Data Types:

In Python, data types represent the kind of value a particular variable can hold. Some of the fundamental data types in Python include:

**Integers (int):** Whole numbers without any decimal point. Example: 5, -10, 1000.

**Floating-point numbers (float):** Numbers that have a decimal point. Example: 3.14, 2.71828, -8.0.

**Strings (str):** Ordered sequence of characters enclosed in single (') or double (") quotes. Example: "Hello, World!", 'Python', '123'.

**Booleans (bool):** Represents truth values True or False. Often used for logical operations and conditions.

**Lists (list):** Ordered and mutable collections of items. Items can be of different data types and are enclosed in square brackets []. Example: [1, 2, 3], ['apple', 'banana', 'cherry'].

**Tuples (tuple):** Ordered and immutable collections of items enclosed in parentheses (). Example: (1, 2, 3), ('a', 'b', 'c').

**Dictionaries (dict):** Unordered collection of key-value pairs enclosed in curly braces {}. Example: {'name': 'Alice', 'age': 25, 'city': 'New York'}.

**Sets (set):** Unordered and mutable collections of unique items enclosed in curly braces {}. Example: {1, 2, 3}, {'apple', 'orange', 'banana'}.

Python also supports type conversion functions like int(), float(), str(), list(), tuple(), dict(), set(), etc., which allow conversion between different data types when possible.

Here's an example showcasing different data types:

# Integers and Floats

age = 25

height = 5.9

# Strings

name = "Alice"

greeting = 'Hello, World!'

# Booleans

is\_student = True

is\_working = False

# Lists

numbers = [1, 2, 3, 4]

fruits = ['apple', 'banana', 'cherry']

# Tuples

coordinates = (10, 20)

colors = ('red', 'green', 'blue')

# Dictionaries

person = {'name': 'Bob', 'age': 30, 'city': 'London'}

# Sets

unique\_numbers = {1, 2, 3, 4, 5}

unique\_letters = {'a', 'b', 'c'}

Conditional Statement:

Conditional statements in Python are used to perform different actions based on whether a certain condition evaluates to True or False. The main types of conditional statements in Python are the if, else, and elif statements.

# If-else statement

age = 20

if age >= 18:

print("You are an adult.") # Code to execute if the condition is true

else:

print("You are a minor.")

#if-elif-else statement

x = 10

if x > 10:

print("x is greater than 10") # Code to execute if condition1 is true

elif x < 10:

print("x is less than 10") # Code to execute if condition2 is true

else:

print("x is equal to 10") # Code to execute if all conditions are false

Functions:

Functions in Python are blocks of organized, reusable code designed to perform specific tasks. They provide a way to break down complex problems into smaller, more manageable parts, making code modular, readable, and easier to maintain.

Functions are defined using the def keyword followed by the function name and parentheses containing optional parameters. They may or may not return a value.

Example:

def greet():

print("Hello, there!")

Lambda function:

A lambda function in Python is a small, anonymous function defined using the lambda keyword. Unlike regular functions created using the def keyword, lambda functions can be created quickly and don't require a formal def statement or a unique name.

Example:

add = lambda x, y: x + y

result = add(3, 5)

Loops in python:

Loops in Python are control structures that allow you to execute a block of code repeatedly. They help automate repetitive tasks and iterate over sequences like lists, tuples, dictionaries, strings, and more. There are two primary types of loops in Python: for loops and while loops.

1. **for loop:** for loops iterate over a sequence of elements like lists, tuples, strings, dictionaries, etc.

Example:

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

1. **while loop:** while loops iterate until a certain condition becomes False.

Example:

i = 0

while i < 5:

print(i)

i += 1

Using break and continue:

i = 0

while i < 10:

if i == 5:

break # Stops the loop when i equals 5

i += 1

if i % 2 == 0:

continue # Skips even numbers

print(i)