THE PLAN

- how to command line
    - ls
    - cd
    - pwd
    - mkdir
    - rm
    - touch
    - cp
    - mv
    - man
    - what is sudo
    - nano, gedit, vim, etc (command line text editors)
- do you have installed?
    - first run `python --version`
    - if no python/ python says not 2.7.xx
    - else run `sudo apt install python-minimal`
    - check for `pip --version`
    - run `sudo apt install python-pip`
- IDLE (Integrated Learing and Devlopment Enviroment)
    - shell where you can run all the python things
    - launch `python`
    - print statements
    - import this
    - '_' is last thing
    - exit()
    - Documentation:
        - https://docs.python.org/2.7/library/index.htm
- Notes
    - everything in python is a object
    - dynamically typed
- Variable stuff
    - `x=1
- Strins
    - at ever you put in this to a string
    - putting strings together with `str1 + str2`
    - individual values with `str[0]`
    - to find length `len(str)`
    - slicing with `str[0:3]`, `str[:]`, `str[0:-2]`, `str[:4]`
    - many more useful funcl
- Lists
    - mutable can change insides after assignments
    - `l = []` or `l = list()`

    - can also add values during creations `l = [1,2,3]`
    - can mix values `l = [1,'hello', []]`
    - access elements with `l[i]`
    - slice with `l[0:3]` all slicing rules from the end apply
    - gets length of list `len(l)`
    - cat lists together with `l1+l2`
    - sort with `l.sort()` keep in mind .sort() occurs in place and does not return anything to return something use `sorted()`,
    - use `l.sort(key = func, revere = bool)` for example pass in len to sort by the len function
    - `set()`
    - many useful func
- Tuples
    - immutable
    - tuples are faster than lists
    - cant change length or change internals
    - used when data should not change
    - make with `t = (1,2,3,4,5)` or `t= tuple(1,2,3,4,5)`
- Dictionaries
    - mutable
- basically a hashmap
    - uses key:value
- `d = {}` or `d = dict()`
    - `dict = {"one":1, "two":2, "three":3}`
- access a element using `d[key]`
    - can set a element above with `d[key]=value`
- get a list of keys with `d.keys()`
    - get a list of values with `d.values()`
    - get a list of tuples with `d.items()`
- Conversion
    - wrapping an object in either `str(),list(),tuple(),dict(),int(),float()` will attempt to convert that object into that data type
    - `type()`
- main.py file
    - python is interpreted line by and is not complied
    - Running a script `python main.py`
- Conditionals

```
- `if, elif, else`
- boolean operators
- `is, and, not, in, ==, >=, <= , != `
- talk about indentation here
```

- loops

````
- `range(start,end,step)`
- `for i in loop`
- `for k,v in dict.items()`
- `while`
````

- Comprehensions

````
- haha list comprehension go brr
- `[x.strip() for x in l]`
- `{x:x*10 for x in l}`
````

- Exceptions

~~~
- ```python
    try:
        f()
    except:
        print(oops)
    ```

- ```python
    try:
        f()
    except ValueError:
        print(e)
    ```

- else,finally exist look at google
~~~

- Files

````
- Reading a file
````

~~~
    - `file = open(f,'r')`

    - ```
        for line in file
            print(line)
        file.close()
        ```

    - you can read and iterate through a file in all sorts of different ways

- Writing a file
~~~

~~~
    - ```
        file = open(filename, 'w') # this will overwrite the file even if it
exists
        file.write('some text')
        file.close()
        ```

- There are other ways to open files use google
~~~

- Functions

~~~~
- ```
    def add(a,b):
        return a+b
    ```

- ```
    def subtract(a = 10, b = 1)
        return a-b
    ```

- function must be defined before its used

- trick by putting main at top and running it at bottom

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

- using functions
~~~~

~~~~
    - ```
        add(1,3)
        subtract(3)
        subtract(b=2)
        subtract(b=1,a=3)
        ```

    - scope
~~~~

- Classes

~~~~
- ```
    class Person:
      def __init__(self, name, age):
        self.name = name
        self.age = age

      def greet(self):
        print("Hello my name is " + self.name)

    p1 = Person("John", 36)
    p1.greet()
    ```

- use `dir()` to learn about a class

- you can over write all sorts of inbuilt methods in python, makes your classes
play nice with pythonic syntax
~~~~

- Modules

~~~
- ```
    import math
    from math import sqrt
    import math as m
    from math import sqrt,tau
    from math import *

    #careful of overwirting stuff
    # example
    sqrt= 5
    sqrt()
    ```
~~~