**PYTHON  
What is Python and how it works:**Python is a dynamic, interpreted language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

**Applications of Python:**

|  |  |
| --- | --- |
| Field | Framework |
| Web Development | Django CMS, Flask, Pyramid, FastApi |
| Scientific & Numeric | SciPy, Pandas, matplotlib |
| Data Analysis | NumPy |
| Machine Learning | Keras, Scikit-learn, PyTorch, TensorFlow |
| Game Development | PyGame, Panda3D, |
| CAD Apps | Used in Blender |
| Business Apps | Odoo, Tryton |
| Desktop GUI | PyQT, kvy |
| Web Scraping Apps | Scrapy, beautifulsoap |
| Automation | Selenium |
| Image Processing | OpenCV |

**Variables:  
  
Types of variables**

|  |  |
| --- | --- |
| Text Type: | Str |
| Numeric Types: | **int, float, complex** |
| Sequence Types: | **list, tuple, range** |
| Mapping Type: | **Dict** |
| Set Types: | **set, frozenset** |
| Boolean Type: | **Bool** |
| None Type: | **NoneType** |

**Syntax of different Types of Variables.**

|  |  |
| --- | --- |
| x = "Hello World" | str |
| x = 20 | **int** |
| x = 20.5 | **float** |
| x = 1j | **complex** |
| x = ["apple", "banana", "cherry"] | **list** |
| x = ("apple", "banana", "cherry") | **tuple** |
| x = range(6) | **range** |
| x = {"name" : "John", "age" : 36} | **dict** |
| x = {"apple", "banana", "cherry"} | **set** |
| x = frozenset({"apple", "banana", "cherry"}) | **frozenset** |
| x = True | **bool** |
| x = None | **NoneType** |

**Main Attributes of Each type:**

|  |  |  |
| --- | --- | --- |
| Code | Data Type | Main Features / Specs |
| x = "Hello World" | Str | Immutable. Ordered. Supports indexing & slicing. Holds text. |
| x = 20 | Int | Whole number. Unlimited length. Supports arithmetic ops. |
| x = 20.5 | Float | Decimal numbers. Approximation possible due to precision. |
| x = 1j | complex | For complex numbers. Format: real + imagj (e.g., 3 + 4j) |
| x = ["apple", "banana", "cherry"] | List | Mutable. Ordered. Allows duplicates. Indexable. Can mix data types. |
| x = ("apple", "banana", "cherry") | Tuple | Immutable. Ordered. Allows duplicates. Indexable. Faster than list. |
| x = range(6) | Range | Immutable sequence of numbers. Often used in loops. Lazy-evaluated. |
| x = {"name" : "John", "age" : 36} | Dict | Mutable. Unordered (as of Python 3.6+ it maintains insertion order). Key-value pairs. Fast lookups. |
| x = {"apple", "banana", "cherry"} | Set | Mutable. Unordered. No duplicates. Useful for membership tests & set operations. |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset | Immutable version of set. Hashable (can be dict keys). |
| x = True | Bool | Subclass of int (True = 1, False = 0). Logic operations. |
| x = None | NoneType | Represents absence of value. Used to initialize or reset variables, or return "nothing". |

**Github Commands:**

**Git has 4 stages**

* **Untracked** – The file is new and not part of Git’s history yet. Git is ignoring it until you add it.
* **Staged** – The file’s changes are marked to be included in the next commit.
* **Committed** – The file’s changes are saved in the local repository’s history.
* **Pushed** – The commit is uploaded from the local repository to a remote repository (e.g., GitHub).
* **Add Command:**This Stages the changes on local

Git add [file Paths]

**Example:**

Git add . //Dot represents current working directory

Git add folder/

Git add filename.txt filename2.txt folder2/

* **Commit Command:**  
  This saves the staged changes into the local repository’s history.  
  git commit -m "Commit message here"

**Example:**

git commit -m "Fixed login bug"

git commit -m "Added new feature in user profile"

* **Push Command:**  
  This uploads the committed changes from your local repository to the remote repository.  
  git push [remote-name] [branch-name]

Example:

git push origin main

git push origin feature/login