INTERNATIONALE
HOCHSCHULE

* Assignment Title *

<u>Exploring Python’s Standard Library</u>

Table of Contents

1. Introduction

The collection of programs and conventions that might be used with Python without the need for extra programming languages is referred to to as the Python Standard Library. It has various features that make it simple to use and capable of handling a wide range of jobs. This model addresses a wide range of topics, including as networking, data processing, file and directory access, and more.

Python comes with lots of pre-written programs whenever you install so that developers do not have to learn everything from zero when developing products using this sophisticated system. Applying the requirements of scalability and reusability, the standard library provides an easily understood, dependable product which is an important part of the Python ecosystem.

- ## Importance in Python Development

For many reasons, the Python Standard Library is important for the development of Programming:

Performance and productivity :-

The Standard Library allows developers prevent implementing distinct functions from begin by providing ready-to-use and functioning standards. This not just saves time but also increases the application's reliability, reduces error times, and makes use of well-tested code.

Regularity :-

The library standard encourages regularity in one's work. The usage of common procedures and approaches by developers across projects and companies facilitates the management of the problem-solving process.

All-inclusive support :-

Many functions are supported by the function libraries. It includes modules for data interaction, interacting, dealing with directories and lists, storing data etc. With the support of this comprehensive the solution, developers get the tools they want for a range of applications.

Integration with Python :-

The model of the library's structure follows closely to the syntax and language regulations of Python, providing a solid foundation and hands-on experience. Performance improves overall and better usage of the library's features is made available by this integration.

Language Integrated :-

The modules in the Standard Library are often closely integrated with Python itself, e.g., through syntax (what gets done when you type expressions). The integration of these library features is seamless so it makes the overall programming experience a lot smoother and more natural.

Education & Change :-

The Standard Library is a great place for new Python developers to learn from. An examination of its modules and functions can teach us lessons in best-practice, what the language is capable of with reference to some advanced features thereof; as well how perform common programming pattern actions.

<u>2.System and Operating System Interfaces</u>

Notice that standard Python ships with some modules for interacting with the underlying operating system. Developer write some tasks on the system-level, such as working with files and directories, interacting environment variables, managing processes. The ‘sys’, ‘os’ and ‘shutil’ are main modules of these types. Now, let us move on by learning with the details of each module.

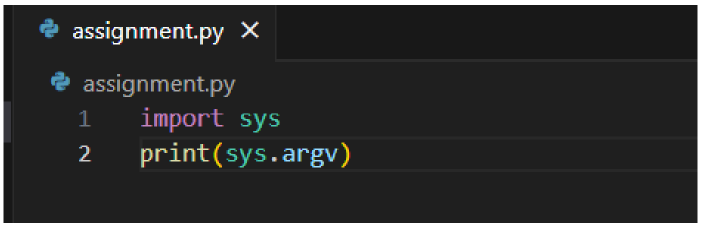
<u>2.1. ’sys’ Module :-</u>

<u>Key Functions and Attributes :-</u>

1. sys.argv :-

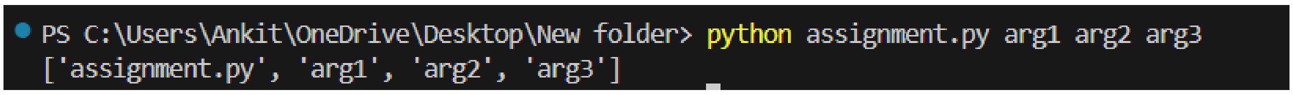
Description : List of arguments passed to the script. The first element is the case name.

Example



```
assignment.py
import sys
print(sys.argv)
```

Run & Output

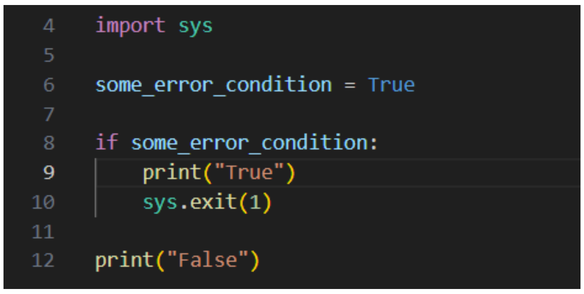


```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python assignment.py arg1 arg2 arg3
['assignment.py', 'arg1', 'arg2', 'arg3']
```

2. sys.exit([status]) :-

Description : Exit the program and find to give the correct status. A status code of 0 indicates success, any non-zero value indicates failure.

Example



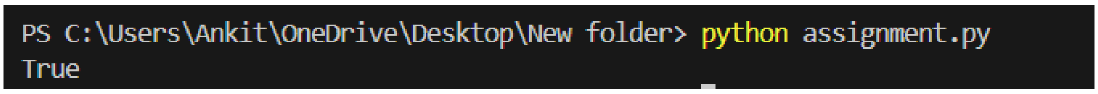
```python
import sys

some_error_condition = True

if some_error_condition:
    print("True")
    sys.exit(1)

print("False")
```

Run & Output



```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python assignment.py
True
```

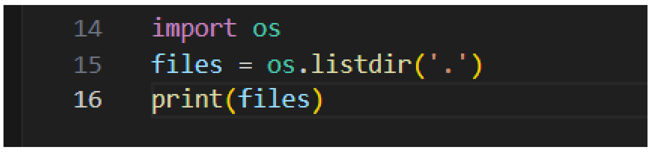
<u>2.2. ‘os’ Module</u> :-

<u>Key Functions and Attributes :-</u>

1. os.listdir(path='.') :-

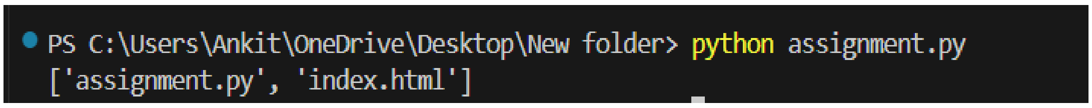
Description : Give all information and names under the path.

Example



```python
import os
files = os.listdir('.')
print(files)
```

Run & Output

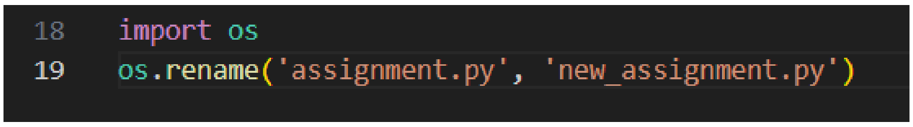


```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python assignment.py
['assignment.py', 'index.html']
```

2. os.rename(source, destination) :-

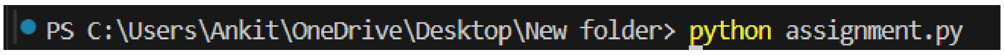
Description : Use to renames a file or directory from source to destination.

Example



```python
import os
os.rename('assignment.py', 'new_assignment.py')
```

Run & Output



```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python assignment.py
```

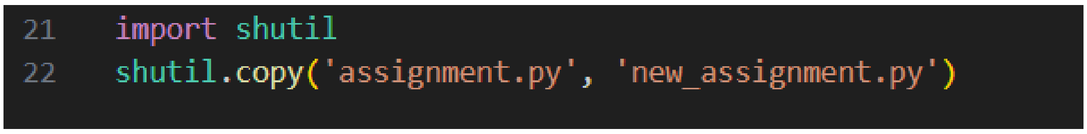
2.3. ‘shutil’ Module :-

Key Functions and Attributes :-

1. shutil.copy(source, destination) :-

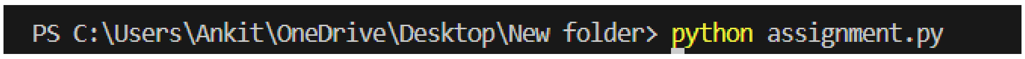
Description : This command used to copy file from source to destination, which may either be a file or a directory.

Example



```python
import shutil
shutil.copy('assignment.py', 'new_assignment.py')
```

Run & Output

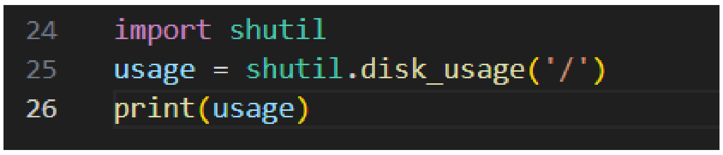


```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python assignment.py
```

2. shutil.disk_usage(path) :-

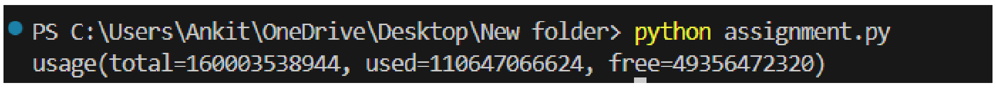
Description : This command used to get the total, used, and free space.

Example



```python
import shutil
usage = shutil.disk_usage('/')
print(usage)
```

Run & Output



```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python assignment.py
usage(total=160003538944, used=110647066624, free=49356472320)
```

3.Data Persistence

Data persistence is being able to save data and keep it through operations. This can be done in Python through various modules available in the standard library. Some techniques we are going to discuss here like pickling object for Serialization and JSON manipulation with python json module.

3.1. The pickle Module :-

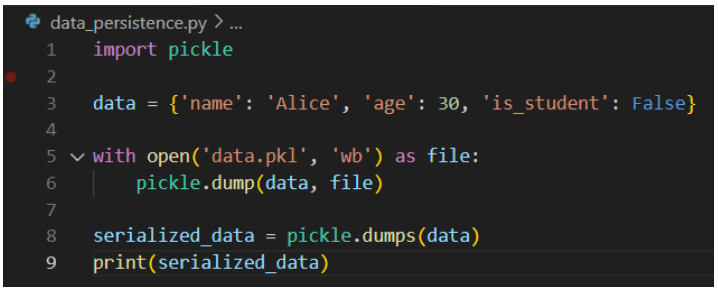
Python objects can be stored and deserialized via pickle. Python objects can be serialized, referred to as pickled, to generate bytes that can be written to a file or exchanged over a wired connection. The serialization of the byte stream to a Python object is known as unpickling.

3.1.1. Basic Operations :-

Pickling (Serialization) :-

Pickle: Python pickle module is used for serializing and de-serializing a python object to byte file. dump function — it writes the serialized object to a file. Alternatively, pickle. dumps() — Serializes the object and return as byte string.

Example


data_persistence.py > ...

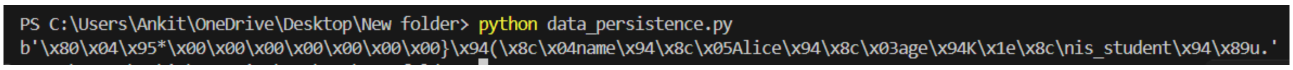
```python
import pickle

data = {'name': 'Alice', 'age': 30, 'is_student': False}

with open('data.pkl', 'wb') as file:
    pickle.dump(data, file)

serialized_data = pickle.dumps(data)
print(serialized_data)
```

Run & Output

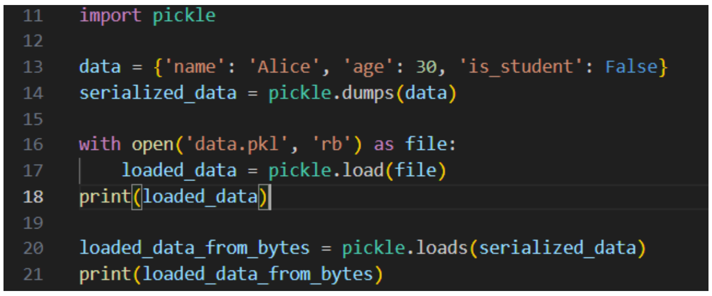


```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python data_persistence.py
b'\x80\x04\x95*\x00\x00\x00\x00\x00\x00\x00}\x94(\x8c\x04name\x94\x8c\x05Alice\x94\x8c\x03age\x94K\x1e\x8c\nis_student\x94\x89u.'
```

Unpickling (Deserialization) :-

To deserialization a Python object, read the processed object from the archive utilizing the pickle.load() function. In contrast, Pickle.load() transforms a byte string back into a Python object.

Example



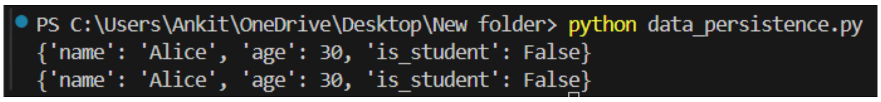
```python
import pickle

data = {'name': 'Alice', 'age': 30, 'is_student': False}
serialized_data = pickle.dumps(data)

with open('data.pkl', 'rb') as file:
    loaded_data = pickle.load(file)
print(loaded_data)

loaded_data_from_bytes = pickle.loads(serialized_data)
print(loaded_data_from_bytes)
```

Run & Output



```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python data_persistence.py
{'name': 'Alice', 'age': 30, 'is_student': False}
{'name': 'Alice', 'age': 30, 'is_student': False}
```

3.1.2. Use Cases and Considerations :-

Use Cases :-

Application State : Using for storing objects such custom classes, functions or python data structures.

Data Transfer : Serialize objects before transmission over a network.

Considerations :-

Security Risks: Take care while pickle or unpickle the data from untrusted sources, it may suffer code execution vulnerabilities.

Compatibility : Pickled object is not in compatible form across different Python versions or implementations.

3.2. The json Module :-

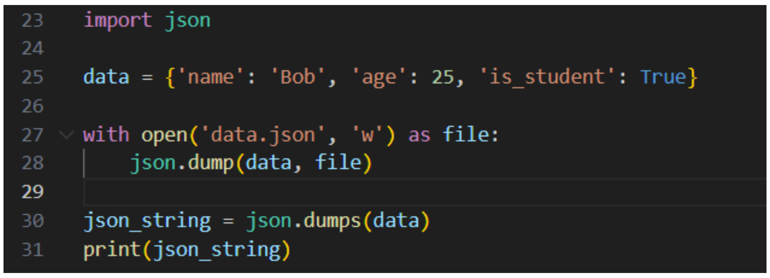
Json Encode and decode JSON data (The python json module this of coarse) evacuates As you know, JSON (JavaScript Object Notation) is a lightweight data interchange format easy for humans to write and read themselves as well computers that there are simple for machines parse and generate. Today JSON is a very popular choice to store and communicate data, especially in web applications.

3.2.1. Basic Operations :-

Encoding (Serialization) :-

Before you convert a Python object to JSON using the json. Writing JSON to a File or json.dumps(). The dumps() to get a JSON formatted string.

Example



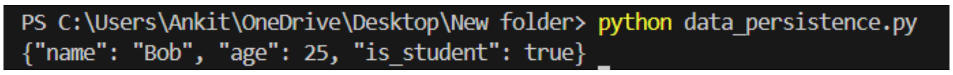
```python
import json

data = {'name': 'Bob', 'age': 25, 'is_student': True}

with open('data.json', 'w') as file:
    json.dump(data, file)

json_string = json.dumps(data)
print(json_string)
```

Run & Output

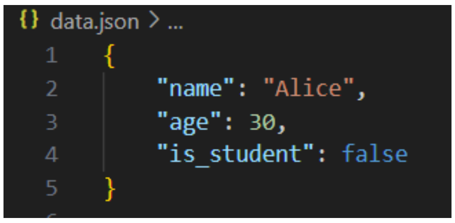


```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python data_persistence.py
{"name": "Bob", "age": 25, "is_student": true}
```

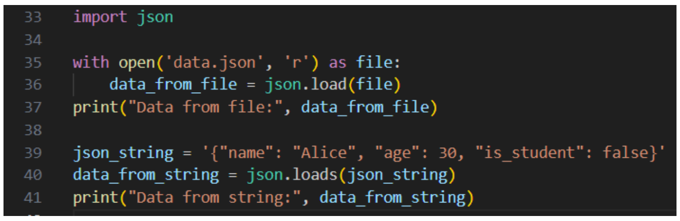
Decoding (Deserialization) :-

The objects in Python can be generated by decoding JSON data utilizing the json.loads() function. A file or JSON is able to read through the use of the load() function. To parse a JSON string, use loads().

Example



```
{} data.json > ...
{
    "name": "Alice",
    "age": 30,
    "is_student": false
}
```

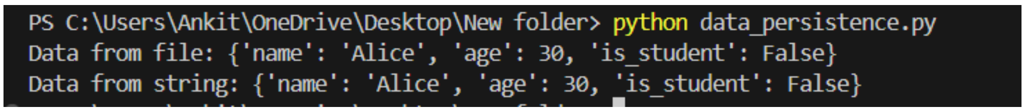


```
import json

with open('data.json', 'r') as file:
    data_from_file = json.load(file)
print("Data from file:", data_from_file)

json_string = '{"name": "Alice", "age": 30, "is_student": false}'
data_from_string = json.loads(json_string)
print("Data from string:", data_from_string)
```

Run & Output



```
PS C:\Users\Ankit\OneDrive\Desktop\New folder> python data_persistence.py
Data from file: {'name': 'Alice', 'age': 30, 'is_student': False}
Data from string: {'name': 'Alice', 'age': 30, 'is_student': False}
```

3.2.2. Use Cases and Considerations :-

Use Cases :-

Data Interchange :- JSON is mostly used to exchange the data between a server and web client

(ex. REST APIs).

Configuration files :- Typically helpful for saving configuration data in human-readable formats.

Data Storage :- suitable for simple data storage cases where readability & portability is important.

Considerations :-

Data Types :- In contrast to Python, JSON only supports a limited amount of data forms. For example, customized Python objects are not supported directly by JSON.

Precision :- Because of its data type limitations, JSON may not be suitable to handle particularly precise numerical values (like floats).

4.Networking and Internet Protocols

Networking and internet protocols are the only way we can connect two systems over a network, or on to the web. Python Standard Library also provides some powerful modules which can be used to perform network communication and protocol handling. This is an overview of the "socket", "http" and "urllib" modules; how they work along with their various typical usage.

4.1. The "socket" Module

4.1.1. Core Concepts :

Sockets :- A socket is an endpoint for sending and receiving data across a network. It forms a communication medium when an IP address is added with port number

Address Families :- The socket module also python-supports a variety of address families. AF_INET (IPv4 addresses) is the most common. AF_INET6: This is used for IPv6 addresses.

Socket Types :- The primary two types of sockets are SOCK_STREAM (for TCP, which is connection based) and SOCK_DGRAM (for UDP).

4.1.2. Considerations :

Error Handling :- The network endpoints can fail due to the failure in a connection, no internet or invalid data. Of course error handling works.

Security :- To ensure data is being transmitted securely to the server, we can use SSL/TLS.

4.2. The “http” Module

4.2.1. Core Concepts :

HTTP Requests & Responses :- HTTP Requests and Responses in client module, you are allowed to send HTTP requests and handle the corresponding http responses. Allows different HTTP methods namely GET, POST, PUT and DELETE.

HTTP Server :- The server module allows you to set up a basic HTTP Server, which is able listen for incoming HTTP requests and respond. You can use this module to define request handlers that should be triggered by different kinds of HTTP requests.

4.2.2. Considerations :

Security :- HTTP is usually unencrypted communication. The data being transmitted should be encrypted using HTTPS (HTTP over SSL/TLS) to ensure secure communication with the RTP server.

Performance :- For production Web Servers, try for highly capable web frameworks or servers built to handle high traffics and complex requests.

4.3. The “urllib” Module

4.3.1. Core Concepts :

urllib.request :- URL opening and reading are managed by urllib.request. It provides many options for handling HTTP requests and receiving information from online sources.

urllib.parse :- The applications for handling and modifying URLs are supplied by urllib.parse. It use to break down URLs into their individual components (such as scheme, network location, and path) then reconstruct the parts into new URLs.

urllib.error :- Error classes for managing exceptions pertaining to URL parsing and fetching are contained in urllib.error.

4.3.2. Considerations :

Error Handling :- If you want to deal with issues like network failures, invalid URLs, or HTTP problems you ought to constantly handle exceptions related to URL access and analysis.

Data handling :- Make sure that the data obtained from URLs is correctly decoded and with regard to the content encoding.

5.Data Structures and Algorithms

5.1. Data Structures

Data structures are way to organize and manage the data efficiently. Python provides many built-in structures and allows for custom implementations.

5.1.1. Lists :-

Description :- Collections of elements which are structured and are able to modified. It may involve a many numbers of elements.

Description :- Operations including insertion, removal, sorting, appending, slicing, and indexing.

Use Cases :- Beneficial for dynamic sets of objects whose components require regular updates or access.

5.1.2. Tuples :-

Description :- Elements organized in an unchanged order. The contents is unchanging after they develop.

Operations :- concatenation, slicing, & indexing.

Use Cases :- Ideal for static data or method returning values, or other fixed-size collection where immutability is crucial.

5.1.3. Custom Data Structures :-

Stacks :- Always use to the Last In, First Out (LIFO) rule. implemented with custom classes or lists.

Queues :- Observe the First In, First Out (FIFO) rule while dealing with queues. apply with the aid of collections or lists.deque.

Linked lists :- Linked lists are made up of nodes that point to one another. beneficial for effective insertions and deletions and dynamic data.

Graphs and Trees :- While trees are networks of connected nodes, graphs are hierarchical structures. used in many different applications, including as network routing and hierarchical data representation.

5.2. Algorithms :-

Algorithms are step-by-step procedures for solving problems. They are important for efficient data processing and other operations.

5.2.1. Sorting Algorithms

Bubble Sort :- The simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Selection Sort :- It Selects the Smallest Element from unsorted And Places at the Begining.

Insertion Sort :- Insertion Sort is Build the sorted list one item at a time by inserting elements into their correct position.

Merge Sort :- Diving the list into halves, recursively sort each half,and then merge sorted parts.

Quick Sort :- Partitions the list using a pivot element and then recursively sort those partitions just like Merge Sort.

5.2.2. Searching Algorithms :-

Linear Search :- A linear search algorithm check the each element of an array or collections once after another, searching for equality as it proceeds.

Binary search :- It goes to half of the lower or upper part of the list then repeatedly searches for the center element.

5.2.3. Graph Algorithms :-

Depth First Search (DFS) :- It explores as far down along each branch before backtracking. Good for traversal and search.

Breadth First Search (BFS) :- It visits all the levels nodes by nodes. Shortest path oriented.

5.2.4. Dynamic Programming :-

Concept :- A technique for modifying time which includes split up a problem into smaller parts and storing the responses of sub-problems that overlap.

Use Cases :- It applies to the matrix chain multiplication, longest common subsequence (LCS), and knapsack problems.

6.Regular Expressions

Regular expressions is effective tools for pattern matching and text modifications. They give you the abilty to match, find, and interact with strings based patterns. Regular expressions are fully embraced by Python's re package.

6.1. Key Concepts

- **Basic Syntax :-**
  Literal characters :- Matches exact character. A matches character 'a'.
  Metacharacters :- Metacharacters are the special characters that have unique meanings in regex.
  “ . “ : Matches any character except a newline character.
  “ ^ “ : Matches the start of a string.
  “ $ “ : Matches the position is end of a string.
  “ * “ : Matches zero or more times of the previous element.
  “ + “ : Matches one or more occurrences of the preceding element.
  “ ? “ : Matches zero or one Repetition of the First Element.
  “ [] “ : Matches any one of the characters in square brackets [abc] means 'a', 'b' or 'c'.
  “ | “ : Acts as a logical OR. Example, ‘a|b’ : This will match either 'a' or 'b'.
  “ () “ : Matches multiple characters together as a single unit.
- **Character Classes :-** Predefined Character Classes:
  “ \d “ : It matches any digit [0-9]
  “ \D “ : Matches any non-digit character.
  “ \w “ : Matches any word character + quantifier for repetition one and more times
  “ \W “ : Matches any non-word character
  “ \s “ : Matches any Whitespace (space, tab, newline)
  “ \S “ : Matches any non-whitespace character.
- **Quantifiers :-**
  “ * “ : Zero or more occurrences.
  “ + “ : One or more occurrences.
  “ ? “ : Zero or one occurrence.
  “ {n} “ : Exactly n occurrences.
  “ {n,} “ : At least n occurrences.
  “ {n,m} “ : pattern occurs between n and m times

6.2. Common Operations

- **Search :-**

Purpose : The purpose of the search is to the find first instance of a pattern in string.

Function : re.search(pattern, string);

- **Match :-**

Purpose : Identify when the pattern starts with a similar letter as the string.

Function: re.match(pattern, string);

- **Find All :-**

Purpose : Identify each instance of a pattern inside a string.

Function : re.findall(pattern, string);

- **Substitute :-**

Purpose : Using a replacement string to replace string .

Function : re.sub(pattern, replacement, string);

- **Split :-**

Purpose : Split a string wherever a pattern occurs.

Function : re.split(pattern, string);

7.Debugging and Testing

7.1. Debugging

- Debugging is the process of finding and fixing errors in a code. When software does't work as expected, computer programmers study the code to determine that why any errors occurred.
- **Techniques for Debugging :-**

1. Print Statements :-

Description : Using print statements in the code to see what values variables are on controls where execution is currentNode This trick comes in handy when you just want to see what is going on through a certain part of the code.

Advantages : Very simple to use and no extra tools needed.

Disadvantages : Adds clutter, and some issues might require more specific fixes than this.

2. Interactive Debugging :-

Description : Pausing code execution at specific locations, pausing it with the help of debuggers and check value in each step interactingively. For this, Python has the built-in debugger ‘pdb’.

Advantages : Show detailed information about program state and execution flow.

Disadvantages : Application of debugging commands and may consume time.

3. Logging :-

Description :- Writing detail info of the program to log file or console output, during runtime. Logging module It enables you to write log messages in a structured way and at different levels (DEBUG, INFO, WARNING, ERROR, CRITICAL).

Advantages : Use to track and analyzing how a program runs; works for production environments.

Disadvantages : You have to setup and configure logging handlers, formatters as well.

7.2. Testing

1. Unit Testing :-

Description : Tests the separate parts or drives of code individually to check that they are working as required.

Advantages : Breaks the code in the pieces to streamline debugging.

Disadvantages : Does’t test interactions between components.

2. Functional Testing :-

Description : Functional testing is use to verify the software under test meets fundamental functional requirements and perform functions as expected.

Advantages : Verifies that the software is working as for an end-user perspective

Disadvantages : Some cases or interactions are not covered.

3. Acceptance Testing :-

Description : Makes sure the application satisfies the acceptance standards set by stakeholders or end-users. Mostly done by QA teams or users.

Advantages : The fulfillment of user requirements and expectations by this software.

Disadvantages : Needs well defined scenarios and acceptance criteria.

4. Regression Testing :-

Description : Verifies that current codebase additions or improvements did not affect any existing features.

Advantages : After an update, make sure that the present features continue stable and effective.

Disadvantages : Might require through coverage and be time-consuming.

8.Conclusion

In the end, developers believe the Python Standard Library as an actual game-changer. It covers so much environment, from complex tasks like networking and algorithm development to basic tasks like data storage and basic system operations. This library provides you with all the necessary tools for coding in a single accessible spot, similar to a Swiss Army knife. It allows you work more efficiently, saves time, and enhances the accuracy of your code. whatever your knowledge of expertise with Python, the Standard Library is an essential resource that will allow you to make informed decisions on almost any programming task.

9.References

- ➔ Zelle, John. (2013). Python Programming: An Introduction to Computer Science
- ➔ Python.org - 2024 - https://docs.python.org/3.9/.
- ➔ W3Schools, 2024 - https://www.w3schools.com/python/
- ➔ GeeksforGeeks – 2024 - https://www.geeksforgeeks.org/python-programming-language/
- ➔ DjangoProject.com – 2024 - https://docs.djangoproject.com/en/stable/
- ➔ Guido van Rossum, Barry Warsaw – 2023 - https://peps.python.org/pep-0008/

Github

https://github.com/ankit-vadadoriya/python_assignment