input(“text”) returns the value (string) input by the user
int(string) converts strings to int
Same as float(), bool(), and str()
“String” in variable tells if the string is part of the variable string
+, -, *, /, // gives int, ** is power to
And and or are used in comparisons of booleans n stuff
If bool:
    Statement
Elif bool2:
    Statement
Else:
    Statement executed if the above 2 are not
# starts a comment
Name = [] is a list
For variable in list:
    Statement
range (#, #) makes a list of numbers starting at first number and ending 1 before second
Tuples are lists that cant be changed, they are made w () instead of []

Def func(input)
    Statement
Thats a function, it takes an object and applies smt to it

Methods take objects and do smt with it

class name():
    X = variable #this is a class variable
    def __init__(self, name, age): #necessary for all classes, basically runs when a obj of class is made and self is basically always needed  btu doesnt need to be put in when making an object
        self.name = name #sets the name
        self.age = age
    def speak(self):
        print(“hello my name is “+self.name)

#inheritance is like this
class name(other class): #the otherclass is the parent class and the child inherits the stuff
    def __init__(self, name, age, colour)
        super().__init__(name, age)
        Self.colour = colour
    def speak(self):
        print(“meow”)

Subclass inherits the attributes and methods of superclass. Superclass can not use subclass methods
Subclasses cna also overwrite (basically) superclass methods so they dont have to do the exact same thing
Classes can have variables that are only available to what is within the class

@classmethod pass cls into the () and can access class variables
@staticmethod basically is a function that doesnt need an object (just classname.funcname()) but is housed in a class

Python cant make public/private stuff so if you want it to act as a private method, put _ right before their name

To import other files do
Import filename
From filename import classname

**Dictionaries**
Similar to sets where they are a list or collection of UNIQUE items

They are also unordered

Keys can be many variables like str, list, tuple,

Made like
Dictionary = {‘key’:value,...}

Access dictionaries with
Dictionary[‘key’]

Add to dictionary with
Dictionary[key] = value

dictionary.values() gives the values in the dictionary

dictionary.items() gives a list of the key and item as a tuple

Modifying keys is easy with just
Dictionary[key] = value
It will change the current value if its in there, if not, it will add the new key of value

dictionary.get(key) returns the keys value
If not in dictionary itll return “none” unless specified with get(key, returnThisIfMissing)

Clear dictionaries with
dictionary.clear()

Delete keys with
Del dictionary[key]
Or
dictionary.pop(key)
Or
dictionary.popitem() (pops last item in dictionary

Copy a dictionary with
Newdictionary = dictionary.copy()
Keep in mind, the items still have the same address so changing one changes both

dictionary.setdefault(key, valueifnotvalue) returns the value, and if it doesnt exist, itll make a key w the specified value

dictionary.update(key:vlaue) will add it to the dictionary, note if it has clashing keys, the newer one wins

**Testing**
For our testing it seems we are using pytest, so i will look into some videos on that here

Test files should be calles filename_test or test_filename

Can be kept separate from main branch

In the test file, test methods/functions like so
test_funcname():
    assertclassname.funcname() == whatever

Basically use assert statements to ensure the functions do what they should. If there is more to this, i shall re-open this