#MyPythonDocumentation

1. The pop() method, remove the last item from your list, but it let you work with it after removing it.
2. % 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.
3. 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.
4. 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.
5. We use tuples when we don't want a list to get changed. we will use parentheses instead of squared brackets.
6. 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.
7. “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.
8. Like and, we use “in” to find out whether a particular value is already in a list.
9. 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.
10. 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.
11. 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.

1. 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.
2. In if statements pay attention to or and and. Examples are in the 3th page of codes.
3. 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.
4. Importan point about indenting: if in for loop else part does not locate beneth if the for loop wont work.
5. In dictionaries deleted key-value pair is removed permanently.
6. 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.
7. 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.
8. 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’

1. Dictionary is useful for storing the results of simple polls.
2. You can loop through all of a dictionary’s key-value pairs, through its keys or through its values.
3. If you want to see everything stored in a dictionary you can loop through the dictionary using a for loop.
4. The method items(), will return a list of key-value pairs.
5. 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.
6. 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.
7. The Key() method is not just for looping: It actually returns a list of all the keys.
8. 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
9. 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.

1. 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.
2. Nesting: To store a set of dictionaries in a list or a list of items as a value in a dictionary.
3. What is Ellipsis in Python?
4. 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.
5. Sometimes it is useful to put a list inside a dictionary.
6. 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.
7. In programing prompt is used to literally "prompt" the user to answer something or enter information that you are asking for.
8. The input() function takes one argument*: the prompt* or instructions, that we want to display to the user so they know what to do.
9. 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.
10. += 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
11. The += operator is shorthand for current\_number = current\_number + 1. Page: 122
12. When you use input() function, Python interprets everything the user enters as a string.
13. 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.
14. 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.
15. 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.

1. 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.
2. A loop that starts with while True will run forever unless it reaches a break statement.
3. If your program gets stuck in an infinite loops, press CTRL-C or just close the terminal window.
4. Rul of thumb: codes for age:

* Age = input(“How old are you?”)

How old are you?

Age = int(age)

1. 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.

1. You can prompt for as much input as you need in each pass through a while loop.
2. 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.
3. 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.
4. 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.
5. The second line is docstring, which describes what the function does.
6. The third line (print line) is the only line of actual code.
7. The last line is calling the function, by the name that we define initially and the info that we put in the parenthesis before.
8. 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.
9. 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.
10. 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.

1. Calling a function several times is a very efficient way to work (like in positional arguments, example animal\_type & pet\_name)
2. 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.

1. 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 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.

1. 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.