

Python Programming – Unit 1

Piyush Pant



outline

1.1 Python Introduction

1.2 Data Types and Type Conversion

1.3 Comments

1.4 Variables, Constants, Operators and Performing
Calculations

1.5 Reading Input from Keyboard

1.6 Print function, Displaying Formatted Output with Fstrings

1.1 What is Python?



High-level, interpreted programming language



Designed by Guido van Rossum in 1991



Open-source and community-driven



Great for beginners and professionals

1.1 Features of Python



Simple and readable syntax



Dynamically typed (program automatically detects and assigns data type at runtime based on assigned value)



Portable and cross-platform



Large standard library



Supports OOP, functional, and procedural styles



1.1 Use Cases of Python



Web development (Flask, Django)



Data Science (Pandas, NumPy)



Machine Learning (scikit-learn, TensorFlow)



Scripting and
Automation (pyautogui)



Game Development (Pygame)

1.1 Advantages Over Other Languages



Less boilerplate code



Rapid development



Extensive third-party libraries



Strong community support

1.1 Hello World Example

Example:

```
print('Hello, world!')
```

- Simple syntax
- No need to define main function or semicolons

1.2 Built-in Data Types

- Numeric: int, float, complex
- Text: str
- Boolean: bool
- Sequence: list, tuple, range
- Set types: set, frozenset
- Mapping: dict

1.2 Numeric and String Types

Examples:

```
x = 5    # int
```

```
y = 3.14 # float
```

```
z = 'Hi'  # str
```

Used in calculations and text processing

1.2 Sequence Types

Examples:

```
lst = [1, 2, 3] # list
```



```
tpl = (1, 2, 3) # tuple
```

```
rng = range(1, 4) # range
```

Useful for storing multiple values

```
list(range(1,4))
```

```
[1,2,3]
```

Feature	List	Tuple
Mutability	 Mutable Once created can change (add or remove elements)	 Immutable
Syntax	[1, 2, 3]	(1, 2, 3)
Speed	Slower	Faster
Methods	More (append, remove, etc.)	Fewer (count, index)
Use Cases	Dynamic data	Fixed collections

1.2 Set and Mapping Types

Examples:

```
s = {1, 2, 3}      # set
```

```
d = {'a': 1, 'b': 2}  # dict
```

- Sets are unordered and unique
- Dicts store key-value pairs

1.2 Boolean and Binary Types

Examples:

```
flag = True      # bool  
b = bytes(5)     # binary
```

1.2 Type Conversion

Implicit: Python converts automatically

```
x = 5 + 3.2 # float
```

Explicit: Use `int()`, `float()`, `str()`, etc.

```
int('10') # 10
```

1.3 Comments in Python

Improve code readability

Used to describe logic or
disable code

1.3 Single-line and Multi-line Comments

Single-line:

```
# This is a comment
```

Multi-line:

```
'''
```

This is a

multi-line comment

```
'''
```

1.3 Use Cases of Comments



Explain complex code



Temporarily disable code lines



Clarify input/output behavior



Add TODOs or notes for developers

1.4 Variables in Python

- Containers for storing values
- Dynamically typed

Example:

```
name = 'Alice'
```

```
age = 25
```

1.4 Variable Naming Rules

- Start with a letter or underscore (_)
- Can contain letters, digits, underscores
- Cannot be a keyword (e.g., 'if', 'for')
- Case-sensitive

1.4 Constants in Python

- Convention: use uppercase for constants
- Python doesn't enforce immutability

Example:

PI = 3.14159

1.4 Arithmetic Operators

`+` Addition

`-` Subtraction

`*` Multiplication

`/` Division

`//` Floor Division

`%` Modulus

`**` Exponentiation

1.4 Comparison & Logical Operators

Comparison: `==`, `!=`, `<`, `>`, `<=`, `>=`

Logical: `and`, `or`, `not`

Example:

`x = 10`

`y = 5`

`print(x > y and x != 0)`

1.4 Assignment Operators

- Basic: `=`
- Compound: `+=`, `-=`, `*=`, `/=`, etc.

Example:

```
x = 10
```

```
x += 5 # x = x + 5
```

1.4 Performing Calculations

Example:

```
price = 50
```

```
qty = 3
```

```
total = price * qty
```

```
print(total)
```

1.5 Reading Input from Keyboard

Use `input()` function:

```
name = input('Enter your name: ')\nprint(name)
```


1.5 Converting Input Types

Inputs are strings by default

Convert using `int()`, `float()`, etc.

Example:

```
age = int(input('Enter age: '))
```

1.5 Example: Sum of Two Numbers

```
a = int(input('Enter first number: '))  
b = int(input('Enter second number: '))  
print('Sum:', a + b)
```

1.6 Using print() Function

- Displays output on screen

Example:

```
print('Hello')
```

```
print('Name:', name)
```

1.6 Using Comma in print()

Comma separates multiple values

```
print('Age is', age)
```

1.6 Old Formatting Style

Using `%` operator:

```
name = 'Alice'  
print('Hello %s' % name)
```

1.6 Using format() Method

More flexible formatting

```
print('Hello {}, you are {} years old'.format('Alice', 25))
```

1.6 Formatted Strings (f-strings)

Introduced in Python 3.6

Embed expressions directly

```
name = 'Alice'
```

```
age = 25
```

```
print(f'{name} is {age} years old')
```