Unit:2 Overview

Python Basics: Complete Overview

Instructor: Musfique Ahmed

Introduction to Python

- Versatile programming language.
- Easy to learn and use.
- Supports multiple paradigms: procedural, object-oriented, and functional programming.
- Widely used in web development, data science, AI, and more.

Applications in Projects:

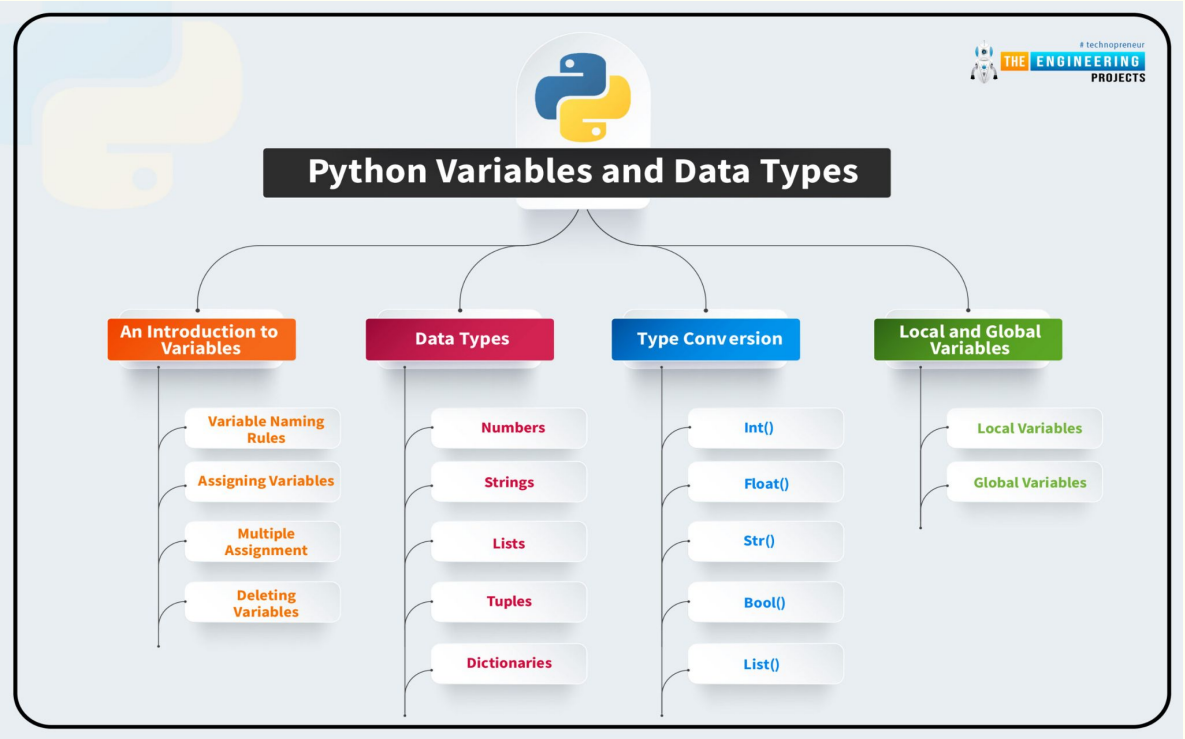
- Web application development.
- Automating repetitive tasks.
- Data analysis and visualization.

Basic Data Types and Casting

- Integers (`int`)
- Floating-point numbers (`float`)
- Strings (`str`)
- Booleans (`bool`)

Casting:

```
x = int("5")
y = float("3.14")
z = str(100)
```



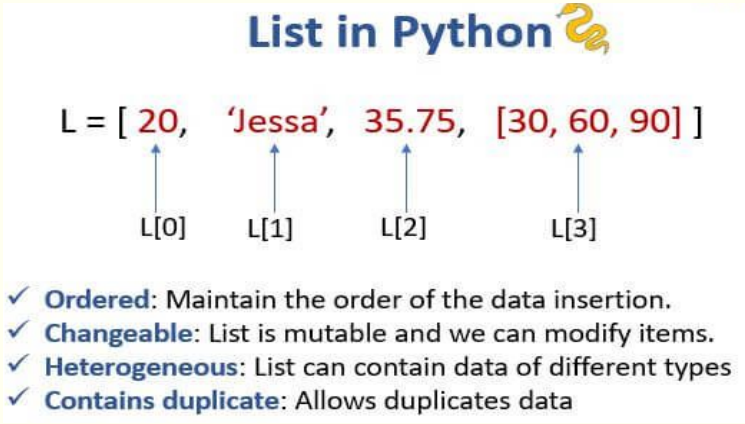
Applications in Projects:

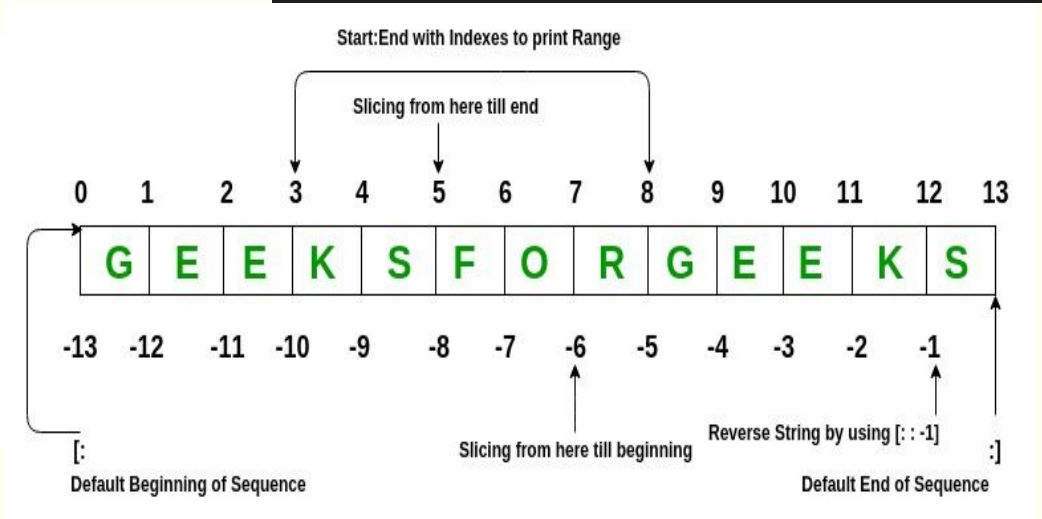
- Converting user input to appropriate types.
- Handling data from external sources like files.

Lists

- **Indexing:** Access elements using their position.
- **Adding/Removing Elements:** `append()`, `remove()`.
- **Modifying Elements:** Change values using indexing.
- **Looping:** Iterate through lists with `for`.
- **Nested Lists:** Lists inside lists.

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)
```





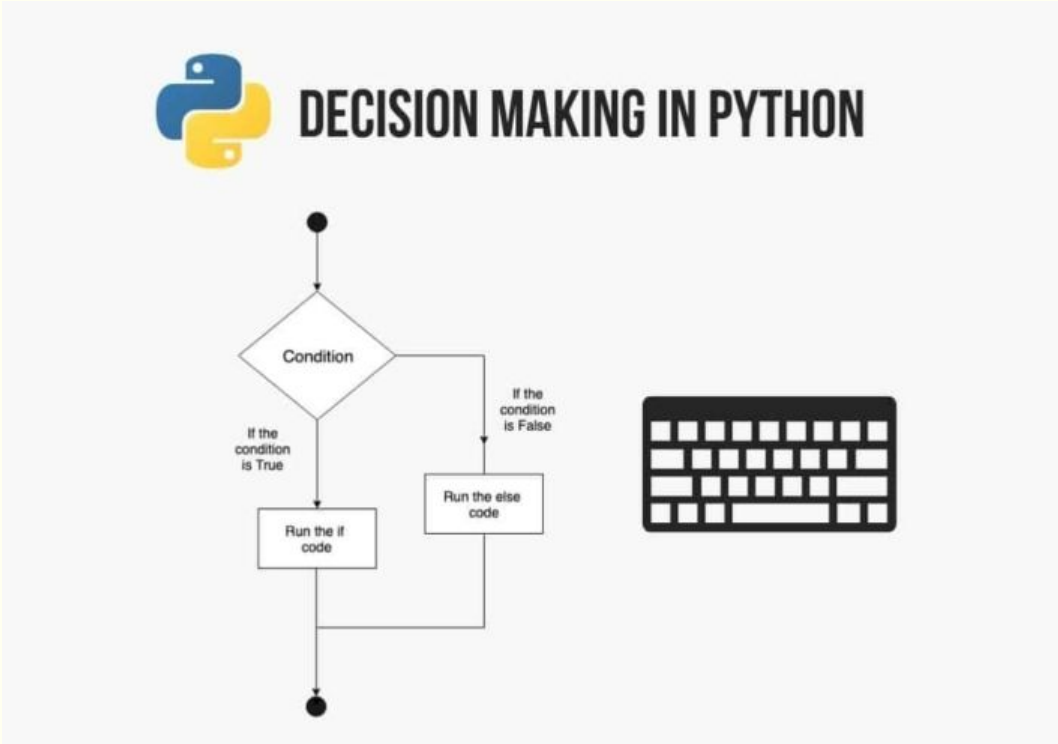
Applications in Projects:

- Storing multiple user inputs.
- Managing collections of data, e.g., shopping carts.

Conditionals

- `if` for conditions.
- `elif` for multiple conditions.
- `else` for default actions.

```
age = 18
if age < 18:
    print("Minor")
elif age == 18:
    print("Just Adult")
else:
    print("Adult")
```



Applications in Projects:

- Implementing user authentication.
- Creating decision-making algorithms.

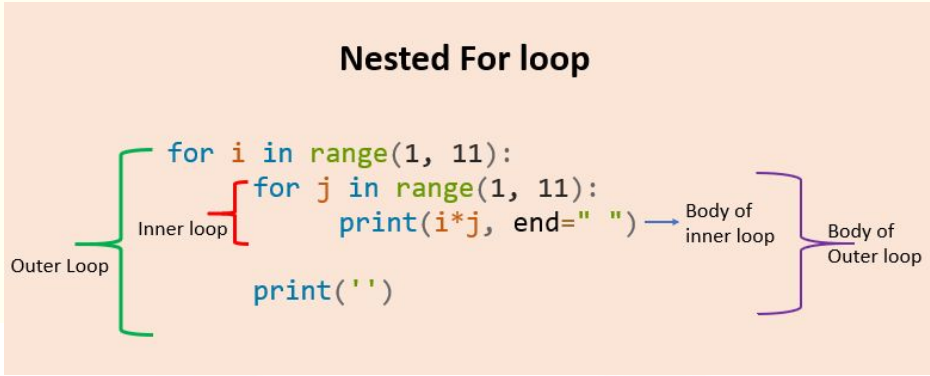
Loops

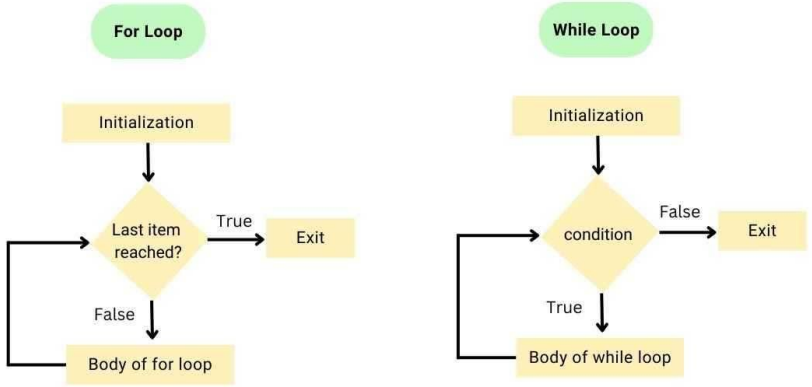
- `for`: Loop over a sequence.
- `while`: Loop until a condition is met.
- **Special Keywords:**
  - `break`: Exit loop early.
  - `continue`: Skip the current iteration.

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

Applications in Projects:

- Processing items in a queue.
- Automating repetitive tasks.





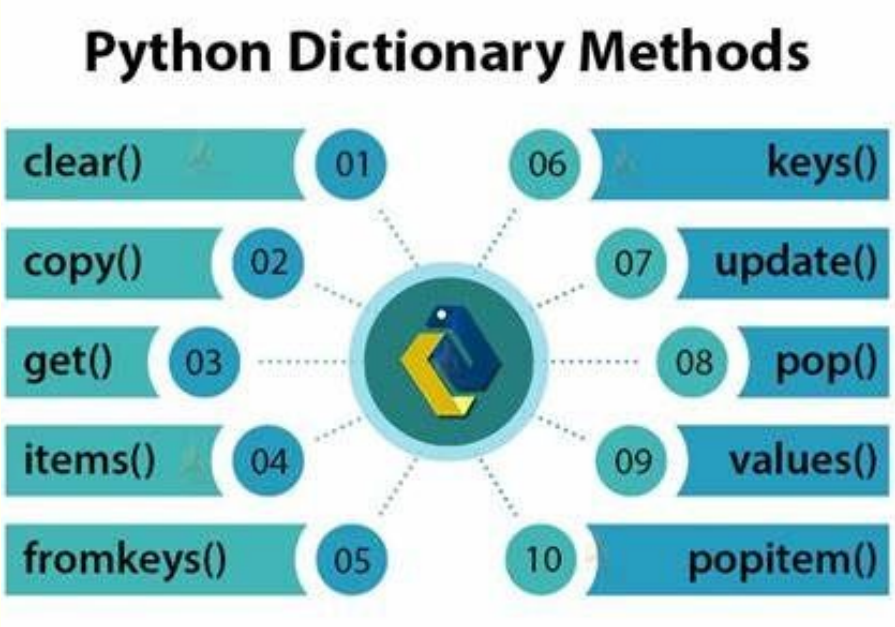
Dictionaries

- Key-value pairs.
- Updating: `dict[key] = value`.
- Deleting: `del dict[key]`.
- Looping through keys, values, or both.

```
person = {"name": "Alice", "age": 25}
person["age"] = 26
del person["name"]
```

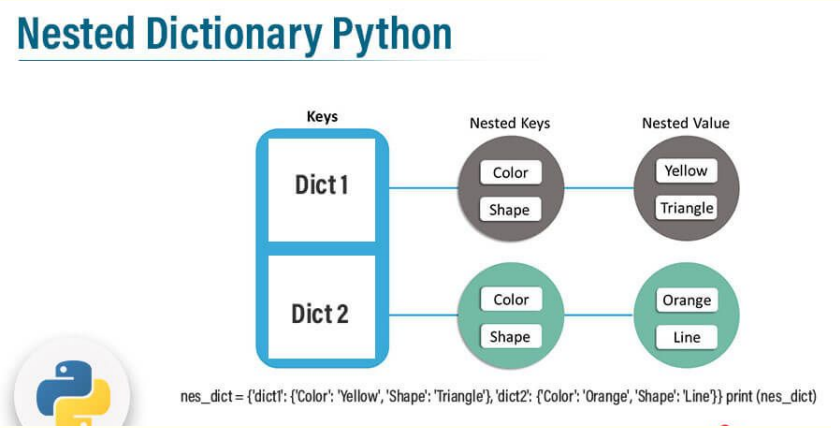
Applications in Projects:

- Storing user profiles or configurations.
- Mapping data to unique keys.



Nested Dictionaries

- Dictionaries within dictionaries.
- Access nested values using multiple keys.




```
students = {
    "101": {"name": "Alice", "grade": "A"},
    "102": {"name": "Bob", "grade": "B"}
}
print(students["101"]["name"])
```

```
[{'Name': 'Paras Jain',
  'Student': [{'Exam': 90, 'Grade': 'a'},
   {'Exam': 99, 'Grade': 'b'},
   {'Exam': 97, 'Grade': 'c'}]},
 {'Name': 'Chunky Pandey',
  'Student': [{'Exam': 89, 'Grade': 'a'}, {'Exam': 80, 'Grade': 'b'}]}]
```

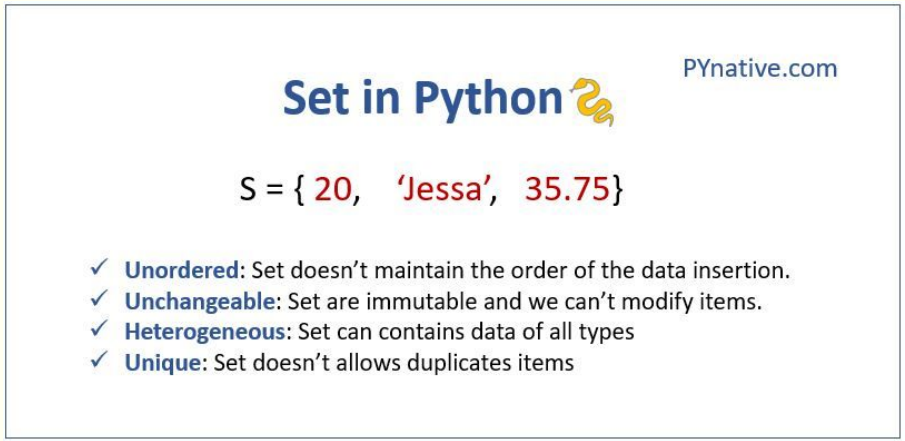
Applications in Projects:

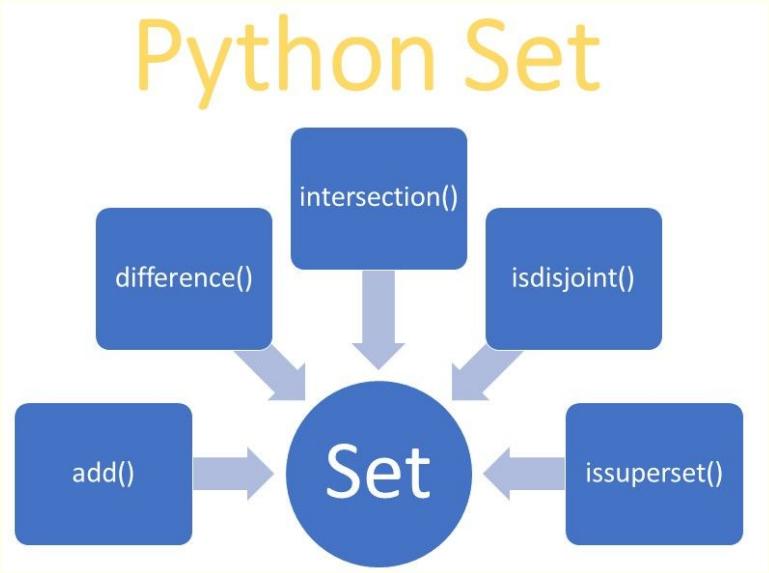
- Managing hierarchical data, e.g., organization charts.
- Handling complex configurations.

Sets

- No duplicate values.
- Operations: union, intersection, difference.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1 | set2)  # Union
```





Applications in Projects:

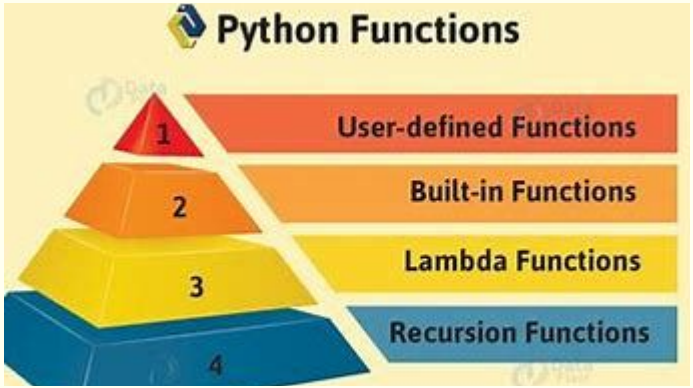
- Removing duplicate entries.
- Performing set operations like recommendations.

Functions

- Create functions using `def`.
- Use keyword and positional arguments.
- Built-in functions: `len()`, `sum()`, `max()`, etc.

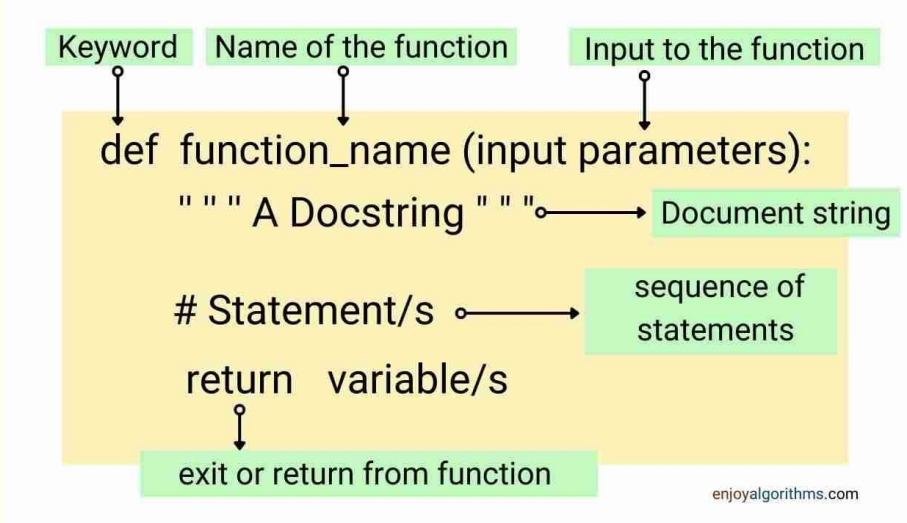
```
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
```



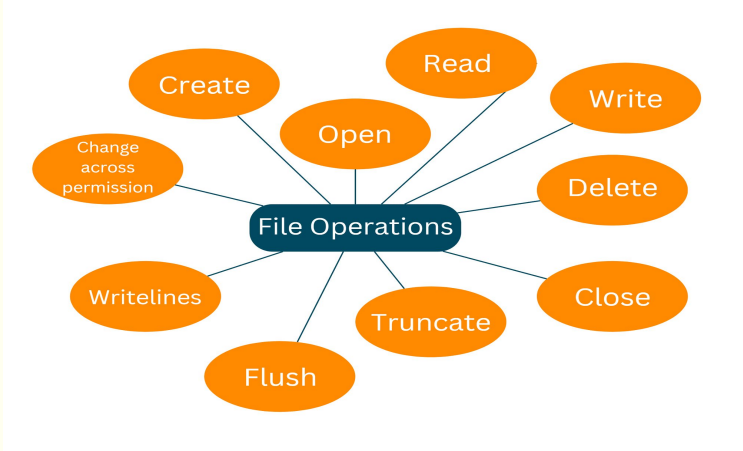
Applications in Projects:

- Modularizing code for better organization.
- Creating reusable utilities.



File Operations

- Open files with `open()`.
- Modes: `r` (read), `w` (write), `a` (append).
- Always close files or use `with`.



Applications in Projects:

- Storing user data.
- Logging application activities.

```
with open("example.txt", "w") as file:
    file.write("Hello, World!")
```

Modules and Packages

```
import math
print(math.sqrt(16))
```


- Import built-in modules: `math`, `os`, etc.
- Create custom modules.

Packages:

- Collection of modules in directories.
- Use `__init__.py` to initialize.

Applications in Projects:

- Extending application functionality.
- Structuring large codebases.

String Manipulation

- Common methods: `split()`, `strip()`, `replace()`.
- Slicing: Access substrings.
- Formatting: `f-strings` or `.format()`.

Applications in Projects:

```
text = "  Hello, World!  "
print(text.strip().replace("World", "Python"))
```

- Processing user input.
- Generating dynamic content for websites.

List Comprehensions

- Create lists in one line.

```
squares = [x**2 for x in range(5)]
print(squares)
```

Applications in Projects:

- Filtering data efficiently.
- Generating test datasets.

Python Debugging Tools

```
import pdb
pdb.set_trace()
```


- `print()` for simple debugging.
- `pdb` module for step-by-step debugging.

Applications in Projects:

- Fixing errors during development.
- Understanding complex code flows.

Logical and Membership Operators

- Logical: `and`, `or`, `not`.
- Membership: `in`, `not in`.

```
nums = [1, 2, 3]
print(2 in nums)  # True
```

Applications in Projects:

- Building search functionalities.
- Creating conditional logic for games.

Summary

- Core Python concepts from data types to debugging.
- Hands-on examples for real-world applications.
- Strong foundation for advanced programming.

Thank You

Do the Quiz Please, you have 10 minutes to do that!