1. Key Features of Python

Python was developed by Guido Van Rossum in 1991.

- **Readability and Simplicity:** Python's syntax is straightforward, highly readable and easy to understand.
- **Dynamically typed:** Python is dynamically typed, that means no need to declare variable types explicitly.
- **Free and Open Source:** Python is free to use, which means you don't have to pay anything to download, install or use it.
- **Platform Independent:** Python is considered as platform independent, which means you can write Python code on one operating system and run it on another without modification.
- **Interpreted Language:** Python is an interepreted language, you write code and the interpreter executes iyt line by line.

Why Python?

- Widely used in the industry
- Data industry
- A lot of Libraries
- Work on frontend, backend, data analysis
- Automation
- Image processing

---

2. Role of Predefined Keywords in Python

Keywords are predefined words that hold a special meaning and have specific purpose in Python. We cannot name a variable which is a Keyword. **Some Predefined Keywords include:**

- **False :** is a boolean value which returns 0
- **None :** returns null value
- **True :** is a boolean value which returns 1
- **and :** combines 2 conditions
- **break :** moves the control to the end of the loop
- **import :** is used to import packages/modules/libraries which is pre-built.

---

3. Mutable and Immutable Objects

- **Mutable Objects : ** Objects/Containers whose state or value can be changed after they are created are called as Mutable Objects or container. Examples : Lists and Dictionaries.

```
list = [1,2,3,True,"Ajay]
list[4]="Bijay"
Output : [1,2,3,True,"Bijay"]
```

- ** Immutable Objects : ** Objects/Containers whose state or value cannot be changed after they are created are called as Immutable Objects or container. Examples : Tuples and String

```
b = "Ajay"
b[0] = 'A'     b[-1] = 'y'
b[1] = 'j'     b[-2] = 'a'
b[2] = 'a'     b[-3] = 'j'
b[3] = 'y'     b[-4] = 'A'
```

---

4. Operators in Python

Operators are special keywords/symbols that are used to perform operations on values or variables.

- **Arithmetic Operators:**
    1. Addition (+)
    2. Subtraction (-)
    3. Multiplication (*)
    4. Division (/)
    5. Modulus (%)
    6. Exponentiation (**)
    7. Floor Division (//)
- **Comparison Operators:**
    1. Equal to (==)
    2. Not equal to (!=)
    3. Greater than (>)
    4. Less than (<)
    5. Greater than or equal to (>=)
    6. Less than or equal to (<=)
- **Logical Operators:**
    1. Logical AND
    2. Logical OR
    3. Logical NOT
- **Assignment Operators:**
    1. Assignment (=)
    2. Addition Assignment (+=)
    3. Subtraction Assignment (-=)
    4. Multipllication Assignment (*=)
    5. Division Assignment (/=)
- **Membership Operators:**
    1. in
    2. not in
- **Identity Operators:**

1. is
2. is not

- **Bitwise Operators:**
  1. AND (&)
  2. OR (|)
  3. NOT (~)
  4. XOR (^)
  5. Left Shift (<<)
  6. Right Shift (>>)

---

5. Type Casting

The process of changing the data type of a value/object is called Type Casting. Types of Type casting:

1. **Implicit Type Conversion:** Python converts the data type into another data type automatically.

```
a = 7
b = 3.0
c = a + b //10.0
d = a * b //21.0
```

2. **Explicit Type Conversion:** Requires user involvement to convert a variable's data type to the desired type.

```
a = 5
n = float(a) //5.0
a = 5.9
n = int(a) //5
```

---

6. Conditional Statements

It helps you to code decisions based on some preconditions.

1. if:

```
a = 33
b = 200
if b > a:
 print("b is greater than a")
```

2. elif:

```
a = 33
b = 33
```

```
if b > a:
 print("b is greater than a")
elif a == b:
 print("a and b are equal")
```

3. else:

```
a = 200
b = 33
if b > a:
 print("b is greater than a")
elif a == b:
 print("a and b are equal")
else:
 print("a is greater than b")
```

4. Nested If:

```
x = 41
if x > 10:
 print("Above ten")
 if x > 20:
     print("and also above 20")
 else:
     print("but not above 20")
```

---

7. Loops

Loop statement allows you to execute a block of code repeatedly.

1. while:

```
n = 7
i = 1
while i < n:
 print(i)
 i = i + 1
```

2. for:

```
a = "pwskills"
for i in a:
 print(i)
//Range:
```

```
for i in range(10):
 print(i, end= " ")
```

3. break:

```
n = 7
i = 1
while i < n:
 print(i)
 i = i + 1
 if i == 3:
     break
else:
 print("This code will be executed when the while is run successfully without any break")
```

4. continue:

```
n = 7
i = 1
while i < n:
 i = i + 1
 if i == 3:
     continue
 print(i)
else:
 print("This code will be executed when the while is run successfully without any break")
```