Data Wrangling in Python

Chapter 2: Python Basics

1. Data Types?

- Data type is a feature that is associated with a piece of data that conveys how that piece can be used.
- There are different data types and those are strings, integer, floats and other non-whole number types.

a. Strings

- String is basically a piece of text and it is usuallu denoted using quotes. They can contain numbers, letters and symbols.
- Anything between those quotes regardless of logic is called as string. It can be done using either single or double quotes.

In [1]:
```
'cat'
```

Out[1]: 'cat'

In [2]:
```
type('cat')
```

Out[2]: str

In [3]:
```
"dog"
```

Out[3]: 'dog'

In [4]:
```
type("dog")
```

Out[4]: str

- We can see that by putting 'cat' and "dog" between type() we can get to know that the interpreter recognizes it as a string.

In [5]:
```
'4'
```

Out[5]: '4'

In [6]:
```
type('4')
```

Out[6]:
```
str
```

- Even numbers stored between quotes are considered as strings.

b. Integers and Floats

- Integers range from -infinity to +infinity. If we enter them the interpreter will return them to us.

In [7]:
```
10
```

Out[7]:
```
10
```

In [8]:
```
type(10)
```

Out[8]:
```
int
```

- Let's check if number stored in quotes is equal to the number not stored in quotes.

In [9]:
```
5 == '5'
```

Out[9]:
```
False
```

- The interpreter claims it as false through boolean operation where the possible outcomes can be either true or false. We found out using '=='.

- We might wonder why we should have a need to store number as strings, some special situations might occur, let's say you try to store Boston's zip code as an integer so as the number starts with zero the compiler will throw an error.

In [10]:
```
02018
```

```
  File "C:\Users\MY PC\AppData\Local\Temp\ipykernel_16380\2700973128.py",
line 1
    02018
      ^
SyntaxError: leading zeros in decimal integer literals are not permitted;
use an 0o prefix for octal integers
```

In [11]:
```
'02018'
```

Out[11]: '02018'

c. Floats, Decimals and non-whole number types

- When a non–whole number is used in Python, Python defaults to turning the value into a float. A float uses the built-in floating-point data type for your Python version. This means Python stores an approximation of the numeric value—an approximation that reflects only a certain level of precision.

In [12]:
```
type(2)
```

Out[12]: int

In [13]:
```
type(2.0)
```

Out[13]: float

- Python can be used to perform calculations, float calculations >>> integer calculations in terms of processing time.

In [14]:
```
2/3
```

Out[14]: 0.6666666666666666

In [15]:
```
2.0/3
```

Out[15]: 0.6666666666666666

In [16]:
```
0.3
```

Out[16]: 0.3

In [17]:
```
0.1+0.3
```

Out[17]: 0.4

- If we need accurate values it's better we use decimal library to get accurate results.
- A module or library is a huge set of code that is used to import for your use, this decimal will get you accurate results like how math was taught.

In [18]:
```
from decimal import getcontext, Decimal
getcontext().prec = 1
Decimal(0.1)+Decimal(0.2)
```

Out[18]:
```
Decimal('0.3')
```

In [19]:
```
getcontext().prec = 4
Decimal(0.1)+Decimal(0.3)
```

Out[19]:
```
Decimal('0.4000')
```

- As the number increases in getcontext().prec the number of decimals increase.
- There are plenty of libraries for numbers in python based on the requirement and those are decimal, math, numpy, sympy and mpmath.

2. Data Containers

- Data Containers have the ability to store multiple data points and these containers are data types as well, Python has few common containers and those are variables, lists and dictionaries.

a. Variables

- Variables give us way to store values, such as strings, numbers or other data containers.
- They should convey what is stored in them. use lowercase with words connected with underscores to describe what is contained.

In [20]:
```
filename = 'file.txt'
```

- As the interpreter didn't return anything, it is an indication that filename variavle contains file.txt. We can get to know by just typing the variable name and running it.

In [21]:
```
filename
```

Out[21]:
```
'file.txt'
```

Object-Oriented Programming

- Python is an object-oriented programming language. The “object” in OOP can be any of the data types we learned about in this chapter such as strings, variables, numbers or floats.

In [22]:
```
who = "my name is Abhijith Kasula"
```

In [23]:
```
who
```

Out[23]:
```
'my name is Abhijith Kasula'
```

- Variables have the ability to store long strings, if grammatically if there is a requirement to use single quote then we should use double quote to store the string in the variable.

b. Lists

- A list is a group of values that have some relationship in common.
- you can create a list of items by placing them within square brackets([]) and separating them with commas.

In [24]:
```
['milk','apples','eggs']
```

Out[24]:
```
['milk', 'apples', 'eggs']
```

In [25]:
```
list = ['milk','apples','eggs']
```

In [26]:
```
list
```

Out[26]:
```
['milk', 'apples', 'eggs']
```

- Not just strings, different data types or combination of datatypes can be stored in lists.

In [27]:
```
[0,1.7,-4,3.99]
```

Out[27]:
```
[0, 1.7, -4, 3.99]
```

- Lists can also store variables, let's say we have variables representing counts of animals we are tracking in an animal shelter.

In [28]:
```
cats = 3
dogs = 7
cows = 5
```

In [29]:
```
farm = [cats,dogs,cows]
```

In [30]:
```
farm
```

Out[30]:
```
[3, 7, 5]
```

- In the above case we used variables as the type of animals and in those variables we stored number of animals of that kind and assigned it to a list.
- We can also create a list of lists.

c. Dictionaries

In [ ]: