teacher’s belief about a student’s ability impacted their ability to learn and perform.
  - Hard to repeat & implement:
    - Teachers already familiar with students
    - Teachers made aware of study being conducted
- Your belief in your ability to learn something or perform a task greatly impacts your ability to learn that thing or perform that task!

Section 0: Mindset

Math is easy!

- In typical math classes, there is a correct solution. You just have to try enough times to find it.

...So is programming!

- Programming not working?
    - You’re not dumb, you just haven’t uncovered the right solution yet!
    - Look at some of the programming achievements out there, many things are possible

Section 0: Mindset

Importance of completing the code on your own:

- Watching the videos will give you familiarity with the content
- Programming the examples with me will allow you to understand written code
- Programming projects on your own will allow you to be creative and ‘think’ in python
    - ...I still kind of think in Matlab from doing so much of it

Don’t copy-paste:

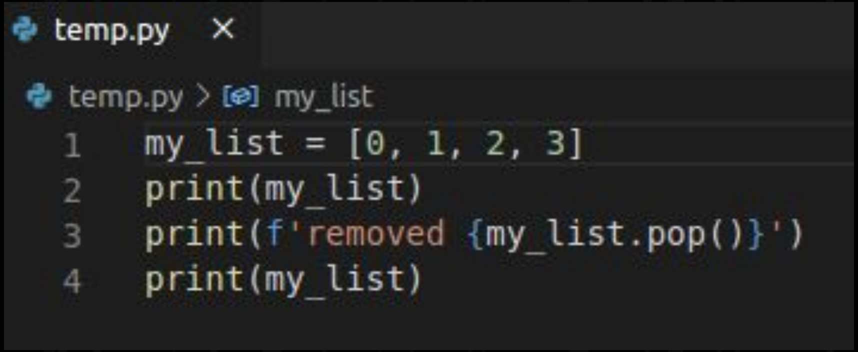
- Currently I don’t provide the code for the examples or projects...not sure I ever will

Section 0: Mindset

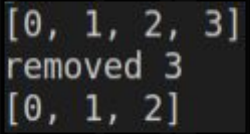
Importance of dabbling...we'll do it a lot in this course

For example: If I pop an item from a list, does it remove the first or last item?

Not sure, just use the prompt or make a temp program to try it out.



```python
my_list = [0, 1, 2, 3]
print(my_list)
print(f'removed {my_list.pop()}')
print(my_list)
```



```
[0, 1, 2, 3]
removed 3
[0, 1, 2]
```

Section 0: Mindset

Bookmark IDE Shortcuts Cheat Sheet Open

- At the beginning of a session choose one shortcut to try in your workflow
- Copy/Paste Example:
  - mouse copy paste
  - Mouse highlight, ctrl-c, ctrl-v
  - end, shift+up, ctrl-c, ctrl-v, ctrl-v
  - alt+shift+down

Bookmark Python Documentation Open

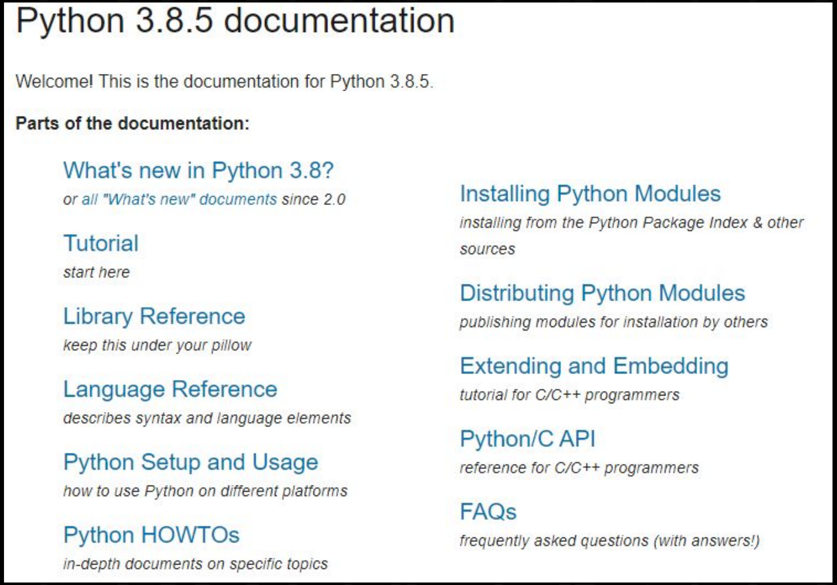
- https://docs.python.org/3/

Time Requirements

- This course is currently being run in 2-hour blocks, 4 days/week, for the next two weeks.
- -> approx 16 hours of instruction will be available through this course
- It could end up being longer
- If you can’t attend the live sessions, feel free to keep up on YouTube
- Both the live sessions and pre-recorded sessions will be posted

Python Documentation

- Bookmark the Documentation!
  - https://docs.python.org/
- Did you bookmark it?
- Learning to explore and reference the documentation is the most important thing you can learn in this course
- Always have this open when you code, try to reference it first!
- Then search online (google, stackoverflow, etc.)



Python 3.8.5 documentation

Welcome! This is the documentation for Python 3.8.5.

Parts of the documentation:

What's new in Python 3.8?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Python Setup and Usage
how to use Python on different platforms

Python HOWTOs
in-depth documents on specific topics

Installing Python Modules
installing from the Python Package Index & other sources

Distributing Python Modules
publishing modules for installation by others

Extending and Embedding
tutorial for C/C++ programmers

Python/C API
reference for C/C++ programmers

FAQs
frequently asked questions (with answers!)

1. Introduction to Python

Introducing Python

- What is python?
  - Interpreted language with a wide variety of uses
- Where do you get python?
  - https://www.python.org
- Why was Python originally created?
  - Created by Guido van Rossum ~1990
  - Dev time for sys admin tasks in C was too long
  - Bridge between shell and C languages
- How is it used today?
  - Backend Web Servers (YouTube & Instagram)
  - Machine Learning & Data Science (nearly everyone)
  - Growing use in Finance and ...everywhere else

Python is Your New Calculator

- Open command prompt or terminal
- type “python” (windows) or “python3” on linux/mac
- Run a few samples:
    - 1 + 1
    - X = 4
    - Y = X - 20
    - 19 - 4
    - _ + 1
    - for i in range(5): print(i)
- Break it out anytime you need to make a quick calculation

Numbers

- Integer (int, …, -2, -1, 0, 1, 2, ...)
- Floating Point (float, aka decimals, -3.14, 12.62, 0.99999)
- Boolean (True, False)
- Complex (a + bi where i = sqrt(-1), yikes!) *

- Warning! Python uses dynamic typing

* Note python uses ‘j’ instead of ‘i’ for the imaginary number. Like an EE!

Basic Number Operations

+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Strings

Strings contain character including digits & special characters.

For Example:

x = ‘hello world’

Also:

x = “hello world”

Doc-Strings:

“”””

hello world

“”””

Common String Operations:

```
name = ‘Jason Barhorst’

name.upper()

name.lower()

name.title()

names = name.split(‘ ‘)

names[0] + ‘ ‘ + names[1]

‘ ‘.join(names)
```

Common String Operations:

.upper()

.lower()

.title()

String Slicing:

x = ‘abcdefg’

What are the following?

x[0], x[1], x[-1], x[:], x[::-1], x[1, 3], x[1, 3, -1], x[:, :, 2]

f-Strings: More like F-Yeah Strings!

The bad way:

```
x = 1
y = 2

print('x = ' + str(x) + ', y = ' + str(y))
```

String formatting the old way:

```
print('x = {}, y = {}'.format(x, y))
```

Using f-stings (requires Python 3.6 or higher):

```
print(f'x = {x}, y = {y}')
```

f-Strings: More like F-Yeah Strings!

```python
x = 1
y = 2.12341234
z = '3.1234123412'

# {variable:width.precision{type}}
# use f for float, s for string, and... d for integer
print(f'{x:15d}, {y:15.3f}, {z:15s}')
```

```python
# include '!r' after variable to print using __repr__
x = "[1, 2, 3, 4]"

# displays nicely for end users
print(x.__str__())
print(f'x = {x}')

# more useful representation for dev
print(x.__repr__())
print(f'x = {x!r}')
```

Variable Names: Good, Bad, & Stupid

Good:

- start_with_lowercase, use ‘_’ to separate words
- Just use ‘_’ for throw away variables
- Using meaningful names ‘target_value = …”

Bad:

- UsingCamelCase (this is not the standard in Python)
- Poor naming scheme (x, x1, xl, xXxx, xxXx, Il1I11I1l, blue=’red’, etc.)

Stupid:

- Using built in function names (int, float, min, max, etc.)

A First Program

1. Create a file called ‘my_budget.py’
2. Ask the user for their savings account balance using input().
3. Print the value of the savings account with a ‘$’ in front of the number.

The output could look something like:

```
Congrats! You have $4.22

Congrats! You have $49999.42
```

Did you properly display the decimal length?

Problems? Where’s the comma!?!

Second Program (Probably should have been first :)

1) Ask the user what their name is
2) Print a greeting to the user

Possible output:

```
Hi Chuck, that’s a nice name.

Hello Chuck! Nice “name” Chuck!
```

Functions (be sure to type up the examples)

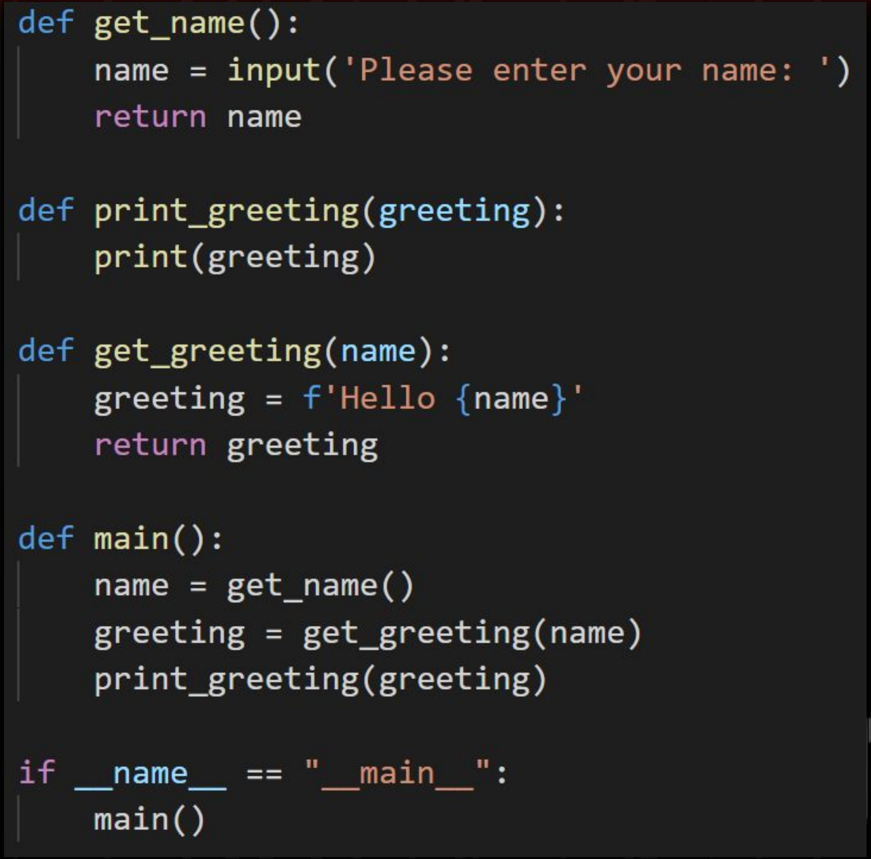
Created using the keyword ‘def’

Spacing is extremely important

Inputs optional

Output (return) optional

WTF is that ================>>>



```python
def get_name():
    name = input('Please enter your name: ')
    return name

def print_greeting(greeting):
    print(greeting)

def get_greeting(name):
    greeting = f'Hello {name}'
    return greeting

def main():
    name = get_name()
    greeting = get_greeting(name)
    print_greeting(greeting)

if __name__ == "__main__":
    main()
```

Update Budget Program

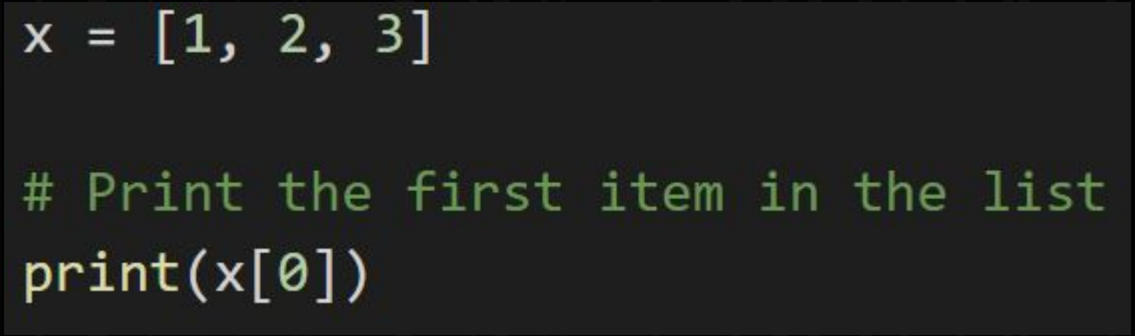
1. Create a main() function to contain the code you want to run
2. Modify ‘my_budget.py’ to make use of __name__ == ‘__main__’
3. Create a function with no inputs, that asks the user to enter their savings account balance and returns the value as a float
4. Create a function has one input ‘savings’ and displays the amount with two decimal places
5. Create a function with no inputs, that asks the user for their monthly income and returns the value as a float
6. Create a function that has one input ‘monthly_income’ and displays the amount with two decimal places

BarCode

2. Data Structures

Lists (Arrays)

- Start a list with ‘[‘ end with ‘]’
- Separate values with a ‘,’
- Data type is ‘list’
- Lists can contain numbers, strings, even other lists
- Python is extremely flexible for working with list, terribly flexible
- Reference individual items with brackets, index 0 is the first position



```
x = [1, 2, 3]

# Print the first item in the list
print(x[0])
```

List Methods

append ...an item to the end

clear

copy

count ...number of elements that match input

extend ...append items from another list

index ...finds the index of a specified value

insert ...inserts an item at specified location

pop ...removes end item, or item in specified loc

remove ...removes first instance of item

reverse ...different from [::-1]

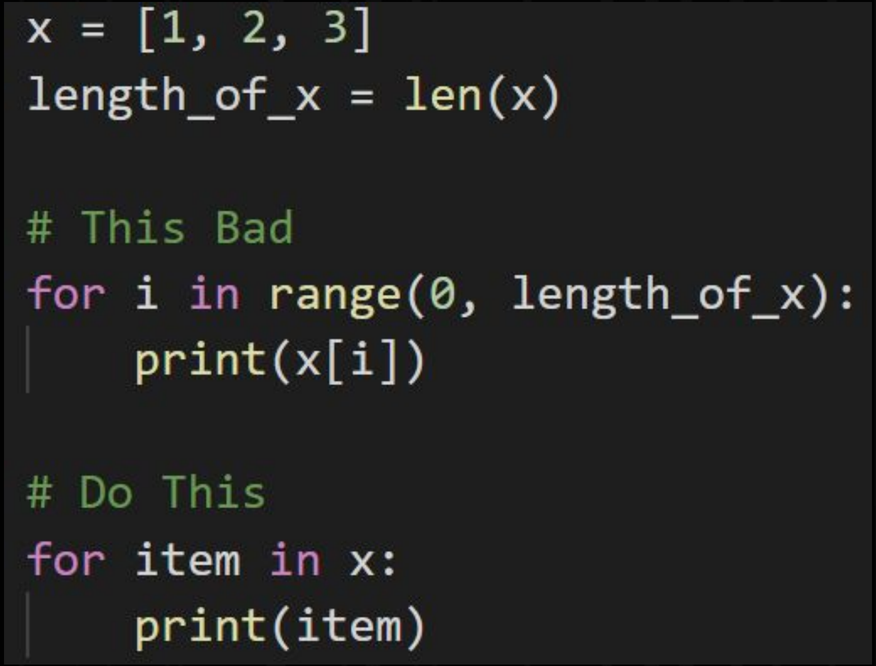
sort ...it sorts in place

Useful Built In Functions:

min, max, sorted, sum, len

Other Data Structures are dead to me...For Loops!

- We’ll talk about tuples, sets, dictionaries later
- Let’s take a look at ‘for’ loops because they often work with loops



```python
x = [1, 2, 3]
length_of_x = len(x)

# This Bad
for i in range(0, length_of_x):
    print(x[i])

# Do This
for item in x:
    print(item)
```

Programming Challenge

We don’t have a built in mean (average), lets build one

1. Write a function named ‘mean’ (this is kind of a bad name)
2. It takes one input, a list of numbers
3. Using the functions/methods on the previous page, calculate the average
4. Return the average value

$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$

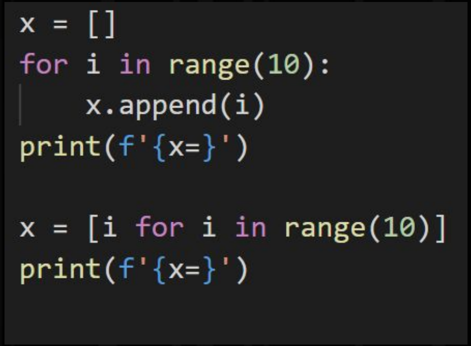
$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

Budget Program Update

1) Create a list containing your monthly bills by…
    a) Make a list of the bill names
    b) Make an empty list to store bill values
    c) Use a for loop to iterate over the name of the bill
        i) On each iteration, use ‘input’ to collect the bill data
2) Display each bill and value
3) Use sum to calculate the total monthly bills and display the value

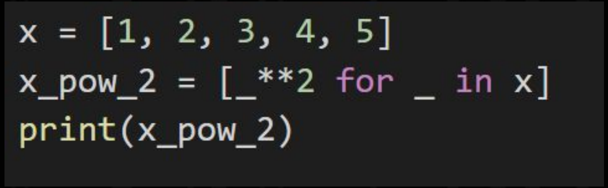
List Comprehensions

- A simple way to create a list with a for loop



```python
x = []
for i in range(10):
    x.append(i)
print(f'{x=}')

x = [i for i in range(10)]
print(f'{x=}')
```

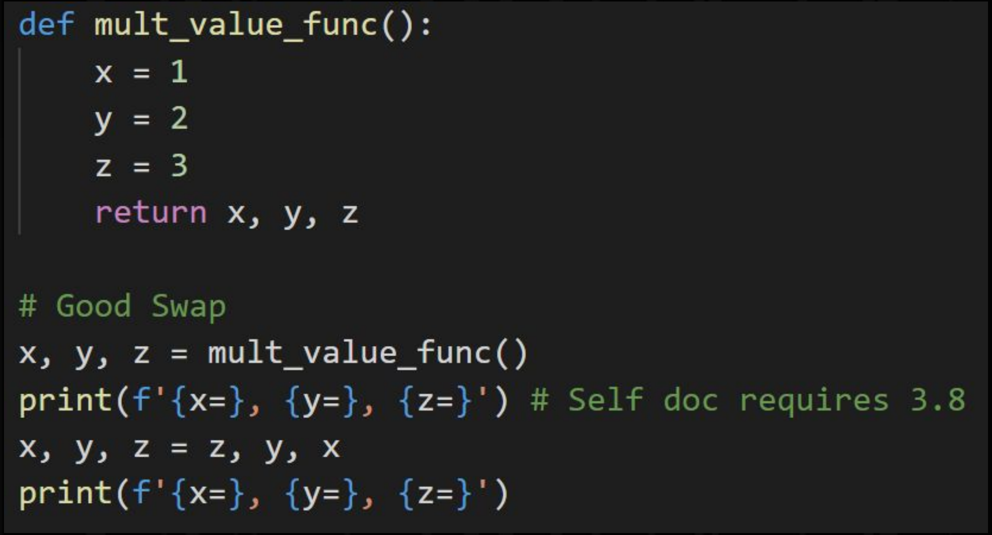
- Perform an operation on each element of a list



```python
x = [1, 2, 3, 4, 5]
x_pow_2 = [_**2 for _ in x]
print(x_pow_2)
```

Tuples

- Immutable
- Way to manipulate multiple items
- Common uses include:
  - Input to a function to provide multiple values
  - Return value of a function to return multiple values
  - Swap functionality
  - zip function



```python
def mult_value_func():
    x = 1
    y = 2
    z = 3
    return x, y, z

# Good Swap
x, y, z = mult_value_func()
print(f'{x=}, {y=}, {z=}') # Self doc requires 3.8
x, y, z = z, y, x
print(f'{x=}, {y=}, {z=}')
```

Logic Operations (Boolean Algebra 1 + 1 = 1?)

- bool type (True or False)

  ==, !=

  >, <, >=, <=

  and, or, not

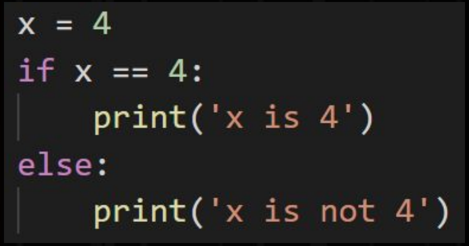
  is, is not

  in, not in

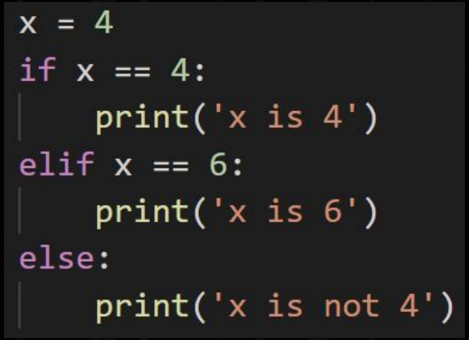
  Try with different data types: [], “”, (), {}, etc.

if, elif, else

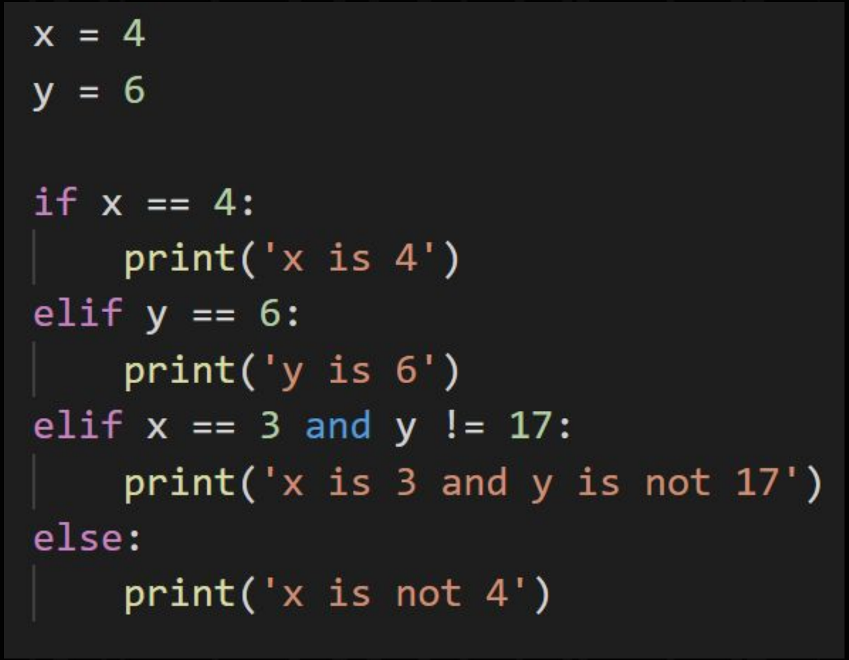
Control which blocks of code run based on values



```
x = 4
if x == 4:
    print('x is 4')
else:
    print('x is not 4')
```



```
x = 4
if x == 4:
    print('x is 4')
elif x == 6:
    print('x is 6')
else:
    print('x is not 4')
```



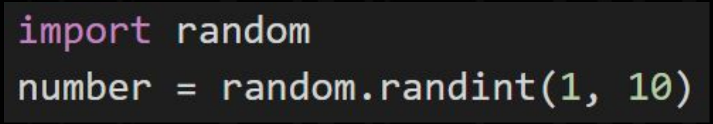
```
x = 4
y = 6

if x == 4:
    print('x is 4')
elif y == 6:
    print('y is 6')
elif x == 3 and y != 17:
    print('x is 3 and y is not 17')
else:
    print('x is not 4')
```

While Loops

Perform a task as long as condition is true

Coding Exercise

Use a while loop to create a number guessing game. The following block of code will generate a number from 1-10 to try to guess. Put it at the top of your program.

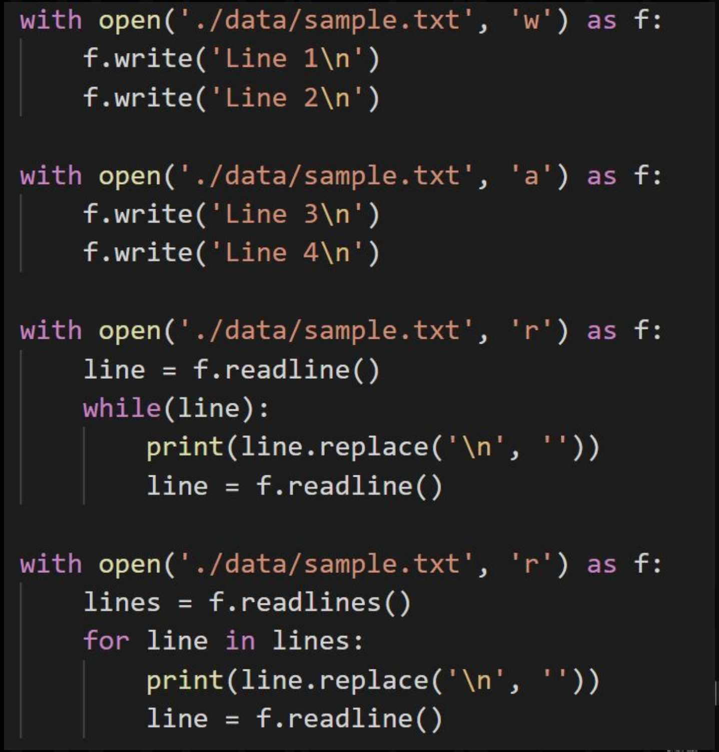


```
import random
number = random.randint(1, 10)
```

1) Use input to prompt the user for a number (convert to int)
2) Use a while condition to keep the user playing until the correct number is chosen
    a) Use if/else respond if they need to guess higher or lower
    b) Collect a new guess using input
3) After the loop has been exited, congratulate the user
4) Bonus: Add a counter to limit how many times they can guess

Reading and Writing Text Files

Use ‘with’ to open file (auto close after)



```python
with open('./data/sample.txt', 'w') as f:
    f.write('Line 1\n')
    f.write('Line 2\n')

with open('./data/sample.txt', 'a') as f:
    f.write('Line 3\n')
    f.write('Line 4\n')

with open('./data/sample.txt', 'r') as f:
    line = f.readline()
    while(line):
        print(line.replace('\n', ''))
        line = f.readline()

with open('./data/sample.txt', 'r') as f:
    lines = f.readlines()
    for line in lines:
        print(line.replace('\n', ''))
        line = f.readline()
```

Reading and Writing CSV Files

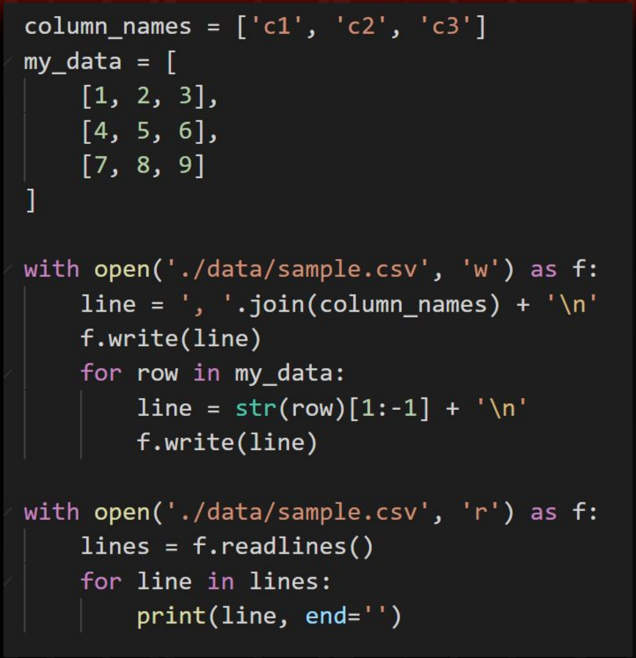
CSV: Comma Separated Values

Often first line is heading, following lines are data

Primitive table or spreadsheet

Use split go from a csv string (one line) to a list

Use split go from a csv string (one line) to a list



```python
column_names = ['c1', 'c2', 'c3']
my_data = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

with open('./data/sample.csv', 'w') as f:
    line = ', '.join(column_names) + '\n'
    f.write(line)
    for row in my_data:
        line = str(row)[1:-1] + '\n'
        f.write(line)

with open('./data/sample.csv', 'r') as f:
    lines = f.readlines()
    for line in lines:
        print(line, end='')
```

If you work with a lot of spreadsheets, never do this…

Use pandas!

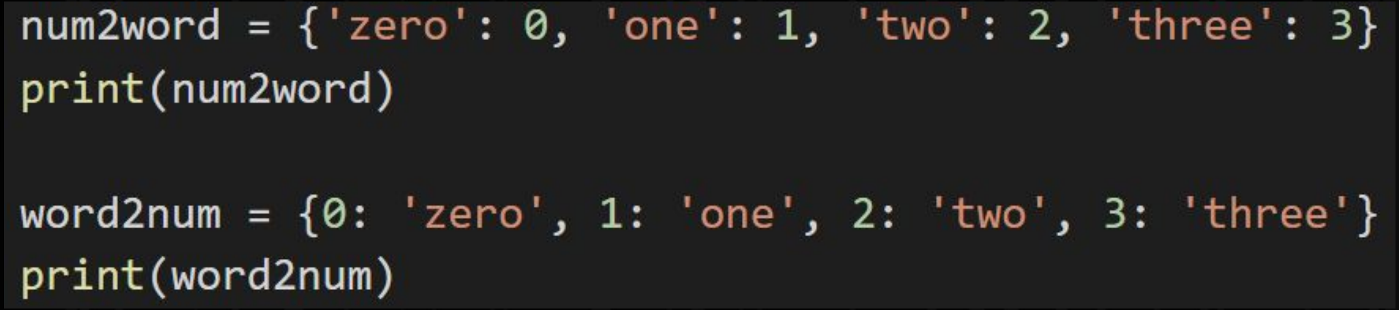
Add Goals to Budget Data

Create a function that asks the user for 3 goals and stores them to a file named ‘goals.txt’

1) Created a variable ‘n’ and set it equal to the number of goals
2) Use a while loop to ask the user to enter a goal on each pass through the loop
3) Write the goals to ‘goals.txt’
    a) This can be done by storing all goals in an array, then writing them all at once
    b) Alternatively, append to the file on each iteration of the loop (this is worse)

Dictionaries

- Use ‘dict()’ or ‘{}’ to create a dictionary
  - Using ‘{}’ is the standard way
- Also known as hash tables
- Map of key, value pairs
  - Use the key to store, update, or read a value
- Separate key and value with ‘:’
- Separate key, value pairs with ‘,’
- Order wasn’t guaranteed, but now it (Python 3.6)



```
num2word = {'zero': 0, 'one': 1, 'two': 2, 'three': 3}
print(num2word)

word2num = {0: 'zero', 1: 'one', 2: 'two', 3: 'three'}
print(word2num)
```

Dictionary Methods

clear

copy

fromkeys

get

items

keys

pop

popitem

setdefault

update

values

- Other items:
  - **
  - del

Dictionaries and JSON Files

JavaScript Object Notation (JSON) used to send data to/from frontend

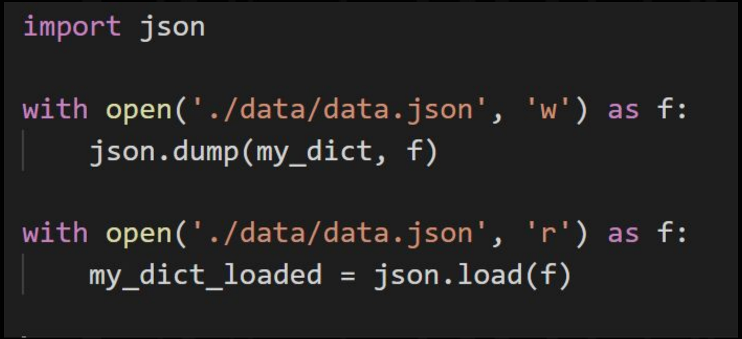
Useful way to store data locally if you don’t want to make a database

Works well with lists and dictionaries.

To save dictionary data to a json-file, use json.dump(dict_name, f_handle)

To load dictionary data from a json-file, use json.load(f_handle)

It’s just that easy



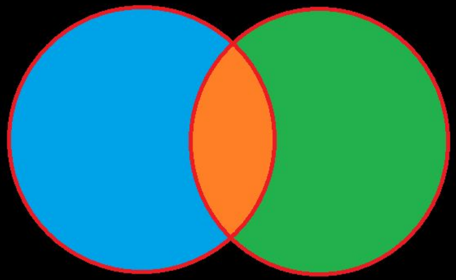
```python
import json

with open('./data/data.json', 'w') as f:
    json.dump(my_dict, f)

with open('./data/data.json', 'r') as f:
    my_dict_loaded = json.load(f)
```

Budget Program Update

1) Reaccomplish the monthly bills task with dictionaries by…
    a) Make a list of the bill names
    b) Create a dictionary with ‘fromvalues’
        i) Set the initial values to 0
    c) Use a for loop to iterate over the keys
        i) On each iteration, use ‘input to collect the bill data
        ii) Update the dictionary with the new value
2) Use loops with dict.items() to display the results
3) Use dict.values() and sum to calculate the total monthly bills and display the value
4) Save the dictionary to a file named ‘bills.json’

Sets

- Use ‘set()’ to create an empty set ‘{}’ creates an empty dictionary
- Separate values with a comma
- Think sets from math class...venn diagrams
- Only contain unique items, unordered and unindexed
- Helpful for finding unique items, checking for completed items, or...for homework using sets
- You will use lists much more often, but when sets fill the role, they should be used



Sets

add

clear

copy

difference

difference_update

discard

intersection

intersection_update

isdisjoint

issubset

issuperset

pop

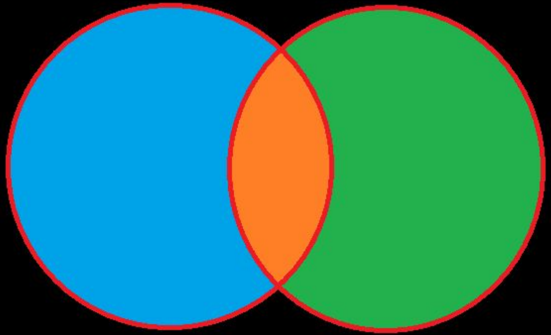
remove

symmetric_difference

symmetric_difference_update

union

update



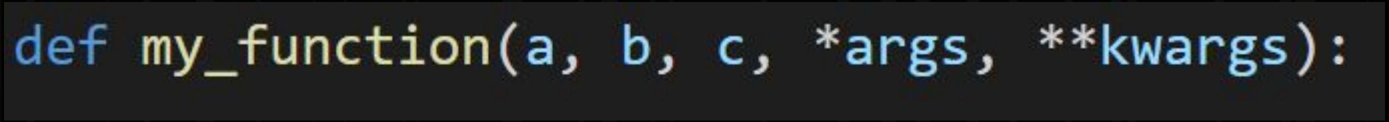
Revisit Function Arguments (*args, *kwargs)

*args accepts any number of additional positional Arguments and stores them in a tuple named args

**kwargs accepts any number of keyword arguments and stores them in a dictionary named kwargs:

- my_flag = False
- **my_dict

Order matters, use:



```python
def my_function(a, b, c, *args, **kwargs):
```

Concludes Basic Python, I/O, and Functions

- int, float, string, bool
- list, tuple, dict, set
- if, elif, else
- for, while
- with, open, import json
- def
- *args, **kwargs

Challenge:

Write a useless program that incorporates each of the above concepts.

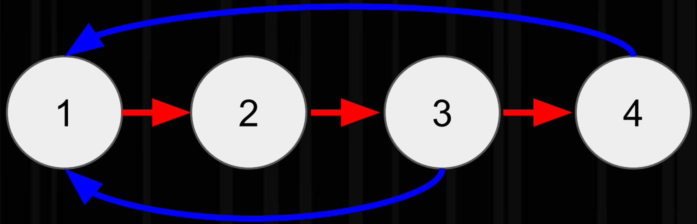
Program Flow Control

State Machine

Infinite Loops

Exit Condition

Enum to make it more clear


$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] \arrow[ll, bend left] & 4 \arrow[lll, bend right]
\end{tikzcd}$$

Try on your own:

Add flow control to the my_budget program for loading and saving budget items (goals, bills, account balances).

Error Checking

Handle errors to keep program running

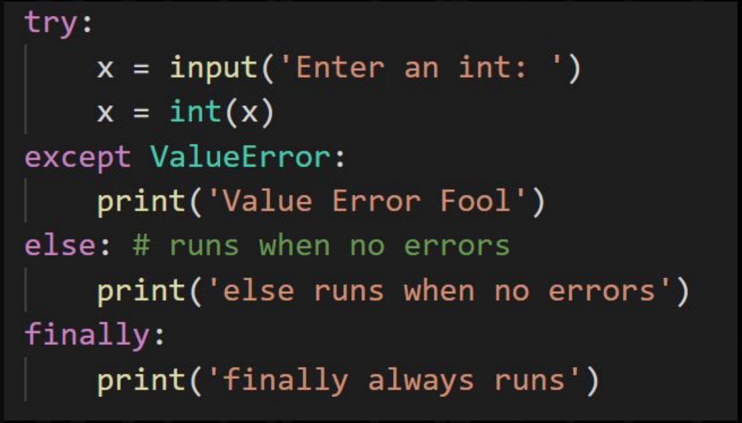
Use try, except, else, finally to handle error cases

Use raise to create your own errors

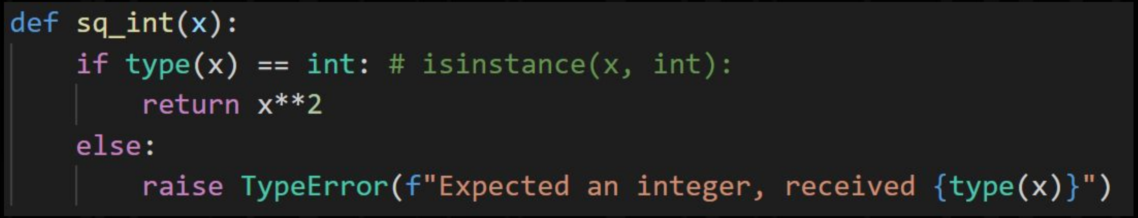
Power to abuse:

- Try instead of handling data types

Use raise to generate your own errors



```python
try:
    x = input('Enter an int: ')
    x = int(x)
except ValueError:
    print('Value Error Fool')
else: # runs when no errors
    print('else runs when no errors')
finally:
    print('finally always runs')
```



```python
def sq_int(x):
    if type(x) == int: # isinstance(x, int):
        return x**2
    else:
        raise TypeError(f"Expected an integer, received {type(x)}")
```

Programming Challenge

Use error handling to write two new input functions. Remember that input always creates a string, but you often want the user to input an int or float.

Write a function named input_int:

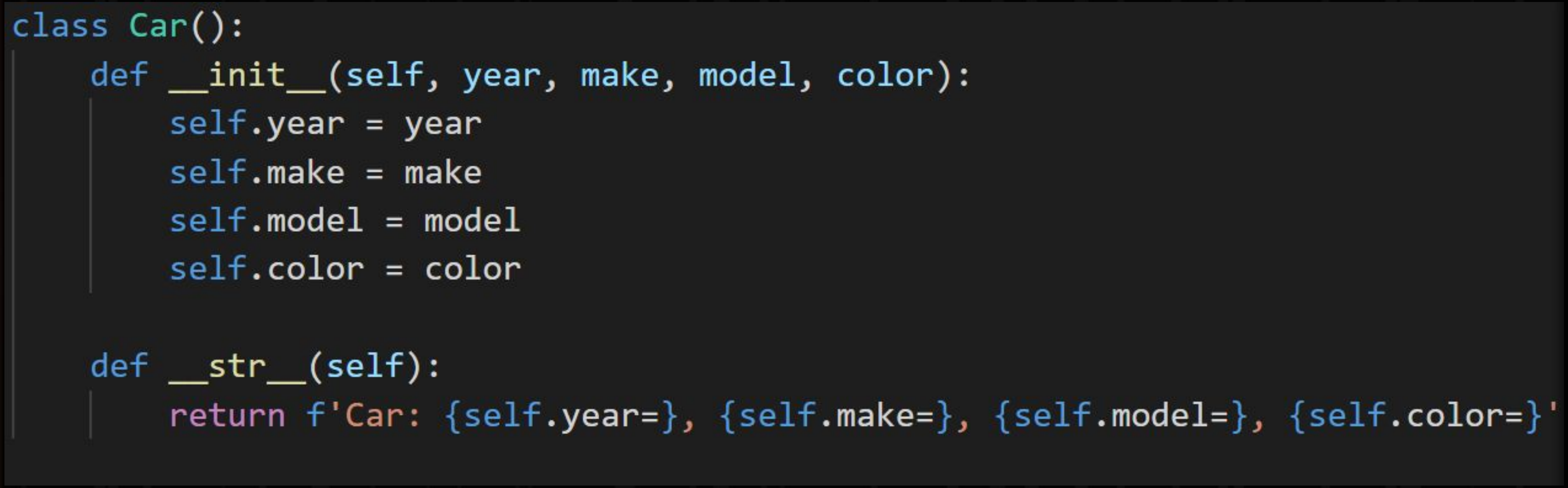
- Accepts a string as input to prompt the user for input
- Returns the input cast as an int
- If it fail, notify the user of the error and try again

Write a similar function named input_float:

- Accepts a string as input to prompt the user for input
- Returns the input cast as a float
- If it fail, notify the user of the error and try again

Object Oriented Programming (OOP)

Many things can be modeled as objects. For example a car:



```python
class Car():
    def __init__(self, year, make, model, color):
        self.year = year
        self.make = make
        self.model = model
        self.color = color

    def __str__(self):
        return f'Car: {self.year=}, {self.make=}, {self.model=}, {self.color=}'
```

It’s worth noting that an entire course could be dedicated to OOP

Magic Methods

__init__

__str__

__repr__

__add__

__sub__

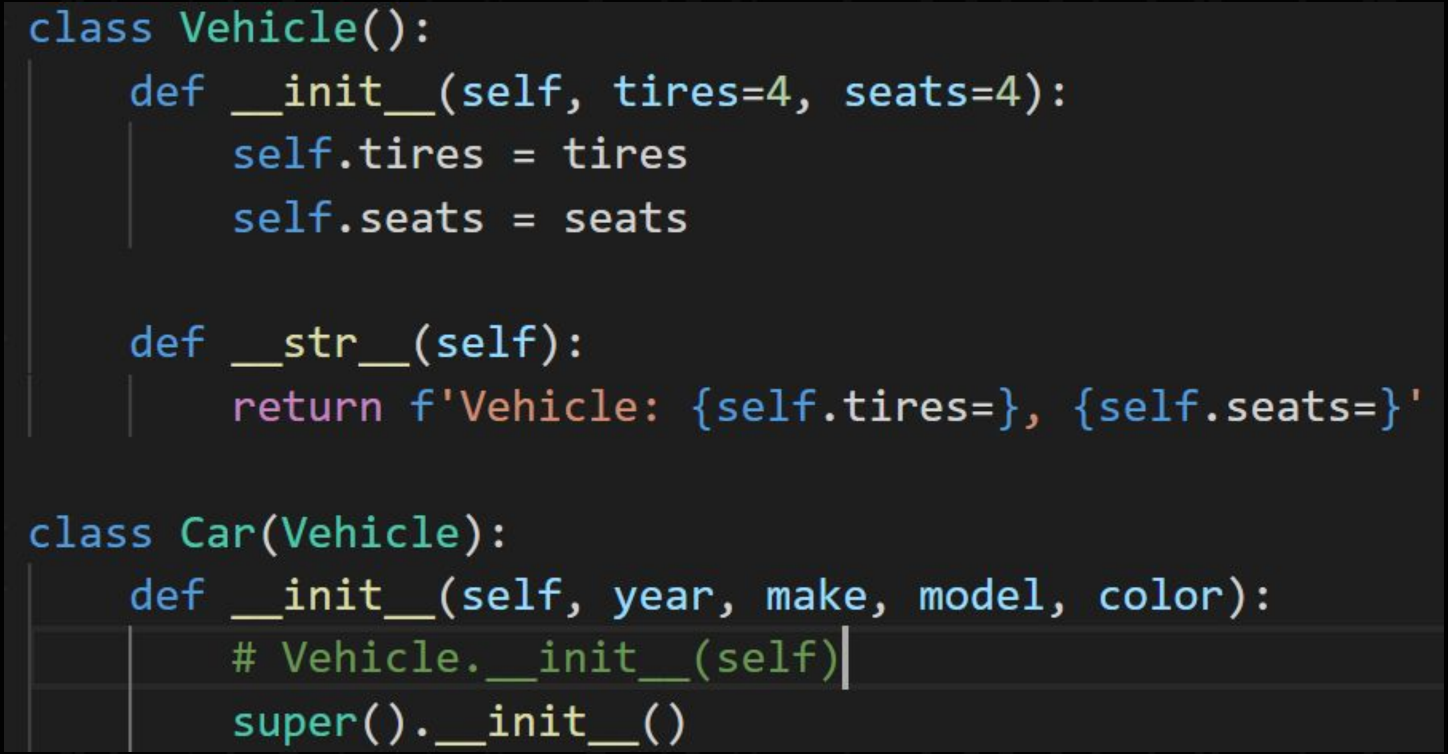
__abs__

__gt__

__lt__ …..and many many more at https://docs.python.org/3/reference/datamodel.html

Inheritance

Objects can be based on higher level objects:



```python
class Vehicle():
    def __init__(self, tires=4, seats=4):
        self.tires = tires
        self.seats = seats

    def __str__(self):
        return f'Vehicle: {self.tires=}, {self.seats=}'

class Car(Vehicle):
    def __init__(self, year, make, model, color):
        # Vehicle.__init__(self)
        super().__init__()
```

Create a class for coordinate points

The class will:

- Be named ‘Point’
- Instance variables x and y defined using __init__
- Define __add__, __sub__, __str__, and __repr__
  - Add x+x2 and y+y2
  - Sub x-x2 and y-y2
- Method to calculate the distance from another point, name it dist
  - Consider using import math, math.sqrt
- Define __abs__ so that abs(point) will return the distance from the origin
- Bonus: use @classmethod decorator to create an origin point:
  - This is similar to constructor overloading in other languages

Decorators

Start with the ‘@’ symbol

Wrapper of code that augments your function.

Common Decorators in Object Oriented Programming:

@classmethod

@staticmethod

@property (similar to a getter in other languages)

@variable_name.setter (similar to setter in other languages)

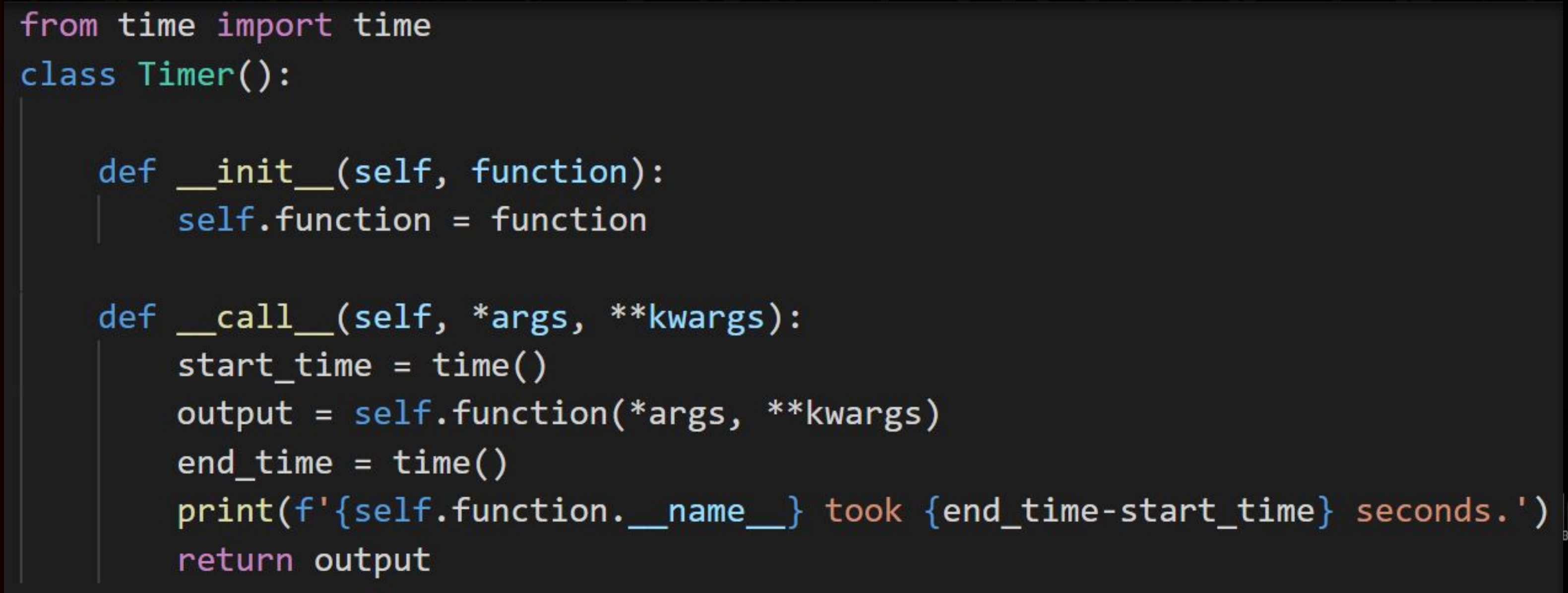
@varable_name.deleter (can be used to clear values)

User Defined Decorators

Functions are objects and can be assigned and passed just like variables.

Your decorator code goes in __call__

Your decorator name will be the class name



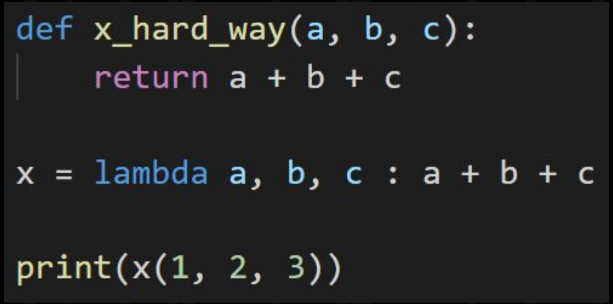
```python
from time import time
class Timer():

    def __init__(self, function):
        self.function = function

    def __call__(self, *args, **kwargs):
        start_time = time()
        output = self.function(*args, **kwargs)
        end_time = time()
        print(f'{self.function.__name__} took {end_time-start_time} seconds.')
        return output
```

Anonymous Functions (Lambda)

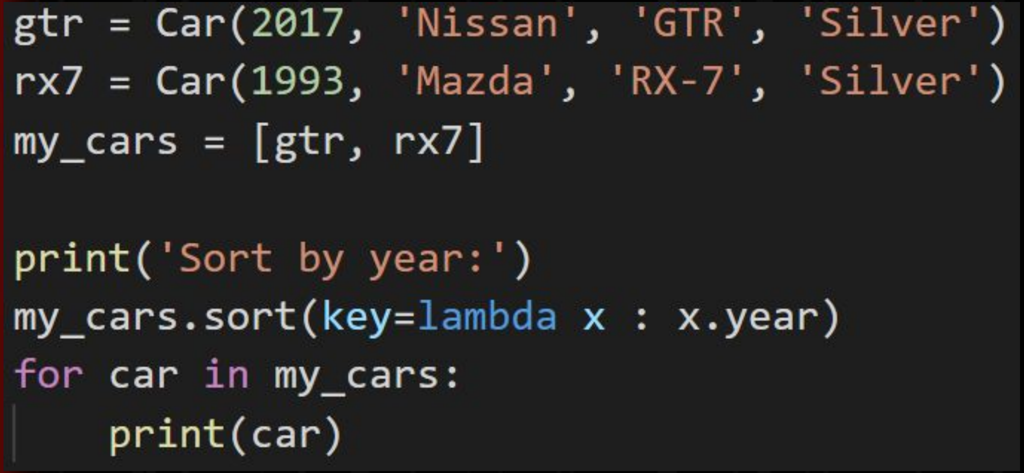
Anonymous function, sometimes called inline functions, are known as lambda functions.

Function has one expression.

Often passed into other functions such as sort



```python
def x_hard_way(a, b, c):
    return a + b + c

x = lambda a, b, c : a + b + c

print(x(1, 2, 3))
```



```python
gtr = Car(2017, 'Nissan', 'GTR', 'Silver')
rx7 = Car(1993, 'Mazda', 'RX-7', 'Silver')
my_cars = [gtr, rx7]

print('Sort by year:')
my_cars.sort(key=lambda x : x.year)
for car in my_cars:
    print(car)
```

Packages

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

math

timedate

unittest

logging

timeit

Math

Common math functions built in

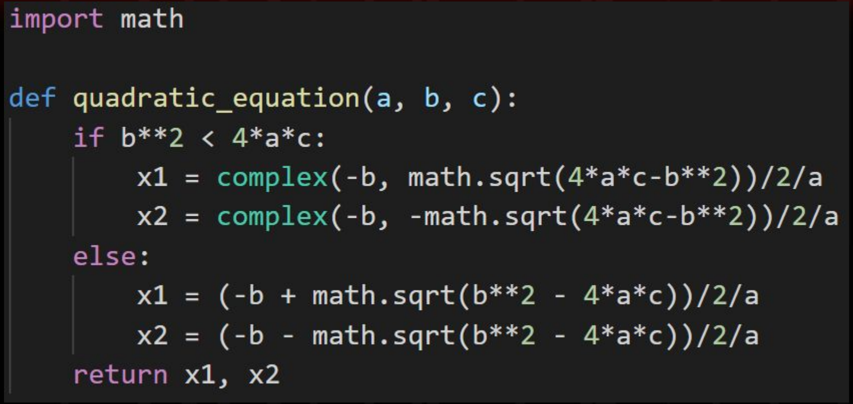
sin, cos, tan, etc.

exp, sqrt, dist, gcd, and others

ceil, floor

Special Constants: inf, nan, pi, and others

log, log10,



```python
import math

def quadratic_equation(a, b, c):
    if b**2 < 4*a*c:
        x1 = complex(-b, math.sqrt(4*a*c-b**2))/2/a
        x2 = complex(-b, -math.sqrt(4*a*c-b**2))/2/a
    else:
        x1 = (-b + math.sqrt(b**2 - 4*a*c))/2/a
        x2 = (-b - math.sqrt(b**2 - 4*a*c))/2/a
    return x1, x2
```

datetime

Most programs will make use of date and time at some point

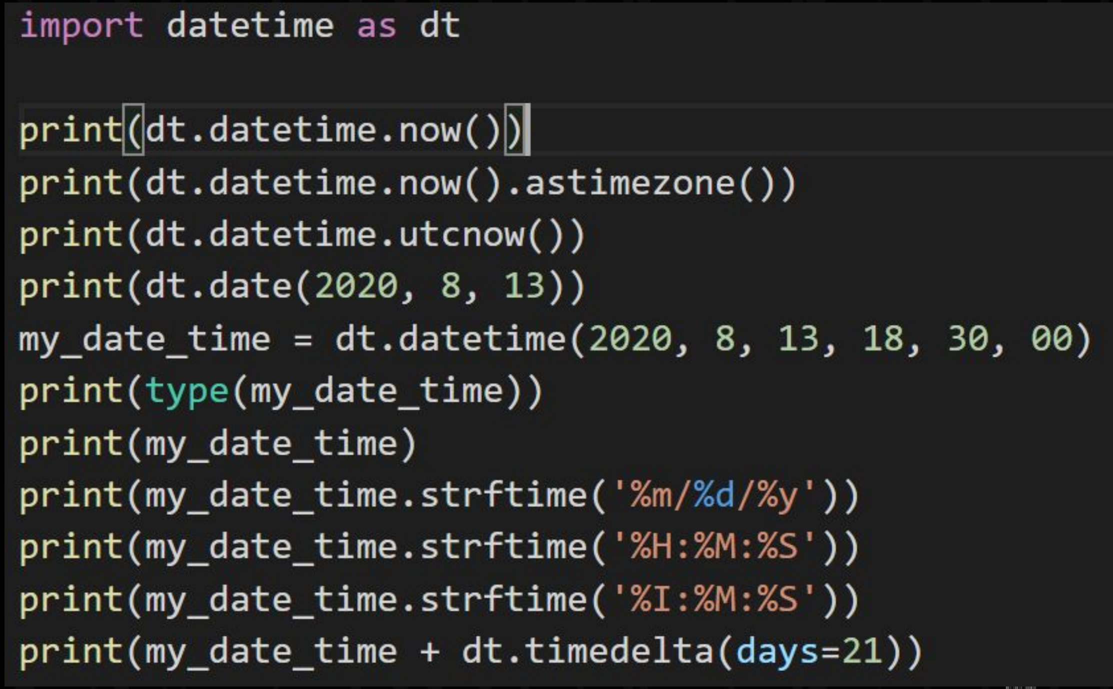
Use ‘now’ for current time

Timezones are tricky

import pytz to help

Server location?

Just use utc for everything!



```python
import datetime as dt

print(dt.datetime.now())
print(dt.datetime.now().astimezone())
print(dt.datetime.utcnow())
print(dt.date(2020, 8, 13))
my_date_time = dt.datetime(2020, 8, 13, 18, 30, 00)
print(type(my_date_time))
print(my_date_time)
print(my_date_time.strftime('%m/%d/%y'))
print(my_date_time.strftime('%H:%M:%S'))
print(my_date_time.strftime('%I:%M:%S'))
print(my_date_time + dt.timedelta(days=21))
```

Testing with unittest

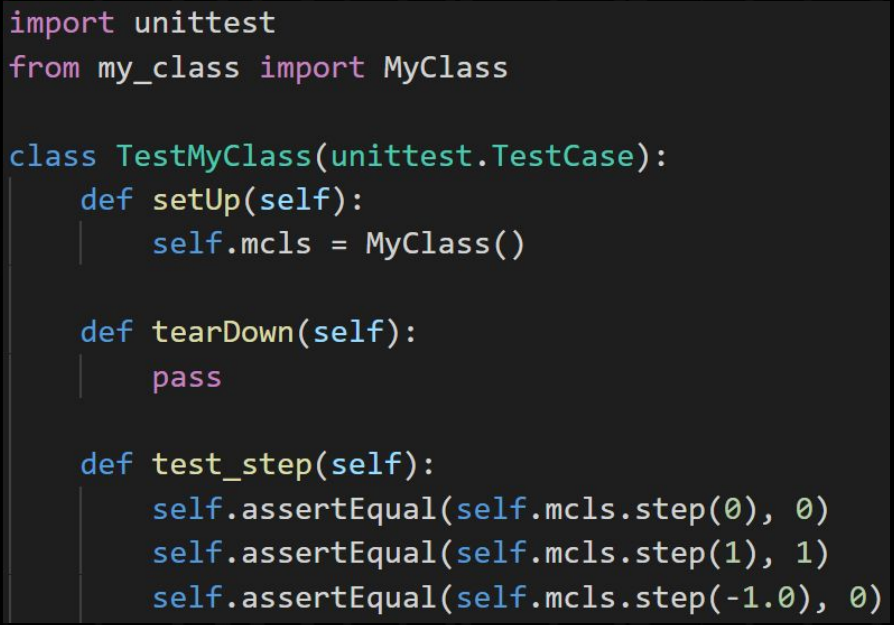
A program to test your program

Test edge cases

(where you expect it to break)

Consider Test-Driven Development

Also consider 3rd party pytest



```python
import unittest
from my_class import MyClass

class TestMyClass(unittest.TestCase):
    def setUp(self):
        self.mcls = MyClass()

    def tearDown(self):
        pass

    def test_step(self):
        self.assertEqual(self.mcls.step(0), 0)
        self.assertEqual(self.mcls.step(1), 1)
        self.assertEqual(self.mcls.step(-1.0), 0)
```

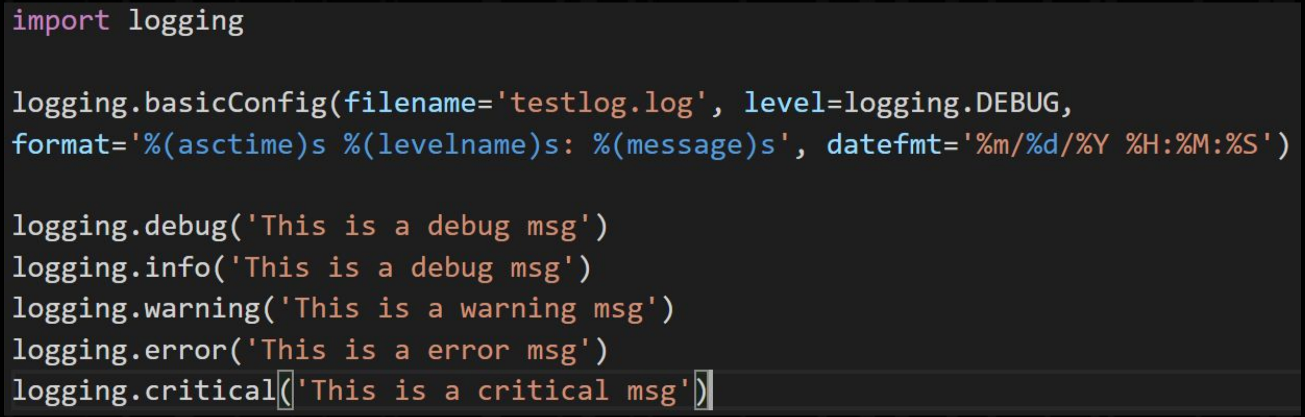
Logging

Levels:

- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

Output to command line or file

Optional formatting



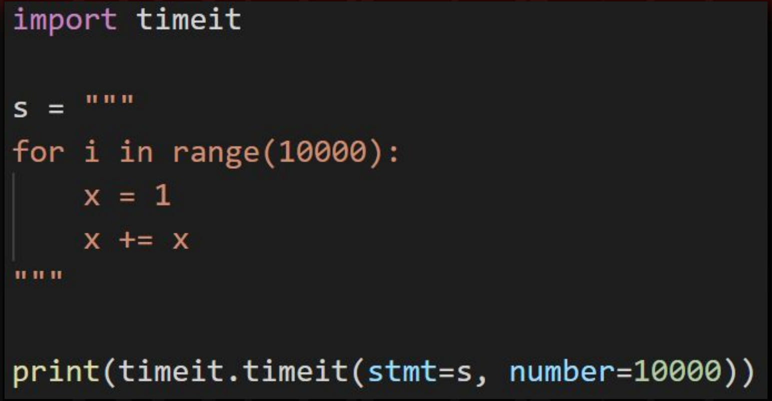
```python
import logging

logging.basicConfig(filename='testlog.log', level=logging.DEBUG,
format='%(asctime)s %(levelname)s: %(message)s', datefmt='%m/%d/%Y %H:%M:%S')

logging.debug('This is a debug msg')
logging.info('This is a debug msg')
logging.warning('This is a warning msg')
logging.error('This is a error msg')
logging.critical('This is a critical msg')
```

timeit

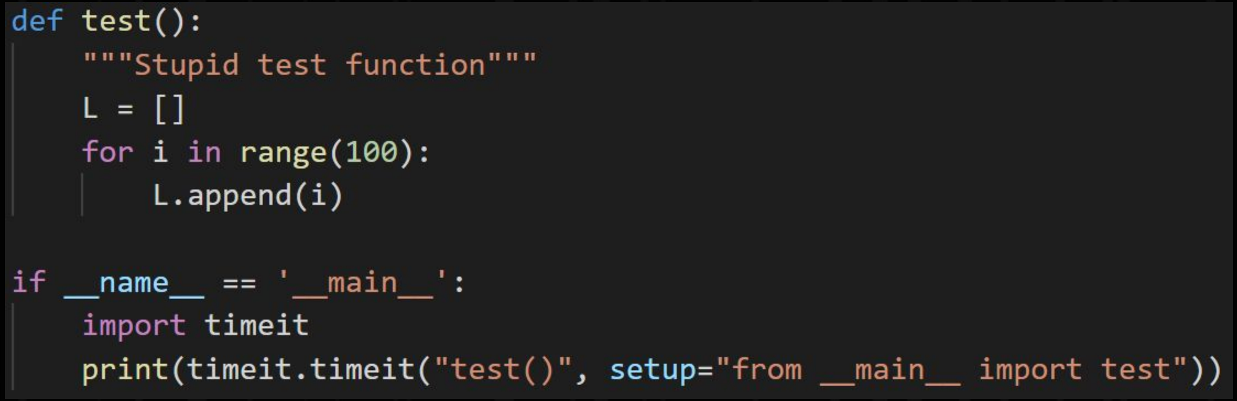
Robust way to time code

Executes many times and returns total

Leave debug mode!!



```python
import timeit

s = """
for i in range(10000):
    x = 1
    x += x
"""

print(timeit.timeit(stmt=s, number=10000))
```



```python
def test():
    """Stupid test function"""
    L = []
    for i in range(100):
        L.append(i)

if __name__ == '__main__':
    import timeit
    print(timeit.timeit("test()", setup="from __main__ import test"))
```

Debugging

You have already been using this alot!

Breakpoints

Step-over (F10)

Step-into (F11)

Run until next breakpoint (F5)

Why put it at the end?

Purest might consider the built-in library pdb for command-line debugging

What We Didn’t Cover (What’s Next?)

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

bytes, bytearray, and bitwise operators

Web Dev: django, flask, pyramid, many others

Data Science: jupyter notebooks, numpy, pandas, matplotlib, bokeh, …

Machine Learning: sci-kit learn, TensorFlow, PyTorch, etc.

GUI Dev: tkinter (builtin), Qt, kivy, …

Building, distributing python code: installer, etc.

∞. Conclusion