Isha Dhiman

AI/ML Enthusiast | Intern @CodroidHub | First-Year Engineering Student

# Artificial Intelligence/Machine Learning UpSkills Notebook

From Basics to Real-World — Starts Your ML Journey

## Basic Fundamentals of Python

Welcome! This notebook will help you learn the basics of Python step by step. Run each code cell, see the output, and try changing the examples to learn by doing!

### Variables

- Variables is a memory location where we can store the data.
- The value inside a variable can be changed at any time — that's why it's called a variable.

In [32…
```
name = "Isha"
age = 21
name
```

Out[32]:
```
'Isha'
```

In [33…
```
age
```

Out[33]:
```
21
```

### Data Types

- Data types tell a program what kind of value is stored in a variable.
- They help decide what actions can be done with that value.

In [34…
```
x = 5
y = 3.14
z = "Hello"
is_fun = True

print(type(x), type(y), type(z), type(is_fun))
```

```
<class 'int'> <class 'float'> <class 'str'> <class 'bool'>
```

### Printing Output

Use `print()` to show results.

In [35…
```
print("My name is", name)
print("I am", age, "years old.")
```

```
My name is Isha
I am 21 years old.
```

### Comments

- Use comments to explain your code.
- Comments increases the readability of your code.

In [36…
```
# This is a comment
2 + 3
```

Out[36]: 5

### Types of Comments

- Single-line comment : Starts with #.

In [37…
```
# This is a single-line comment
```

- Multi-line comment: Use triple quotes (''' ... ''' or """ ... """).

In [ ]:
```
'''
This is a multi-line comment
'''
```

## Input & Type Conversion

Use `input()` to get user input. Use `int()` to convert string to number.

In [29…
```
# Uncomment to test in a real notebook
user = input("What is your name? ")
print("Hello,", user)
```

```
Hello, Isha
```

In [30…
```
type(_)
```

Out[30]: str

## Operators

An operator is a symbol or keyword that performs an action on one or more values (called operands) to produce a new result.

### Types of Operators

**1️⃣Arithmetic Operators**

Used to perform basic mathematical operations like addition, subtraction, multiplication, etc.

Operators:

| Operator | Meaning | Example |
|---|---|---|
| `+` | Addition | `5 + 2` → `7` |
| `-` | Subtraction | `5 - 2` → `3` |
| `*` | Multiplication | `5 * 2` → `10` |
| `/` | Division | `5 / 2` → `2.5` |
| `%` | Modulus (remainder) | `5 % 2` → `1` |
| `**` | Exponent (power) | `5 ** 2` → `25` |
| `//` | Floor Division | `5 // 2` → `2` |

✅ **Example:**

In [39…
```
a = 10
b = 3

print(a + b)   # 13
print(a % b)   # 1
```

```
13
1
```

**2️⃣Assignment Operators**

Used to assign a value to a variable, and sometimes perform an operation while assigning.

Operators:

| Operator | Meaning | Example |
|---|---|---|
| `=` | Simple assignment | `x = 5` |
| `+=` | Add and assign | `x += 2` (same as `x = x + 2`) |
| `-=` | Subtract and assign | `x -= 2` |
| `*=` | Multiply and assign | `x *= 2` |
| `/=` | Divide and assign | `x /= 2` |
| `%=` | Modulus and assign | `x %= 2` |
| `**=` | Exponent and assign | `x **= 2` |
| `//=` | Floor divide and assign | `x //= 2` |

✅ **Example:**

In [40…
```
x = 5
x += 3  # x becomes 8
print(x)
```

```
8
```

3️⃣ **Comparison Operators**

Used to compare two values and return a Boolean (True or False).

Operators:

| Operator | Meaning | Example |
|---|---|---|
| `==` | Equal to | `5 == 5` → `True` |
| `!=` | Not equal to | `5 != 3` → `True` |
| `>` | Greater than | `5 > 3` → `True` |
| `<` | Less than | `5 < 3` → `False` |
| `>=` | Greater than or equal to | `5 >= 5` → `True` |
| `<=` | Less than or equal to | `5 <= 3` → `False` |

✅ **Example:**

In [41…
```
a = 10
b = 5

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

```
True
False
```

4️⃣ **Logical Operators**

Used to combine multiple conditions and return a Boolean result.

Operators:

| Operator | Meaning | Example |
|---|---|---|
| `and` | True if both conditions are true | `(5 > 2) and (5 < 10)` → `True` |
| `or` | True if at least one condition is true | `(5 > 10) or (5 < 10)` → `True` |
| `not` | Reverses the result | `not(5 > 2)` → `False` |

✅ **Example:**

In [42…
```
x = 5

print(x > 2 and x < 10)   # True
print(not(x > 2))         # False
```

```
True
False
```

5️⃣ **Membership Operators**

Used to check if a value is present in a sequence (like a list, string, or tuple).

Operators:

| Operator | Meaning | Example |
|---|---|---|
| `in` | True if value is found | `'a' in 'apple'` → `True` |
| `not in` | True if value is not found | `'b' not in 'apple'` → `True` |

✅ **Example:**

In [43…
```
text = "hello"

print('h' in text)      # True
print('x' not in text)  # True
```

```
True
True
```

6️⃣ **Identity Operators**

Used to compare the memory location of two objects (whether they are actually the same object in memory).

Operators:

| Operator | Meaning | Example |
|---|---|---|
| `is` | True if both refer to the same object | `a is b` |
| `is not` | True if not the same object | `a is not b` |

✅ **Example:**

In [44…
```
a = [1, 2, 3]
b = a
c = [1, 2, 3]

print(a is b)     # True (same object)
print(a is c)     # False (same value, different object)
```

```
True
False
```

# Conditions

Conditions are statements that check whether something is true or false in a program.

They help the program make decisions and choose what to do next.

Conditions are used in if, elif, and else statements to run different blocks of code based on whether the condition is true.

They often use comparison operators (like >, <, ==) and logical operators (like and, or, not) to form meaningful checks.

✅ **Example:**

In [45…
```
age = 18

if age >= 18:
    print("You are an adult.")
else:
    print("You are not an adult.")
```

```
You are an adult.
```

In [46…
```
num = 4
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

```
Even
```

# Loops

Loops are instructions that tell a program to repeat a block of code multiple times.

They help you run the same code again and again, either a fixed number of times or until a condition is no longer true.

Loops save time and make programs shorter because you don't have to write the same instructions many times.

Loops are often used for repeating tasks, processing lists, or checking data one item at a time.

### Types of Loops:

- **for loop** — repeats a block of code for each item in a sequence (like a list, tuple, or string) or for a range of numbers.

✅ **Example of for loop:**

In [47…
```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

```
apple
banana
cherry
```

- **while loop** — repeats a block of code as long as a condition is true.

✅ **Example of while loop:**

In [48…
```
count = 1

while count <= 5:
    print(count)
    count += 1
```

```
1
2
3
4
5
```

## Functions

A **function** is a block of reusable code that does a specific task.

It can take input (called **parameters**) and can return an output.

### Types of Functions in Python

1. **Built-in Functions** These come with Python.

   Example: `print()`, `len()`, `type()`

2. **User-defined Functions** These are created by you using `def`.

3. **Lambda Functions (Anonymous Functions)** These are small, one-line functions.

In [50…
```
def greet(name):
    return "Hello, " + name + "!"

greet("Isha")
```

Out[50]:
```
'Hello, Isha!'
```

## Summary

- Python is simple yet powerful.
- Variables store data.
- Data types include numbers, strings, collections.
- Operators perform actions.
- Conditionals and loops control flow.
- Functions make code reusable.
- Built-in functions make tasks easy.

**Connect @**
Mail (Isha Dhiman): dhimanisha177@gmail.com
Contact: +91-9253165167

# Isha Dhiman

AI/ML Enthusiast | Intern @CodroidHub | First-Year Engineering Student

Isha Dhiman