#MyPythonDocumentation

The pop() method, remove the last item from your list, but it let you work with it after removing it.

% the remainder operator or integer remainder operator or the Modulo Operator works on integers, it is a useful tool for working with numerical information which divides one number by another and return the remainder. The modulo operator doesnt tell you how many times one number fits into another; it just tells you what the remainder is.

format() is one of the string formatting methods in Python3, which allows multiple substitutions and value formatting. This method lets us concatenate elements within a string through positional formatting.

for copying we cannot put two lists equal to one other, because this syntax tells Python to connect the new variable friends\_foods to the list that is already contained in the first list, so now both variable in both lists are already contained in the second and the first list, and the output shows both lists the same which is not the definition of copying.

We use tuples when we don't want a list to get changed. we will use parentheses instead of squared brackets.

When you want to determine whether two values are not equal, you can combine an exclamation point and a equal sign(!=). The exclamation point represents not, as it does in many programming languages.

“and” To check whether two conditions are both True simultaneously, use the key word and to combine the two conditional tests; if each test passes, the over all expression evluates to False. The important point about “and” is: we use and to check multiple conditions.

Like and, we use “in” to find out whether a particular value is already in a list.

A Boolean expression is just another name for conditional test.A Boolean value is either True or False, just like the value of a conditional expression after it has been evaluated. Boolean values provide an efficient way to track the state of a program or a particular condition that is important in your program.

The if-elif-else Chain: Often, you’ll need to test more than two possible situations. Many real- world situations involve more than two possible conditions.

The elif line is really another if test, which runs only if the previous test failed.

* The output is the same as if else, but the purpose of the if-elif-else chain is narrower.
* You can use as many elif blocks in your codes as you like.
* Python does not require an else block at the end of an if-elif chain. Sometimes an else block is useful; sometimes it is clearer to use an additional elif statement that catches the specific condition of interest.

Sometimes it is important to check all of the conditions of interest. In this case, you should use a series of simple if statements with no elif or else blocks.

In if statements pay attention to or and and. Examples are in the 3th page of codes.

Checking that a list is empty: When the name of a list is used in an if statement, Python returns true if at list contains one item, if the conditional test fails, we print a message asking the customer if they really want a plain pizza.

Importan point about indenting: if in for loop else part does not locate beneth if the for loop wont work.

In dictionaries deleted key-value pair is removed permanently.

In dictionaries python doesn’t care about the order in which you store each key-value pair; it cares only about the connection between each key and its value.

Typically, you’ll use empty dictionaries when storing user-supplied data in a dictionary or when you write code that generates a large number of key-valye pairs automatically.

To modify a value in dictionary, give the name of the dictionary with the key in square brackets and then the new value you want associated with the key.

* Ex: alien\_0[‘color’] = ‘yellow’

Dictionary is useful for storing the results of simple polls.

You can loop through all of a dictionary’s key-value pairs, through its keys or through its values.

If you want to see everything stored in a dictionary you can loop through the dictionary using a for loop.

The method items(), will return a list of key-value pairs.

Notice that the key-value pairs are not returned in the order on which they were stored, even when looping through a dictionary. Python doesn’t care about the order in which key-value pairs are stored; it tracks only the connections between individual keys and their values. Variables name and value will make it easier to follow what’s happening inside the loop.

Looping through the keys ia actually the default behavior when looping through a dictionary, so you omit .keys() from the end. SO!! You can access the value associated the names in a dictionary and when the name matches of the list (that we wrote it, we’ll can have the value related to it.

The Key() method is not just for looping: It actually returns a list of all the keys.

If you are primarily interested in the values that a dictionary contains, you can use a values() method to return a list of values without any keys

Sorted() function is using for to get a copy of the keys in order.

* Ex: for name in favorite\_languages.keys()):

This for statement is like other for statements except that we’ve wrapped the sorted()function around the dictionary.keys() method.

To see variables without repetition, we can use a set. A set is similar to a list except that each item in the set must be unique. When you wrap a set() around a list that contains duplicate items, Python identifies the unique itmes in the list and builds a set from those items.

Nesting: To store a set of dictionaries in a list or a list of items as a value in a dictionary.

What is Ellipsis in Python?

It’s common to store a number of dictionaries in a list when each dictionary contains many kind of information about one object. All of the dictionaries in the list should have an identical structure so you can loop through the list and work with each dictionary object in the same way.

Sometimes it is useful to put a list inside a dictionary.

We can nest a list inside a dictionary any time you want more than one value to be associated with a single key in a dictionary.

In programing prompt is used to literally "prompt" the user to answer something or enter information that you are asking for.

The input() function takes one argument*: the prompt* or instructions, that we want to display to the user so they know what to do.

You can store your prompt in a variable and pass that variable to the input() function. This allows you to build your prompt over several lines, then write a clean input() statement.

+= this operator takes the string that was stored in prompt and adds the new string onto the end: prompt +=”\nWhat is your first name?” Page: 119

The += operator is shorthand for current\_number = current\_number + 1. Page: 122

When you use input() function, Python interprets everything the user enters as a string.

The for loop takes a collection of items and executes a block of code once for each item in the collections. In contrast, the while loop runs as long as, or while, a certain condition is true.

While loop: The for loop takes a collection of items and executes a block of code once for each item in the collection. In contrast, the while loop runs as long as, or while, a certain condition is true.

For the program that should run only as long as many conditions are true, you can define one variable that determines whether or not the entire program is active. This variable, called a flag, act as a signal to the program.

* We can write a program to run while the flag is set to TRUE and stop when any of several events sets the value of the flag to FALSE.
* As a result, our overall while statement need to check only one condition: whether or not the flag is currently True.
* Then, all our other tests (to see if an event has occured that should set the flag to FALSE) can be neatly organized in the rest of the program.

To exit a while loop immediately without running any remaining code in the loop, regardless of the results of any conditional test, use the break statement. The *break* statement directs the flow of your program; you can use it to control which lines of code are executed and which aren’t, so the program only executes code that you want it to, when you want it to.

A loop that starts with while True will run forever unless it reaches a break statement.

If your program gets stuck in an infinite loops, press CTRL-C or just close the terminal window.

Rul of thumb: codes for age:

* Age = input(“How old are you?”)

How old are you?

Age = int(age)

Important point: A for loop is effective for looping through a list, but you shouldn’t modify a list inside a for loop because Python will have trouble keeping track of the items inside a list. SO:

* To modify a list as you work through it, use a while loop. Using a while loop with lists and dictionaries allows you to collect, store, and organize lots of input to examine and report later.

You can prompt for as much input as you need in each pass through a while loop.

When you want to perform a particular task that you’ve defined in a function, you call the name of the function which is responsible for it.

If you need to perform one task multiple times throughout your program, you just call that function dedicated to handling that task, and the call tells Python to run the code inside the function.

In function the first line is telling Python that you are defining a function: function definition. In the parantheses we will have the info we need.

The second line is docstring, which describes what the function does.

The third line (print line) is the only line of actual code.

The last line is calling the function, by the name that we define initially and the info that we put in the parenthesis before.

The value in the parentheses is called an argument. An argument is a peice of info that is passed from a function call to a function. When we call the function, we place the value we want the function to work with in parentheses.

Positional Argument, in positional arguments order matters. You can use as many positional arguments as you need in your functions. Python works through the arguments you provide when calling function and matches each one with the coresponding parameter in the function’s definition.

Keyword Argument, is a name-value pair that you pass to a function. You directly associate the name and the value within the argumen, so when you pass the argument to the function, there’s no confusion. They free you from having worry about correctly ordering your argument in the function call, and they clarify the role of each value in the functions call.

* In Keyword argument, the function definition won’t change but when we call the function we explicitly tell Python each argument should be matched with.
* Order doesn’t matter in Keyword Argument.

Calling a function several times is a very efficient way to work (like in positional arguments, example animal\_type & pet\_name)

Default value. If an argument for a parameter is provided in the function call, Python uses the argument value. If not, it uses the parameter’s default value.

* So when you define a default value for a parameter, you can exclude the corresponding argument you’d usually write in the functional call.
* Important Point: When you use default value, any parameter with a default value needs to be listed after all the parametes that don’t have default values. This allows Python to continue interpreting positional arguments correctly.

The function doesn’t always have to display its output directly. Instead, it can process some data and then return a value or a set of values. SO...

* The value the function returns is called a return value.
* Return values allow you to move much of your program’s grunt work into the functions, which can simplify the body of your program.
* Return Statement takes a value from inside a function and sends it back to the line that called the function. The return statement takes a value from inside a function and sends it back to the line that called the function.

To make an argument optional, we can give it an empty default value and ignore the argument unless the user provides a value. Optional values allow functions to handle a wide range of use cases while letting function calls remain as simple as possible.

Artbitrary function: The astrick in the parameter name (\*) tells Python to make an empty tuple called (anything) to make an empty tuple called (that same anything) and pack whatever values it receives into this tuple.

If you want a function to accept several different kinds of arguments, the parameter that accepts an arbitrary number of arguments must be placed last in the function definition.

* First Python matches the positional and keyword arguments and then collects any remaining arguments in the final parameter. Meaning: if the function needs to take a size of the pizza, that parameter must come before the parameter \*toppings.

Reminder slice notation [:] makes a copy of the list.

In *Using Arbitrary Keyword Arguments* double asterisks before the parameter will cause Python to create an empty dictionary and pack whatever key-value pairs it recieves into this dictionary. Within the function, you can access the key-value pairs in the related parameter just as you would for any dictiolnary.

Moduel and import statement:

* Moduele is the file ending in .py that contains the code you want to import into yourprogram.
* We can store our functions in a seprate file called a module and then importing that module into your program. An import statement tells Python to make the code in a module available in the currently running program file.
* Storing the function in module will let us hide the details of our program’s code and focus on its higher-level logic.
* It also allows us to reuse functions in many different programs.
* When you store the functions in seprated files, you can share those files with other programmers without haviing to share your entire program.
* Knowing how to import functions also alows you to use libraries of functions that other programmers have written.
* To call a function from an imported module, enter the name of the module you imported followed by the name of the function, seprated by a dot.

EX: module\_name.function\_name(). Please be aware that this approach to importing will make every function from the module available in your program.

Importing Specific functions

* EX: from module\_name import function\_name
* You can import as many functions as you want from a module by seprating each function’s name with comma:
* EX: from module\_name import funciton\_0, function\_1, function\_2

Alias is an alternate name similar to a nickname for the function, you’ll give the function this special nickname when you import the function.

* EX: from modume\_name import function\_name as fn

We can also provide an alias for a module name. Ex: import module\_name as mn

Importing All Funcitons in a Module

You can tell Python to import every function in a module by using the asterick (\*) operatore. The asterisk in the import statement tells Python to copy every function from the module into program file, and since all the functions are imported, you can call each function by name without dot notation.

* EX: from module\_name import \*

Instantiation is making an object from a class. We work with instances of a class.

I.P: By convention, capitalized names refer to classes in Python.

I.P: The parentheses in the class definition are empty when!! We want to create the class from scratch.

I.P: What is method? Method is a function that is a part of a class. So!! Everything we learned about functions applies to method as well.

I.P: the \_\_init\_\_() method is a special method Python runs automatically whenever we create a new instance based on the Dog class.

* Those two leading and trailing underscores, they are convention to help prevent Python’s default method names from conflicting with your method names.
* When we make an instance of a class, Python will call the –init—() method from the Dog class.
* Python needs \_\_init\_\_() method to creat an instance from the class.
* The –-init—-() method creates an instance representing one particular object and set the related attributes using the values we provided.
* The –-init—-() method has no explicit return statement, but Python automatically returns an instance representing that.
* Whenever we want to make an instance from the Dog class, we’ll provide values for only the last two parameters.
* The only practical difference for now is the way we’ll call methods.
* What is attribute? Variables that are accessible through instances are attributes.

Dot notation is used often in Python. This syntax demonstrates how Python finds an attribute’s value.

You can create as many instances from one class as you need, as long as you give each instance a unique name or it occupies a unique spot in a list or dictionary.

Reminder! Class will tell Python how to make individual instances.

Reminder! Capitalized name: Class. Lowecase name: instance created from a class.

Reminder!

* Attributes are like: *self.parameter*
* Attributes takes the values stored in the paramete name and stores it in the variable name, which then attached to the instance being created.
* So!! The variables that are accessible through instances like this are called attributes.

Every attribute in a class needs an initial value, even if that value is 0 or an empty string. In some cases, such as when setting a default value, it makes sense to specify this initial value in the body of the \_\_init\_\_() method; if you do this for an attribute, you don’t have include a parameter for that attribute: self.sth = 0

We can change an attribute value in three ways:

* Change the value directly through an instance. (The simplest way to modify the value of an attribute is to access the attribute through an instance. We will use dot notation to access the attribute and set the value directly)
* Set the value through a method.(Here instead of accessing the attribute directly, you pass the new value to a method that handles the updating internally.)
* Increment the value through a method. (To increment an attribute’s value by a certain amount rather than set an entirely new value.)
* Note We can use methods like this to control how users of your program update values such as an odometer readingm but anyone with access to the program can set the odometer reading to any value by accessing the attribute directly. Effective security takes extreme attention to detail in addition ro basic checks like those shown here.

Inheritance: If the class you’re writing is a specialized version of another class you wrote, we can use inhertance.

If a class inherits from another, it automatically takes on all the attributes and methods of the first class. The original class is called the parent class, and the new class is the child class. The child will have all the attributes and method but also it is free to define new attributes and methods of its own.

* So!!! In child class you can add any new attributes and methods necessary to differentiate the child class from parent class.
* I.P: When you create a child class, the parent class must be part of the current file and must appear before the child class in the file.

The super(). function is a special function that helps Python make connection between the parent and child class. This function tells Python to call the \_\_init

--() method from the parent class, which gives the child instance all the attributes of its parent class.

- The name *super* comes from a convention of calling the parent class a superclass and the child class a *subclass*.

To override any method from the parent class that doesn’t fit what you’re trying to model with the child class, with defining a method in the child class with the same name as the method you want to override in the parent class. Python will disregard the parent class method and only pay attention to the method you define in the child class.

* So!!! When you use inheritance, you can make your child classes retain what you need and override anything you don’t need from the parent class.

IMPORTANT POINT(USEFUL FOR FIRST EXERCISE IN PAGE: 177)

* When you setting a default value in the body of the \_\_init\_\_() method; if you do this for an attribute, you don’t have to include a parameter for that attribute.

Modules are to store classes in and then import the classes into your main program.

* When you move the class to a module and import the module, you still get all the same functionality, but you keep your main program file clean and easy to read.
* You also keep the logic in a seperate files; once your classes work as you want them to, you can leave those files alone and focus on the higher-level logic of your main program.
* You can store as many classes as you need in a single module, although each class in a module should be related somehow.

Exceptions are special objects to handel errors while a program is running.

You can import as many classes as you need into a program file.

* You import multiple classes from a module by seprating each class with a comma.
* Once you’ve imported the necessary classes, you’re free to make as many instances of each class as you need.

You can also import an entire module and then access the classes you need using dot notation.

And you can import all the classes in a module, but it is not recommended, here some reasons why?

* It’s helpful to be able to read the import statement at the top of the file and get a clear sense of which classes a program uses, and with this approach it’s unclear.
* This approach can also lead to confusion with names in the file, because if you accidently import a class with the same name as somthing else in your program file, you can create errors that are hard to diagnose.

The *Python standard Library* is a set of modules included with every Pyhton installation.

* You can use any function or class in the standard library by including a simple import statement at the top of your file.
* EX: If you are creating a dictionary and want to keep track of the order in which key-value pairs are added, you can use the OrderedDict class from collections module. (Instances of OrderdDict class behave almost exactly like dictionaries except they keep track of the order in which key-value pairs are added.

Class names should be writen: CamelCaps and without underscores

* Every class should have docstring immediately following the class definition.
* Whithin a class you can use one blank line between methods

Instances and models should be written in lowercase with underscores between words.

* Each module should also have a docstring describing what the classes in a module can be used for.
* Within a module you can use two blank lines to seprate classes.
* Important Point: If you need to import a module from the standard library and a module that you wrote, place the import statement for the standard library module first. Then add a blank line and the import statement for the module you wrote. In programs with multiple import statements, this convention makes it easier to see where the different modules used in the program come from.

Json modules allows us to save user data so it isn’t lost when your program stops running.

When you want to work with the information in a text file, the first step is to read the file into memory.

Intersting point: read() returns an empty string when it reaches the end of the file and that empty string shows up as a blank line.

* Also print statement add a newline each time we call it.

*Relative file path* will tell Python to look for a given location relative to the directory where the currently running program file is stored.

*Absolute file* path will tell Python where exactly the file is on the computer.

With syntax will tell Python to open and close properly.

Readlines() method takes each line from the file and stores in a list. This list is then stored in lines, which we can continue to work with after with block ends.

When Python read from a text file, it interprets all text in the fle as a string. If you read in a number and want to work with that value in a numerical context, you’ll have to convert it to an integer using the int() function or convert it to a float using the float() function.

To write text to a file, you need to call open() with a second argument telling Python that you want to write to file.

* You can open a file in write mode(‘w’), read mode (‘r’), or append mode (‘a’), or a mode that allows you to write and read the file (‘r+’).
* If you omit the mode argument, Python opens the file in read-only mode by default.

Important Point: The open() function automatically creates the file you’re writning to if it doesn’t already exist.

* BE CAREFULL opening a file in write mode if it does exist, Python will erase the file before returning the file object. So!! If you want to add content to a file instead of writing over existing content, you can open the file in *append mode*, and Python won’t erase what you already have there.

Whenever an error occurs that makes Python unsure what to do next, it creats an exception object.

* If you write code that handles exception, the program will continue runninng.
* If you don’t handle the exception, the program will halt and show a traceback, which includes a report of the exception that was raised.
* Exceptions are handled with try-except blocks.

A simple way to store the data that user entered is using the json module.

* The json module allows you to dump simple Python data structres into a file and load the data from that file the next time the program runs.
* You can also use json to share data between different Python programs.

Refactoring is The process of improve the code by breaking it up into a series of functions that have specific jobs.

* Refactoring makes your code cleaner, easier to understand, and easier to extend.
* Removing the bulk of its logic into one or more functions.

Important point: A function should return a value that we expecting, or it should return None.

A Unit test verifies that one specific aspect of a function’s behavior is correct.

A *test case* is a collection of unit tests that together prove that a function behaves as it’s supposed to, within the full range of situations you expect it to handle.

* A good test case considers all the possible kinds of input a function could receive and includes tests to represent each of these situiations.
* It’s often good enough to write tests for your code’s critical behaviors and then aim for full coverage only if the project starts to see widespread use.
* How to write a test case for a function?
  + Import the unittest module and the function you want to test.
  + Create a class that inherits from the unittest.
  + TestCase, and write a series of methods to test different aspects of your function’s behavior.

You can write test for a class. If you have passing tests for a class you’re working on, you can be confident that improvements you make to the class won’t accidently break its current behavior.

Reminder: Assert methods test whether a condition you believe is true at a specific point in your code is indeed true.

Important point: Why setUp() method is usefull? When you’re testing your own classes, the setUp() method can make your methods easier to write. You make one set of instances and attributes in setUp() and then use these instances in all your test methods. This is much easier than making a new set of instances and attributes in each test method.

What is *screen*? We create a display window by calling pygame.display.set\_mode().

*Surface* in Pygame is a part of the screen where you display a game element.

* Each element in the game, like the aliens or the ship, is a surface.

What is an *event*? It is an action that the user performs while playing the game, such as pressing a key or moving the mouse.

* To make our program respond to *events*, we’ll write an event loop to listen for an event and perform an appropriate task depending on the kind of event that occured.

Colors in Pygame are specified as RGB colors: a mix of red, green, blue. Each color value can range from 0 to 225.Red is: (255,0,0). Green is: (0,255,0). Blue is: (0,0,255)

Responding to a KeyPress: Whenever the player presses a key, that keypress is registered in Pygame as an event.

* Each event is picked up by the pygame.event.get() method.
* In Our check\_events() function what kind of events to check for.
* Each keypress is registered as a KEYDOWN event.
* When a KEYDOWN event is detected, we need to check whether the key that was pressed is on that triggers a certain event.
* If the right arrow key is pressed, we increase the ship’s rect.centerx value to move the ship to the right.
* Continuous motion: We’ll have our game detect a pygame.KEYUP event so we’ll know when the right arrow key is released; then we’ll use the KEYDOWN and KEYUP events together with a flag called moving\_right to implement contiuous motion.
* When the ship is motionless the moving\_right flag will be False.
* When the right arrow key is pressed we’ll set the flag to True, and when it’s released we’ll set the flag to False again.
* The Ship class controls all attributes of the ship, so we’ll give it an attribute called moving\_right flag.
* The update() method will change the position of the ship, so we’ll give it an attrubute called moving right and update() method to check the stats of the moving\_right flag.
* The update() method will change the position of the ship if the flag is set to True.
* We’ll call this method every time we want to change the position of the ship.

The Sprite class is intended to be used as a base class for the different types of objects in the game. There is also a base Group class that simply stores sprites. A game could create new types of Group classes that operate on specially customized Sprite instances they contain.