# Database Basics

SQL is a program for storing data? You can take tables and join them in different way (inner joins, equi-joins, outer joins).

Relational vs Non-relational databases are different places where data is stored and how it’s stored and manipulated.

**Is the main thing I need to know about relational databases and non-relational databases is that they are 2 ways to store data, depending on what you need to do in your company?**

* **I don’t really understand the difference because I don’t really understand what it’s applications are. They talk about transactions a lot for relational databases, so would they store credit cards because they’re faster and more reliable but can’t store variable types of information, which is where NoSQL databases come in? Is that all I should know?**
* **Should I know how to code SQL? Or will I eventually learn how to interact with these databases?**

# Data Frames and Series

Are data frames and series just ways to make tables in pandas?

# Extract, Transform, and Load

Is this just what happens when you pull data from a database so the end use can make a decision? Like facebook would extract and transform data about your profile and load it on your screen. Or amazon would extract and transform your credit card information from their database to load it on your screen so you can use it?

# JSON, XML, CSV

JSON (JavaScript Object Notation) stores data in text, using C-family language structure

* Usually a text representation of dictionaries/lists/etc.

XML (external markup language) is human-readable and machine-readable code for “marking up” text (e.g. HTML)

CSV (comma-separated values) is a file that stores data in tables using commas in the language

* Usually text representation of rows and columns

TXT

* Just text

# Regular Expressions

Is this just code that searches for and finds text? Which you can replace if you want, or do something else with it?

# Python

**Style**

* Surround top-level function and class definitions with two blank lines.
* Method definitions inside a class are surrounded by a single blank line.

**Numpy** is a wrapped library that connects to ForTrans and a number of other C libraries to make numerical calculations in python code RUN faster (matrices, etc.)

**Pandas** (“kinda replaces excel”) allows you to analyze and manipulate data. E.g. Table of chemical ID (column), names (column), formula (column), export ait (column), etc. A Huge matrix of data, and what this allows me to do is graph, sort, call specific mathematical functions, etc.

**SciPy** gives you statistics, regression, optimization, etc.

**Iterables**

**-Sequenced**

*Strings* are made up of *characters*. These are like lists for only characters.

*Ranges* are made up of *index\_values*. These are like lists for only numbers.

*[Lists]* are made up of *elements*. These iterables are the most useful.

*(Tuples)* are made up of *variables*. These are like lists that you cannot change.

**-Not Sequenced**

*{Dictionaries}* are made up of *items*, which consist of *keys* that are assigned to a corresponding *value*. Dictionaries are iterables but they are also ao mapping type. They take one type of data and connect it (“map it”) to another. This is also called a “hash map” or a “hash table.”

**-Subscript and Slicing**

*Subscripting* is when you print only 1 item from an iterable

*Slicing* is when you print a series of items from an iterable

**Loops**

While-loops use a truth expression as the constraint

For-loops usually use a range expression as the constraint

**If, elif, and else**

These are not loops, but they are filtering code that determines which direction the code will go. They are only run once, or not at all (because they were skipped over).

**Functions**

You can make up your own functions, meaning, you can make your own structure of procedures to follow when you call this made-up function. That way it simplifies your code when you have to keep calling this function.

* **Parameters** - the general way the function is defined in terms of what it will accept as input ( e.g. def factorial(number) )
* **Argument** - the actual, specific input to a function when you call it ( e.g. factorial(4) )

**Create a Matrix with List Comprehension**

[List Comprehension](https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions)

**Objects and Classes**

Instantiation - calling an object into existence by a function

Shallow copy vs Deep copy. One is a new reference to the same object and the other creates a different object.

**Questions**

~~Logical Expressions~~

1. ~~What’s the difference between~~

~~1 and 2~~

~~And~~

~~bool(1 and 2)~~

If, Elif, and else

1. ~~Error = false in the if, elif, else section. Is error a special variable? What exactly does that do? Does python recognize error as something special to do if this variable is acted upon or something? What purpose does this serve at the beginning of the example?~~ **~~Was it to set up lines 41-44 to manipulate the boolean logic?~~**
2. ~~If elif else: is the error the fact that the situation where the revenue equals 1000 then the message will still say congrats you’re “above” the break even point? But that wouldn’t cause an error...is it dividing an int by a float? Or does python know to convert that into a float already (that’s what I think I heard before)?~~
3. ~~The print on line 53, how come that print happens on both the if and the else statements and not just the else statement because it looks like it’s nested in the else. Is it because it’s not indented?~~

~~User\_input.py file~~

1. ~~Why does it spit out “Violas rule!” every time?~~

File reading and writing

1. ~~What does exit = “” mean?~~
2. ~~Did you override the data variable in line 29?~~
3. ~~Do you always have to store a new file as a variable in the code if you want to manipulate it? Do you always store it as a variable even if you don’t want to manipulate it? Line 19~~
4. ~~Line 16, do you always end a row of data with \n?~~
5. ~~Lines 14-16, why couldn’t you use variableA + “,”, variableB + “,”, variableC + “\n” on one line of code?~~
6. ~~Line 20, if you manually feed it something to write, does that first manual feed always get put in as headers? How come it doesn’t get put in as the data? Or is it just the first row, and the computer doesn’t really interpret it as a header vs data but rather, row 1 vs row 2?~~

Iterables

1. ~~Why does print(“Last from range:”, example\_range[-1]) spit out “4” when the range was 0 to 5? Is it because the 5 is exclusive in the range?~~
   1. ~~And then why are the first two “(0, 2)”? That seems like 0, 1, and 2 to me. Unless the 2 is exclusive in this notation as well.~~

Functions

1. ~~Why do you have to have that running=True variable at line 37?~~
2. ~~Why does the main() function keep running even when the input is already given? Why doesn’t the code just close out? I guess this is related to my previous question.~~
3. ~~For the main() function, is there a way to say “for “make” in input(prompt)” instead of having to set the prompt’s input equal to another variable and seeing if that variable matches your input? In other words, is there a way to bypass the need for a variable and just say, “if the input of the prompt was “make” do this”?~~
4. ~~I’m struggling to understand how recursions can be useful outside of factorials. Maybe an example would help. It kind of seems like a more complicated way to do a for-loop.~~

More Functions

1. I’m trying to understand the use of parameters for functions. The parameters for a function seem like they can be written inside the function, like for the quadratic example, instead of using a, b, and c as parameters, we could have no parameters and ask for input from the user to give us an a, b, and c, and assign those inputs as variables which we would use to find the solutions. That would be valid too right? So what’s the benefit of using parameters over simply assigning those variables inside the function?
2. In lines 43-45, when you put an if statement, you don’t always need to put the else?
3. Could we go over more recursion examples? I could not come up with my own example without writing something without exceeding the max recursion cycles.

Try Except

1. Is running=True a variable?
2. I couldn’t get my raise to work.

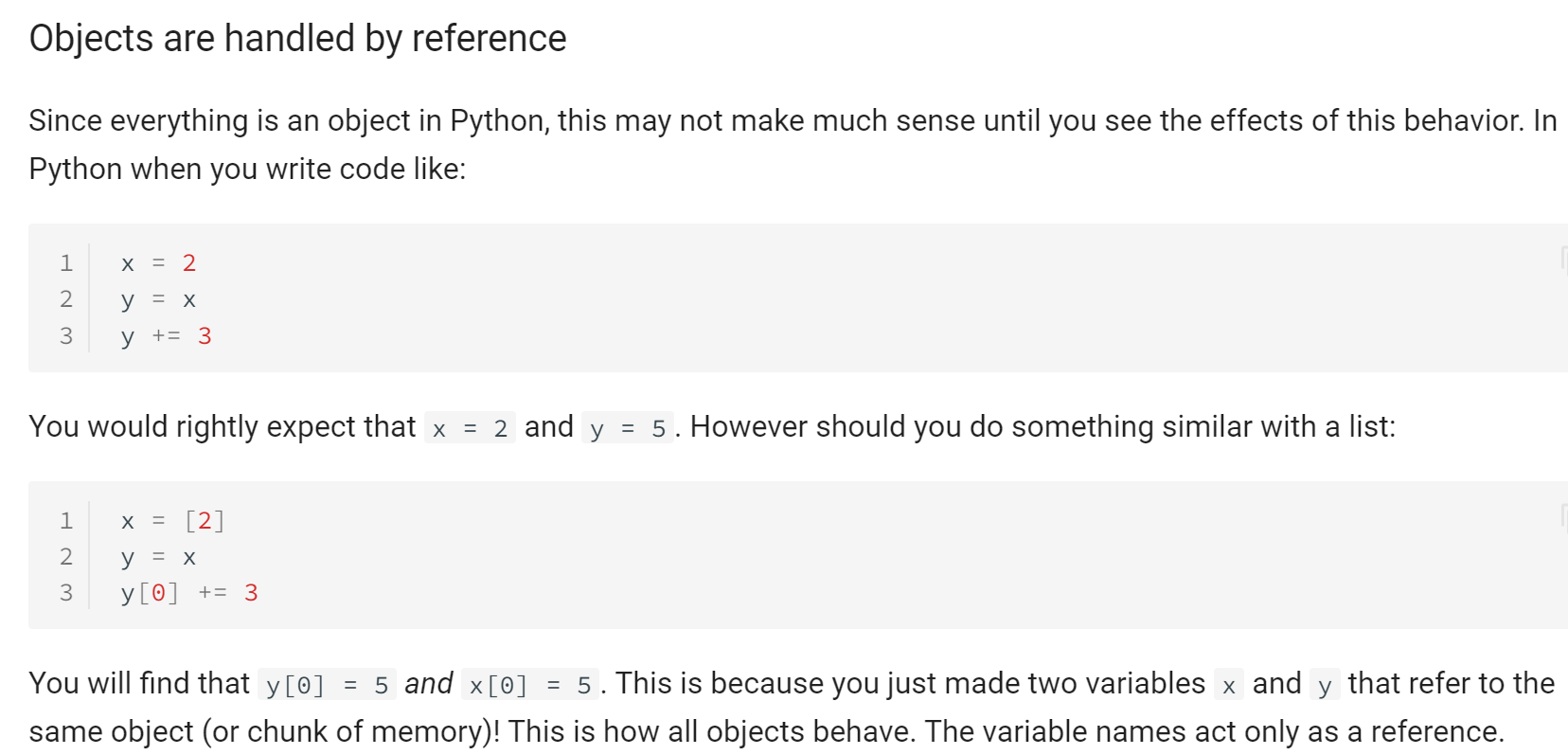
Imports

1. Matrix question (LOVE THIS PROBLEM, cuz it’s relevant to what I want to get into). What exactly is happening? The randomizer is putting a random element into each row, and then each row is being iterated on for size “size”? Which ones are the rows and which ones are the columns? Can we go over this process one element at a time in the whole matrix?

Feedback

1. Make vocab words explicit. Have its own box. Bolded, with a formal definition that we can refer back to when we forget what that word means. I like that you use the formal language of the python code instead of dumbing it down.
2. Scaffold the “Hone Your Skills” with example problems, provide some example code to read for chapter 24, or provide some riddles to solve when defining functions. You do this sometimes, like for the List Comprehension example, and that was gold. I loved that I had a specific resource to refer to, and a specific challenge to do according to the skill level I’m at, because I tried looking for some challenges or problems online but sometimes the answers implied that I knew some other function or method or thing that I didn’t actually know yet.

Objects and Classes

1.  Why is the second example “how all objects behave” but that’s not how the first example behaved?
2. Can you give some examples of why you might want to use classes? I can’t conceive of one, so I don’t think I understand its utility, other than vaguely understanding that it groups a bunch of characteristics for something you want to keep track of, or store.
   1. Do classes always have a function like def \_\_init\_\_(self, name, color, favorite\_food): ?
3. Can we go over the Cat() example and go over what is a method (the functions right?), a member (the species variable right?), and what are the attributes (anything inside the function that belongs to it? Is it also the string “felis cactus” inside the species variable, since the string belongs to the variable? And the functions are, in turn, attributes to the class?
4. Could you explain this part to me? self is just a variable that could be named anything but is used by convention in Python. It is a reference to the **instance** of the object that is referring to the attribute.
5. So what exactly is object-oriented programming? As opposed to what?
6. So I looked up operator overloading, and I found this page on Real Python <https://realpython.com/operator-function-overloading/> and I saw how you can specify how operators behave with your particular class, like adding to a shopping cart, however, I didn’t understand how some of the other examples could be really useful. Could we go over a couple of their examples? Particularly the len() and the str() ones?