Python Mind Map notes

Mind Map link: <https://orgpad.info/s/8idPfY_Nn8x>

Basics

* **Integrated Development Environments (IDEs) for Python**
  + What is an IDE?
    - An integrated development environment (IDE) is a program dedicated to software development. As the name implies, IDEs integrate several tools specifically designed for software development. These usually include:
      * An editor designed to handle code (with, for example, syntax highlighting and auto-completion)
      * Build, execution, and debugging tools
    - Most common IDEs for python
      * Visual Studio Code
      * PyCharm
      * Spyder
      * Jupyter/JupyterLab
* **Working Directory**
  + The Working directory is the default file path in your computer (or on the cloud) that Python reads or saves files into. For example, ‘C:\thuvu\Desktop\Data Science\ ‘
  + Import the operating system package with ‘import os’
  + Print the current working directory ‘print os.getcwd()’
  + Change working directory new path ‘os.chdir(new\_path)
* **Data Types**
  + In programming, data type is an important concept. Messing with them can cause unwanted errors and break your programs.
  + Different types can do different things.
  + This section explores different built-in data types in Python
    - How types work in Python:
      * Since Python is an interpreted language, the data type is automatically set when you assign a value to a variable
      * X = 1, x is a type int
      * Y = 2.8 y is a type float
      * Z = “hello” z is a type string
      * However, if you want to specify the data type of a variable, this can be done with casting
        + X = str(3) x with be a string
        + Y = int(3) y with be an int
        + Z = float(3) z with be 3.0
      * Check data type of a variable:
        + To verify the type of any object in Python, use the type() function print(type(x))
      * Type Conversion: we can change a variable’s type after it has been set – str(), int(), float(), bool()
    - Numeric type
      * Integer – Number like 1, 2, 3
      * Float – 5.5, 3.12233
      * Complex: represents a number with both a real part and an imaginary part - x = 3 + 4j -> you can use the complex function x = complex(3, 4) which will be the same as 3 + 4j
    - Text Type
      * Represents the data that have text value
      * A string is a collection of one or more characters put in a single quote, double-quote, or triple quote
        + String variables can be declared either by using single or double quotes: hello\_msg = “hello” or ‘hello’
    - Sequence types
      * This is the ordered collection of similar or different Python data types. Sequences allow storing of multiple values in an organized and efficient fashion
        + *List*

A list is an ordered, changeable sequence of elements

It allows duplicate values

It can hold integers, characters, floats, strings, and even objects

You can mix different types in one list, but there is actually no good reason to do so

When to use: When you want to store multiple items in a single variable

Create a list: lists are wrapped around by square brackets x = [1, 2, 3] or y = [‘apple’, ‘banana’]

List operations: <https://docs.python.org/3/tutorial/datastructures.html>

* + - * + *Tuple*

Ordered collection of items. Tuples can contain elements of different types, including numbers, strings, and other collections like lists or even other tuples. Because tuples are immutable, once they are created, their elements cannot be changed, added, or removed.

my\_tuple = (1, “Hello”, 3.14)

Operations: <https://www.geeksforgeeks.org/python-tuples/>

* + - * + *Range*

Range is a built-in function that generates a sequence of numbers. It is commonly used in ‘for’ loops to iterate over a sequence of numbers. The ‘range’ function is particularly usedful when you need to perform a fixed number of iterations or generate a sequence of number for other purposes

start is the number to start at

stop is the number to end at

step is the number to increment by

range(stop)

range(start, stop)

range(start, stop, step)

* + - * Set Types
      * A set is an unordered collection of unique elements. Sets are useful for storing items where duplicates are not allowed, and they provide efficient operations for membership testing, adding, and removing elements.
        + *Set*

A set is an unordered collection with no duplicate emlements

When to use: When you want to create a unique list of elements, and you may want to use mathematical operations on it like union, intersection, difference, and symmetric difference.

Basket = {‘apple’, ‘orange’, ‘apple’, ‘pear’, ‘orange’, ‘banana’} and when you print it print(basket) -> (‘orange’, ‘banana’, ‘pear’, ‘apple’)

Set operations in the link below

*<https://realpython.com/python-sets/>*

* + - * + *Frozen set*

A frozenset is an immutable version of a set. While a set can be modified (elements can be added or removed), a frozenset is static and its contents cannot be changed after it is created. This immutability makes frozenset hashable, meaning it can be used as a key in a dictionary or stored in other sets.

my\_frozenset = frozenset([1, 2, 3, 4])

* + - * Boolean Type
        + Booleans represent one of two values: True or False
      * Binary Types
        + Are used to handle binary data. They are particularly useful when dealing with file I/O operations, network communication, and data serialization where data is not in the form of text string but rather in bytes or byte sequences. Python provide several built-in types to work with binary data: ‘bytes’, ‘bytearray’, and ‘memoryview’.

*Bytes*

*Once created, the contents of a ‘bytes’ object cannot be changed so* ***immutable***

Usage: Suitable for representing binary data that should not be modified

Creation: Using the ‘b’ prefix before a string literal or by calling the ‘bytes()’ constructor

*Bytearray*

The contents of a ‘bytearray’ can be modified which is **Mutable**

**Usage:** Suitable for representing binary data that needs to be changed

**Creation:** using the ‘bytearray()’ constructor.

*Memoryview*

Usage: Provides a view of another binary data object without copying it, allowing for efficient manipulation of a large data sets

Creation: Using the ‘memoryview()’ constructor

* + - * None Type
        + Is a special constant representing the absence of a value or a null value. It is an object of its own datatype – the ‘NoneType’. ‘None’ is often used to signify ‘nothing’ or ‘no value’ in a function or a variable, and it is commonly used as a default value for option function arguments.
        + Characterisics:

There is only one instance of ‘None’, and it is a singleton. This means that ‘None’ is always the same object in memory.

Falsiness: ‘Non’ is considered ‘False’ in a Boolean context.

Type: ‘Non’ is of the ‘NoneType’

X = None

* **Variables**
  + Variables are containers for storing data values.
  + A variable is created the moment you first assign a value to it, for example:

**x = 5**

**y = “John”**

* **Functions**
  + Functions take in inputs, use it to run a set of code and returns an output
  + Inputs -> Code -> Output
  + Basic Functions
    - print()
    - type()
    - input()
  + Create a Python Function
    - This Function will return the value of 5 \* x

**def my\_function(x):**

**return 5 \* x**

* + - This Function will not return anything

Def my\_function():

Print(“Hello from a function”)

* **Operators**
  + Arthmetic
  + Assignment
  + Comparison
  + Logical
  + Identity
  + Membership
  + Bitwise
* **Python Statements**
  + Expression Statements
  + Conditional Statements
  + Loop Statements
    - Loop Control Statements