Python Course Summarization Part 2

Made by: Rawan Hatem

Collections

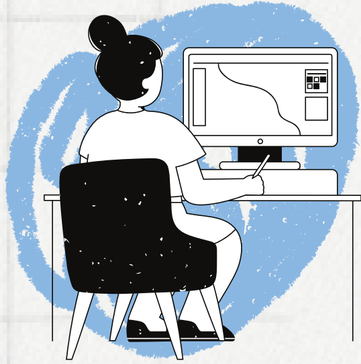
Lists

- The four general-purpose collections are lists, sets, tuples, and dictionaries.
- A collection is a single variable used to store multiple values.
- Lists are ordered and changeable; duplicates are allowed.
- To create a list, surround values with square brackets.
- List elements are accessed using the index operator (e.g., fruits[0]).
- You can iterate over a list using a for loop.
- List methods include append, clear, copy, count, extend, index, insert, pop, remove, reverse, and sort.

Collections
Sets & Tuples

- Sets are unordered and immutable; duplicates are not allowed.
- To create a set, surround values with curly braces.
- Set methods include add, clear, copy, difference, difference_update, intersection, intersection_update, isdisjoint, issubset, issuperset, symmetric_difference, symmetric_difference_update, union, and update.
- Tuples are ordered and unchangeable; duplicates are allowed.
- Tuples are faster than lists.
- To create a tuple, surround values with parentheses.
- Tuple methods include count and index.

Dictionary



- A dictionary is one of the four basic collection types in Python, consisting of key-value pairs.
- Key-value pairs are **ordered** and **changeable**, with **no duplicates** allowed.
- To add a value to a key, type the key, followed by a colon, and then the value. Separate each key-value pair with a comma, like this **{key : value}**
- To **see all the attributes and methods of a dictionary** you can use **“dir”** function pass in dictionary name
- Retrieving the values of a dictionary using the 'get' method and the name of the dictionary
- If the **key** is **not found** in the dictionary, the **'get'** method will **return 'None'**.
- To update an existing key-value pair or insert a new one, use the 'update' method and pass a set of key-value pairs within curly braces.

Dictionary. Cont

- To **remove a key-value** pair, use the '**pop**' method and **pass** the **key** as an argument.
- To **remove the latest key-value** pair inserted you can use **“popitem**”.
- The '**clear**' method **empties the entire dictionary.**
- To **get all** the **keys** within the dictionary, use the '**keys**' method.
  - keys is an object which resembles a list
  - if you need to iterate over all the keys you use for loop to iterate over every key that is returned from the keys method of your dictionary
- To **get all** the **values** within the dictionary, use the '**values**' method.
  - Values is an object which resembles a list
  - If you need to iterate over all the Values you use for loop to iterate over every value that is returned from the values method of your dictionary
- To **get both keys and values** within the dictionary, use the '**items**' method.
  - it returns a dictionary object which resembles a 2D list of Tuples ,this be useful to use a for loop to iterated over every key value pair
  - for key, value in dictionary_name.items(): print(f”{key}: {value}”)

Generate Random Numbers

- Useful methods in the random module include randint, random, and choice.
  - ‘randint(a, b)’ -->> generates a random integer between whole numbers a and b.
  - ‘random()’ -->> generates a random float between 0 and 1.
  - ‘choice(seq)’ -->> picks a random element from a sequence like a list or tuple.
- The shuffle(x) function randomly shuffles the elements in a list x.
- To create a number guessing game in Python:
  - Set variables for low, high, guesses, and a random number between low and high.
  - Use a while loop to let the user guess until they get it right.
  - Give hints to the user if their guess is too high or too low.
  - Count and display the number of guesses it took the user to guess correctly.

Functions

- A **function is**: a block of reusable code that can be invoked to perform a specific task.
- Functions **used to:**
  - **avoid repeating code**
  - **simplify loops.**
- **Define a function :**
  - **def** keyword followed by a **unique function name, parentheses, and a colon.**
- The code that belongs to the function should be indented underneath the function definition.
- To invoke a function, type its name followed by a set of parentheses.
- Functions can accept data directly using arguments, which are placed within the parentheses when invoking the function.
- Arguments are sent to a function's parameters, which act as temporary variables within the function.
- The order of arguments and parameters matters when invoking a function.
- Functions can return a result back to the caller using the return statement.
- The return statement ends a function and sends a result back to the place where the function was invoked.
- The returned value can be assigned to a variable for further use.
- Functions can be used to create complex behaviors by breaking them down into smaller, reusable pieces of code.

Default Arguments

- Default arguments are default values for certain parameters in a function; the default value is used when that argument is omitted when invoking the function.
- Default arguments can reduce the number of arguments that need to be passed into a function, especially if the arguments are consistent most of the time.
- Non-default arguments should always follow default arguments, so the start and end parameters are reversed in the function definition.
- Default arguments make functions more flexible and can reduce the number of arguments users need to pass in, particularly if those arguments are consistent most of the time.

Keyword Arguments

- Keyword arguments are arguments preceded by an identifier, which can make code more readable and allow for flexibility in the order of arguments.
- There are four basic styles of arguments: positional, default, arbitrary, and keyword.
- Non-default arguments should always follow default arguments, so the start and end parameters are reversed in the function definition.
- Default arguments make functions more flexible and can reduce the number of arguments users need to pass in, particularly if those arguments are consistent most of the time.
- Keyword arguments in Python are just arguments preceded by an identifier that matches the name of a function's parameters.

Module

- A module is a Python file containing code that you want to include in your program.
- You can include a module using the "import" keyword.
- There are built-in modules available in the standard Python library, and you can also create your own.
- To see a list of all modules available in the standard library, use the "help" function and pass in the word "modules".
- Some examples of built-in modules are math, string, time, and pickle.
- To see the variables and functions available in a module, use the "help" function with the module name, such as "help(math)".
- To access variables and functions within a module, use the format "module_name.variable_or_function".
- You can give a module a nickname or alias using the "import module_name as alias" format.
- Another way to import is by using "from module_name import variable_or_function", but this can cause name conflicts.
- To create a module, create a new Python file within your project folder and declare your desired variables and functions.
- To import a module you created, use the "import module_name" format.
- When using variables or functions from a module, it's best practice to prefix the name with the module name to avoid confusion and potential name conflicts.

Scope Resolution

- Variable scope refers to where a variable is visible and accessible.
- Variables declared within a function have a local scope.
- Different functions can't see inside of each other, but they can see inside of their own function.
- Scope resolution follows the LEGB rule: Local, Enclosed, Global, Built-in.
- When using a variable, Python will first look for a local instance of that variable. If not found, it will move to the enclosed scope, then the global scope, and finally the built-in scope.
- Functions can't see inside of other functions, so arguments are passed to make functions aware of them.
- Variables can have different versions or instances within different scopes.
- Global variables are defined outside of any functions and can be accessed from any part of the code.
- Built-in variables come from Python's built-in scope, such as the mathematical constant 'e'.
- Variables can share the same name, as long as they are in different scopes.
- If there is no local or enclosed version of a variable, and a global version exists, Python will use the global version following the LEGB rule.

Exception Handling

- Exception handling in Python is about managing events that interrupt the normal flow of a program during execution.
- An exception is detected during execution and causes an interruption in the program's flow. For example, dividing a number by zero will cause an exception.
- A very basic form of exception handling is to surround any code that is considered dangerous with a try block. This code might cause an exception, and if it does, it can be caught and handled.
- To catch and handle exceptions, an except block is added after the try block. This block prevents the program from being interrupted and allows for something else to be done.
- It is not considered good practice to have a single except block that handles all exceptions. Instead, it is better to first handle specific exceptions when they occur.
- Additional except blocks can be added to handle specific exceptions, such as a ZeroDivisionError or a ValueError.
- It is optional to display the exception that occurs, but it can be helpful in understanding what went wrong.
- An else statement can be added to the end of the accept blocks to specify what should be done if no exceptions occur.
- A finally clause can be added to specify what should be done whether or not an exception is encountered. This is useful for closing files that have been opened.
- The text covers exceptions in Python, how they work, and how they can be handled using try and except blocks.

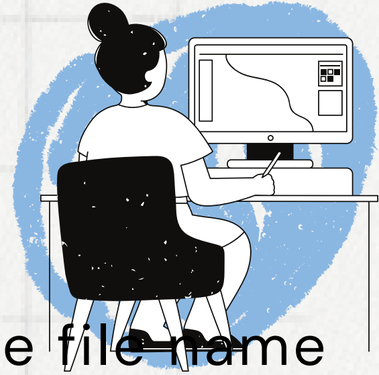
File Detection

- The os module is recommended for this task.
- To check if a file exists, use "os.path.exists()" and pass the file path as an argument.
- If the file exists, it returns True; otherwise, it returns False.
- To check if a file is located at a specific path, use "os.path.isfile()" and pass the path as an argument.
- If the location is a file, it returns True; otherwise, it returns False.
- To check if a location is a directory, use "os.path.isdir()" and pass the path as an argument.
- If the location is a directory, it returns True; otherwise, it returns False.
- The video is the beginning of a mini-series about doing stuff with files using Python.
- The code for the file detection demonstrated in the video will be provided in the comments section.

Read a File

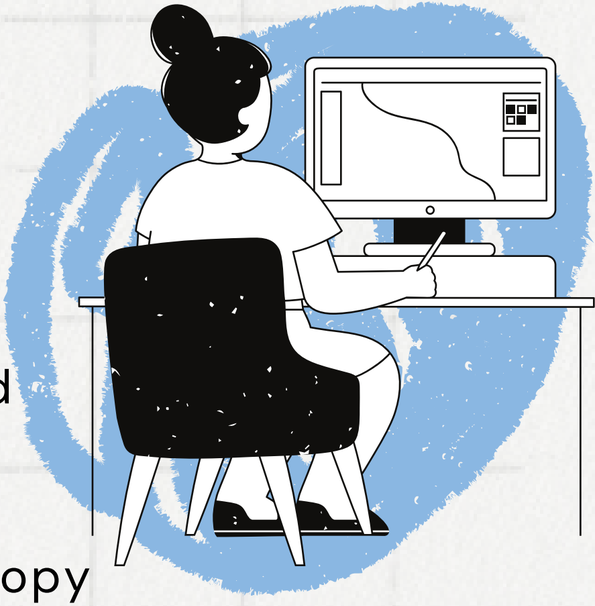
- how to read a file in Python.
  - creating a text file "text_file.txt"
  - To read the contents of the file, use the open() function, followed by the name of the file. -> with open (text_file.txt ,'r') as file
  - The print(file.read()) function is then used to print the contents of the file to the console.
  - The with open() function is used to automatically close the file after it has been opened.
  - A try and except block can be used to catch and handle exceptions, such as a "file not found" error.
  - If an exception occurs, the program can print a message indicating that the file was not found.

write File



- To write to a file, you need to use the open() function and specify the file name or path as the first argument.-> with open (text_file.txt ,'w') as file
- The second argument in the open() function is the mode, which is set to 'w' for writing.
- A new line character can be added to the text using the newline character.
- After running the program, a new text file should appear in the project folder.
- If the text in the file is changed, it will overwrite the current file.
- To append text to an existing file, you can change the mode to 'a'.

Copy File



- The recommended module for this task is shutil.
- There are three functions in shutil for copying files: copyfile(), copy(), and copymode().
- copyfile() copies the contents of a file.
- copy() copies the contents of a file and the permissions mode, and can copy to a directory.
- copymode() copies the permissions mode only.
- copyfile() is the function that will be used in this tutorial.
- To copy a file using copyfile(), use shutil.copyfile(source, destination).
- The source is the file to be copied, and the destination is the new name and location of the copied file.
- If the destination is not in the same directory as the source, provide the full file path for the destination.
- The arguments for copy() and copymode() are the same as copyfile().
- The difference between copy() and copymode() is that copy() copies the file contents and permissions, while copymode() only copies the permissions.

Move File

- Create two variables: "source" for the location of the source file and "destination" for the new location.
- The source file can be specified by just the file name if it's in the same directory, but a full path may be needed if it's in a different location.
- Similarly, the destination can be specified by just the file name if it's a known location, such as the desktop.
- It's recommended to use a try/except block to handle any exceptions, specifically file not found errors.
- Before moving the file, check if there's already a file in the destination using os.path.exists().
- If there is, you can choose to let the user know or override the file. To override, use os.replace(source, destination).
- The same method can be used to move directories by specifying the source and destination directories.

Delete File

- The recommended module for this task is "os".
- To delete a file, use " os.remove(file_path) ".
- It is possible to delete a file using a variable for the file path, like "os.remove(path)".
- If a non existent file is attempted to be deleted, a "FileNotFoundError" exception is thrown.
- To handle this exception, use a "try/except" block and print "File not found".
- The "os" module does not remove empty folders; to delete an empty folder, use "os.rmdir(folder_path)".
- If there's a permission error when deleting a folder, it can be handled with an "except" block and print "Permission denied".
- To delete a folder that contains files, import "shutil" and use "shutil.rmtree(path)".
- Be cautious when using "shutil.rmtree" as it deletes the directory and all contained files.