Beavertronics 5970 presents Programming Stuff

What will be covered

Command Line

- Changing dirs, listing files, Opening programs

Gvim/Gedit

- Moving around
- Modelines

Python Basics

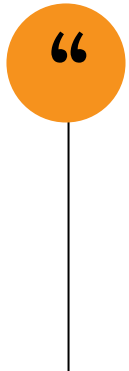
- Hello World
- Sequence, Iteration, Control
- Actual syntax: indentation, colons, parentheses

Object Oriented

- Attributes, Methods
- Instance vs Classes
- Scheduler structure, Inheritance

1 Command Line

“Linux is supreme”
-Dave Inman



Navigating the Filesystem

Directory: A place in the filesystem where data is stored. Also called folder.

Path: The directory you are within and everything above it

Search Path: A mega path which contains a sequence of directories; each one is searched to find executable programs. (ex: $PATH)

Commands

Navigating the Filesystem

cd

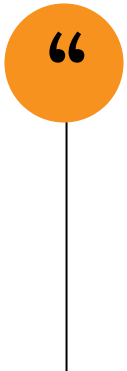
ls

mv/cp

rm

2 Editors

“Vim is supreme”
-Dave Inman



USE MOUSE LESS, KEYBOARD MORE

Moving Around in Vim

- Move with h, j, k, l -- like arrow keys but not
- h is left, l is right
- j is down, k is up

VIM IS MODAL

More Vim Commands: google drive

Editing in Vim

- Want to actually type?
  - Use i to enter insert mode
- Want to stop typing jklhjkkjjjjljlh in your code?
  - Use esc to enter visual mode
- Want to change things quickly?
  - Use : commands for line editing

DON'T MESS UP THE GITHUB

please

OUR MODELINE:
 # vim: set sw=4 noet ts=4 fileencoding=utf-8:

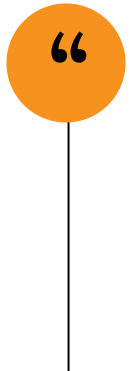
Vim Settings

- .vimrc
  - Overarching settings for vim files

- modelines
  - # vim: set ....
  - Can overwrite vimrc settings
  - USE OUR MODELINE OR DON'T PUT ON GITHUB

3 Python Basics

“Ruby is supreme”
-Dave Inman



Basic Usages

Variables/Attributes

```
var = [‘this is an array containing a string!’]
```

Functions/Methods

```
def func(args):
    do_something_here
```

Objects/Classes

```
class Everything():
    has attributes and methods and things
```

Hello World

Basic Usages

Print sends a message to the standard output (meaning the terminal)

Quotes indicate a string as opposed to a number or variable name

A common example of a function is the “Hello World” function

5970

Python Basics

Function Examples

Define your function!

```
def hello():
        print('Name?')
        in = input()
        print('Hello, {}')
                .format(in)
```

Call your func!

```
hello()
```

Get results!

In terminal:
python3 <file_name>

Things

For

for <word representing an individual> in <set of individuals>:

    do_something

While

while <condition>:

    do_something

If/Else

if <condition>:

    do_something

else:

    do_something_else

Modifiers: and, or, not

For Loop Example:

```
tree = [‘trunk’, ‘branches’, [‘twigs’, ‘leaves’]]

for main_component in tree:
        print(main_component)
        for small_thing in main_component:
        # Accesses other dimensions by looping
                print(small_thing)
```

trunk, branches, [‘twigs’, ‘leaves’]

t r u n k, b r a n c h e s, twigs, leaves

While loop example

```
import wpilib
def Robot():
        # returns True if the power is on
        def __init__(self):
                self.motor = ctre.WPI_VictorSPX(1)
        def move(self):
                while self.get_power_status()
                        self.motor.set(1)
# This is an instance of Robot
robot = Robot()
robot.move()
```

If/Else Example

```
import Robot
robot = Robot()
# This is a dict with keys paired to values as {key : value}
dio_ports = {arm_encoder : 1, pot : 2, limit_switch : 3}

if dio_ports[arm_encoder] == 1:
        robot.arm.cowabunga(dio_ports[arm_encoder])
else:
        Robot.yell_at_people_for_messing_with_ports()
```

By now you should have noticed...

Python requires indentation.

You'll figure it out.

4 Object Oriented Programming


“And on the fifth day God said
‘Let there be OOP’”
-Dave Inman

OOP: Objects

```
class Class():
    def __init__(self, attr):
        self.obj = attr
        # objects belonging to a class are attributes of that class
        self.object = ['list', 'with', 'strings', 'inside']
        self.object = 'string'
        self.object = {'key' : 'value'} # dict
    def instance_method(self):
        print("{} was passed in to this class".format(self.obj) +\
            " when it was made into an object!")
```

OOP: Method Calling

```
class Class():
    def __init__(self, attr):
        self.obj = attr
        # objects belonging to a class are attributes of that class
        self.object = ['list', 'with', 'strings', 'inside']
        self.object = 'string'
        self.object = {'key' : 'value'} # dict
    def instance_method(self):
        print("{} was passed in to this class".format(self.obj) +\
            " when it was made into an object!")
```

Making an instance:

```
import Class
obj = ['foo', 'bar']
new_class = Class(obj)
new_class.instance_method()
```

This yields in the std output:

['foo', 'bar'] was passed in to this class when it was made into an object!

OOP: Object Example

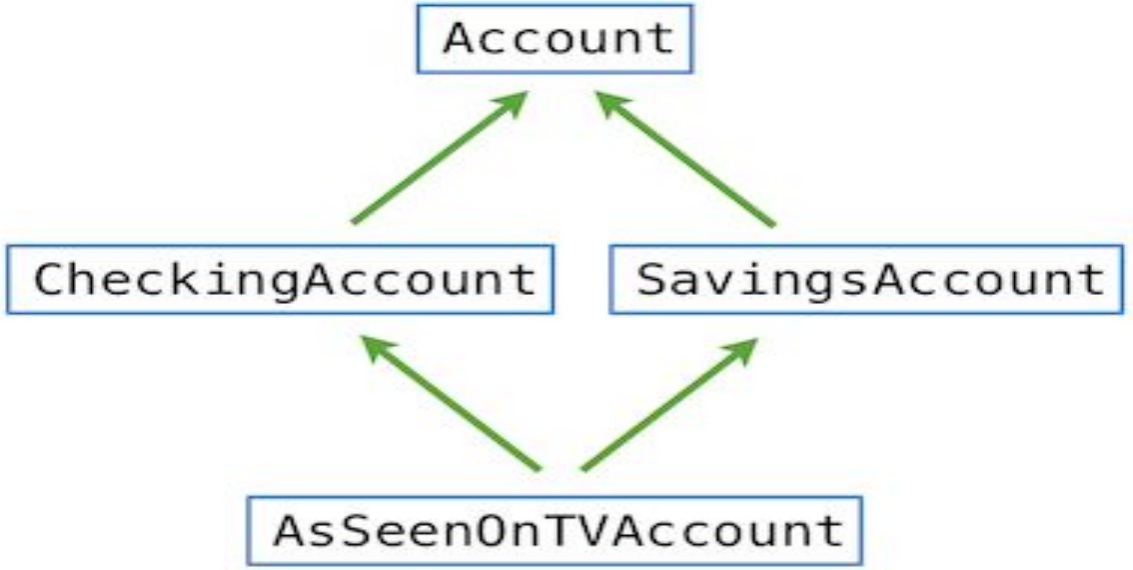
```
class Dewey():
    def __init__(self, postman_response):
        self.bark = postman_response
        self.tricks = [‘circus dog’, ‘yoga’, ‘come on snort’, ‘bang’]
        self.favorite_sound = ‘high pitched bark’
        self.good_boy = {‘Who’s a good boy?’ : self}
    def see_postman(self):
        print(“{}”.format(self.bark))


response = ‘yap yap yap’
dewey_instance = Dewey(response)
dewey_instance.see_postman()
```

OOP: Object Inheritance

Example:
import Account
class SavingsAccount(Account):
....

Our Usage:
from wpilib import TimedRobot
class Robot(wpilib.TimedRobot):
....



5970

Good job

Any questions ?

lollyinman@gmail.com

Or just ask me at robotics

https://github.com/lollyi/2019code5970