**Python is used widely across many industries and by some of the largest organizations in the world due to its simplicity, readability, extensive libraries, and strong community support.**

1. **Tech Companies**

**Google**

**Facebook/Meta**

**Instagram:**

1. **Financial Services**

**JPMorgan Chase**

**Goldman Sachs**

**Citigroup**

1. **E-commerce & Retail**

**Amazon**

**eBay**

**Walmart**

**4. Media & Entertainment**

**Netflix**

**Disney**

**Industrial Light & Magic (Lucasfilm)**

**5. Automotive & Manufacturing**

**Tesla**

**BMW**

**6. Aerospace & Defense**

**NASA**

**Boeing**

**7. Education & Research**

**MIT, Stanford, Harvard**

**Coursera, edX, Udemy**

**8. Healthcare & Pharma**

**Pfizer, Novartis**

**GE Healthcare**

**Variables**

In Python, variables are used to store data that can be used and manipulated throughout a program. Python is dynamically typed, meaning you don't need to declare the type of a variable explicitly

Declaring Variables

name = "Amit" # string

age = 30 # integer

height = 5.9 # float

is\_active = True # boolean

Variable Naming Rules

* Can include letters, numbers, and underscores.
* Must start with a letter or an underscore.
* Cannot start with a number.
* Cannot use reserved keywords (if, for, class, etc.).

**Dynamic Typing**

You can change the type of a variable anytime

x = 5 # x is int

x = "hello" # now x is str

Assigning Multiple Variables

a, b, c = 1, 2, 3

x = y = z = 0 # same value for all

**Operands and expressions**

For example, in the expression x + y

* x and y are **operands**
* + is the **operator**

Operands can be:

* **Literals**: 5, 10.2, 'hello'
* **Variables**: a, price, name
* **Data structures**: [1, 2], { 'key': 'value' }, etc.

Operator can be:

|  |  |  |
| --- | --- | --- |
| **Type** | **Operators** | **Example** |
| Arithmetic | +, -, \*, /, //, %, \*\* | a + b |
| Comparison | ==, !=, <, >, <=, >= | x < y |
| Logical | and, or, not | a and b |
| Assignment | =, +=, -=, etc. | x += 5 |
| Bitwise | &, ` | , ^, ~, <<, >>` |
| Membership | in, not in | 'a' in text |
| Identity | is, is not | a is b |

Expressions

combination of **operands** and **operators**

Example:-

a + b # arithmetic expression

x \* (y - 3) # expression with parentheses

len(name) # function call expression

a > b # comparison expression (evaluates to True/False)

not is\_valid # logical expression

**Conditional statements**

Conditional statements allow you to execute different blocks of code depending on whether a condition is True or False.

1. **if Statement**

x = 10

if x > 5:

print("x is greater than 5")

1. **if-else Statement**

x = 3

if x > 5:

print("x is greater than 5")

else:

print("x is 5 or less")

1. **if-elif-else Statement**

x = 10

if x < 0:

print("x is negative")

elif x == 0:

print("x is zero")

else:

print("x is positive")

**4.Nested if Statements**

x = 15

if x > 10:

if x < 20:

print("x is between 10 and 20")

**5. Conditional Expressions (Ternary Operator)**

Used to write simple if-else statements in a single line.

x = 5

result = "Even" if x % 2 == 0 else "Odd"

print(result)

**Comparison Operators Used in Conditions:**

== : equal to

!= : not equal to

> : greater than

< : less than

>= : greater than or equal to

<= : less than or equal to

**Logical Operators:**

* and: both conditions must be true
* or : at least one condition must be true
* not: reverses the truth value

x = 7

if x > 5 and x < 10:

print("x is between 5 and 10")

**Loops**

1. for loop:- Used for iterating over a sequence like a list, tuple, dictionary, string, or a range.

Example

for i in range(5):

print(i)

1. while loop:- Repeats as long as a condition is True.

Example

i = 0

while i < 5:

print(i)

i += 1

i = 0

while i < 5:

i = i + 1

print(i)

1. Nested Loops:- Loops inside loops. Both for and while can be nested.

Example

for i in range(3):

for j in range(2):

print(i, j)