Python:
Python is a popular, high-level, general-purpose programming language known for its clear, readable syntax that makes it easy to learn and use for tasks ranging from web development and data science to artificial intelligence and task automation. It's a versatile, open-source language supported by a vast community and an extensive ecosystem of libraries and frameworks, allowing developers to create diverse applications efficiently and on multiple platforms.
Simple program:

```
print('Hello, world!')
print('python is programming langauage')
```

User Input :

```
name = input("What is your name?\n")
print("Hello" + format(name))
time=input("Enter the appointment time")
print(time)
```

Data types and Variable Declaration:
In Python, there is no explicit variable declaration syntax. A variable is created the moment a value is assigned to it. The assignment uses the syntax:

python

variable_name = value

Key points about Python variable declaration:

- Variables do not need a declared type; Python infers the type from the assigned value.
- Variable types can change dynamically during the program.
- Variable names can contain letters, digits, and underscores but cannot start with a digit.
- Variable names are case-sensitive.
- You can assign any Python object to a variable, including numbers, strings, lists, objects, etc.

Examples:

```
x = 10          # x is an integer
x = "hello"     # now x is a string
name = "John"   # string variable
pi = 3.14       # float variable
```

To check the type of a variable:

python

print(type(x))

This flexibility allows for simple and dynamic variable use in Python without explicit declaration commands.

Practice : Addition of 2 numbers

```
# Ask the user to enter the first number
num1 = float(input("Enter first number: "))

# Ask the user to enter the second number
num2 = float(input("Enter second number: "))

# Calculate the sum of the two numbers
sum = num1 + num2

# Display the result
print("The sum of", num1, "and", num2, "is", sum)
```

Practice : Swap 2 Numbers

```
# Get two numbers from the user
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

print("Numbers before swapping:")
print("First number:", num1)
print("Second number:", num2)

# Swap using tuple unpacking
num1, num2 = num2, num1

print("Numbers after swapping:")
print("First number:", num1)
print("Second number:", num2)
```

Data Types in Python :

Python has several built-in data types to handle various kinds of data. The core data types include:

- Numeric Types: int (integer numbers), float (floating-point numbers), and complex (complex numbers).
- Text Type: str for strings.
- Sequence Types: list, tuple, and range which hold collections of items.
- Mapping Type: dict which holds key-value pairs.
- Set Types: set and frozenset for unique collections.
- Boolean Type: bool representing True or False.
- Binary Types: bytes, bytearray, and memoryview for binary data.
- None Type: None Type representing the absence of value.

Python is dynamically typed, so the data type is determined when a value is assigned, and you can check the type of a variable using the type() function.

These data types allow Python to store and manipulate different forms of data efficiently and enable diverse programming paradigms such as numeric calculations, text processing, sequence handling, and more.

Datatype Conversion in Python :

```
# Integer example
x = 10
print(x)          # Output: 10
print(type(x))    # Output: <class 'int'>

# Float example
y = 3.14
print(y)          # Output: 3.14
print(type(y))    # Output: <class 'float'>

# String example
z = "Hello, Python!"
print(z)          # Output: Hello, Python!
print(type(z))    # Output: <class 'str'>

# Type conversion examples
```

```
# Convert integer to float
a = float(x)
print(a)          # Output: 10.0
print(type(a))    # Output: <class 'float'>

# Convert float to integer (decimal part truncated)
b = int(y)
print(b)          # Output: 3
print(type(b))    # Output: <class 'int'>

# Convert string to integer (string must represent a valid integer number)
c = int("42")
print(c)          # Output: 42
print(type(c))    # Output: <class 'int'>

# Convert string to float (string must represent a valid float number)
d = float("3.14159")
print(d)          # Output: 3.14159
print(type(d))    # Output: <class 'float'>
```

Operators in python

Python operators are special symbols used to perform operations on variables and values. They are categorized into several types based on the operation they perform:

- **Arithmetic operators**: Perform basic mathematical calculations such as addition (+), subtraction (-), multiplication (*), division (/), modulus (%), exponentiation (**), and floor division (//).
- **Assignment operators**: Assign values to variables, including simple assignment (=) and compound assignments like +=, -=, *=, /=, %=, etc.
- **Comparison operators/Relational operators:** Compare values and return Boolean results such as equal (==), not equal (!=), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=).
- **Logical operators**: Used for logical operations like and, or, and not.
- **Identity operators**: Check if two operands refer to the same object (is, is not).

- **Membership operators**: Test if a value exists in a sequence (in, not in).
- **Bitwise operators**: Perform bit-level operations like AND (&), OR (|), XOR (^), NOT (~), left shift (<<), and right shift (>>).

Arithmetic operators:

```
a = 15
b = 4
print(a + b)   # Addition: 19
print(a - b)   # Subtraction: 11
print(a * b)   # Multiplication: 60
print(a / b)   # Division: 3.75
print(a // b)  # Floor Division: 3
print(a % b)   # Modulus: 3
print(a ** b)  # Exponentiation: 50625
```

Assignment operators example:

```
x = 5
x += 3  # Equivalent to x = x + 3
```

Comparison operator example:

```
x = 10
y = 20
print(x < y)  # True
```

Logical operator example:

```
a = True
b = False
print(a and b)  # False
```

Membership Operators:

```
list1 = [1, 2, 3, 4, 5]
print(3 in list1)     # True
print(7 not in list1) # True

string1 = "Hello World"
print('H' in string1) # True
print('z' not in string1) # True
```

```
dict1 = {1: "A", 2: "B"}
print(1 in dict1)     # True (checks for key presence)
```

These operators are fundamental in Python programming to manipulate data and control program flow effectively.

Conditions in Python

Conditions use if, elif, and else statements to execute code blocks based on Boolean expressions. A condition evaluates to True or False, and the related block runs only if the condition is True. For example:

```
age = 20
if age >= 18:
    print("Eligible to vote.")
else:
    print("Not eligible to vote.")
```

Python also supports shorthand forms like the ternary operator for concise conditional expressions:

```
res = "Pass" if marks >= 40 else "Fail"
```

Loops in Python

Python mainly provides two types of loops:

- While loop: Repeats a block of code as long as the given condition is true.
- For loop: Iterates over a sequence (like a list, tuple, or string) or range of numbers.

While loop syntax and example:

```
count = 0
while count < 3:
    print("Hello Geek")
    count += 1
```

The loop runs as long as the condition is true; once the condition becomes false, the loop terminates, optionally followed by an else block that runs if the loop completes normally without a break:

```
count = 0
while count < 3:
    print("Hello Geek")
    count += 1
else:
    print("Loop finished")
```

For loop example to iterate over a list:

```
for num in [1, 2, 3]:
    print(num)
```

Python supports break to exit loops early and continue to skip the current loop iteration and proceed to the next one.