Python Programming - Beginner Level Notes

Prepared for: Bikkad IT Institute Students
Topic: Core Python Fundamentals
Date: February 2026

---

1. Python Outputs

What is Output?

Output means displaying results or messages on the screen. In Python, we use the `print()` function to show output.

Basic Syntax

```
print("Hello, World!")
```

Important Points

- `print()` displays text, numbers, or variables on the console
- Multiple items can be printed using commas
- By default, `print()` adds a new line after each output
- Use `sep` parameter to change separator between items
- Use `end` parameter to change what comes after the output

Code Examples

Example 1: Basic Print
```
print("Welcome to Python")
print(100)
print(3.14)
```

Output:
```
Welcome to Python
100
3.14
```

Example 2: Multiple Items
```
print("Name:", "Rahul", "Age:", 25)
print(10, 20, 30, 40)
```

Output:
```
Name: Rahul Age: 25
10 20 30 40
```

Example 3: Using sep and end

```
print("Apple", "Banana", "Mango", sep=", ")
print("Hello", end=" ")
print("World")
```

Output:

```
Apple, Banana, Mango
Hello World
```

---

2. Python Data Types

What are Data Types?

Data types define what kind of value a variable can store. Python has several built-in data types.

Main Data Types

Data Type	Description	Example
int	Whole numbers	10, -5, 0
float	Decimal numbers	3.14, -2.5, 0.0
str	Text/String	"Hello", 'Python'
bool	True or False	True, False
list	Ordered collection	[1, 2, 3]
tuple	Immutable collection	(1, 2, 3)
dict	Key-value pairs	{"name": "Raj"}
set	Unique items	{1, 2, 3}

Table 1: Common Python Data Types

Important Points

- Python automatically detects data type (Dynamic Typing)
- Use `type()` function to check data type
- Each data type has specific operations
- Strings use quotes (single or double)
- Lists use square brackets [ ]
- Tuples use parentheses ( )
- Dictionaries use curly braces { }

Code Examples

Example 1: Checking Data Types

Integer

```
x = 10
print(type(x)) # <class 'int'>
```

Float

```
y = 3.14
print(type(y)) # <class 'float'>
```

String

```
name = "Python"
print(type(name)) # <class 'str'>
```

Boolean

```
is_active = True
print(type(is_active)) # <class 'bool'>
```

Example 2: Collection Data Types

List

```
fruits = ["Apple", "Banana", "Mango"]
print(type(fruits)) # <class 'list'>
```

Tuple

```
coordinates = (10, 20)
print(type(coordinates)) # <class 'tuple'>
```

Dictionary

```
student = {"name": "Amit", "age": 20}
print(type(student)) # <class 'dict'>
```

Set

```
numbers = {1, 2, 3, 4, 5}
print(type(numbers)) # <class 'set'>
```

---

3. Variables

What is a Variable?

A variable is a container that stores data values. It is like a label that points to a value in memory.

Variable Rules

- Must start with a letter (a-z, A-Z) or underscore (_)
- Can contain letters, numbers, and underscores
- Cannot start with a number
- Case-sensitive (age and Age are different)
- Cannot use Python keywords as variable names
- Use meaningful names for clarity

Valid Variable Names

```
name = "Rahul"
_age = 25
student1 = "Priya"
total_marks = 450
```

Invalid Variable Names

```
1student = "Error" # Cannot start with number
my-name = "Error" # Cannot use hyphen
class = "Error" # Cannot use keyword
```

Important Points

- No need to declare variable type in Python
- Variables can change type during program execution
- Multiple variables can be assigned in one line
- Variable names should be descriptive
- Use snake_case for variable names (recommended)

Code Examples

Example 1: Basic Variable Assignment

```
name = "Sneha"
age = 22
height = 5.4
is_student = True

print(name)
print(age)
print(height)
print(is_student)
```

Example 2: Multiple Assignment

Assign same value to multiple variables

```
x = y = z = 100
print(x, y, z) # 100 100 100
```

Assign different values

```
a, b, c = 10, 20, 30
print(a, b, c) # 10 20 30
```

Example 3: Variable Reassignment

```
marks = 85
print(marks) # 85

marks = 90 # Changed value
print(marks) # 90

marks = "Excellent" # Changed type
print(marks) # Excellent
```

---

4. Comments in Python

What are Comments?

Comments are notes in code that Python ignores. They help explain code to other programmers or remind yourself later.

Types of Comments

1. Single-Line Comments (use `#`)

This is a single-line comment

```
print("Hello") # This comment is after code
```

2. Multi-Line Comments (use triple quotes)

```
"""
This is a multi-line comment.
It can span multiple lines.
Used for longer explanations.
"""
print("Python")
```

Important Points

- Comments are not executed by Python
- Use comments to explain complex logic
- Comments improve code readability
- Single-line comments start with #

- Multi-line comments use triple quotes (""" or ''')
- Good comments explain WHY, not WHAT

Code Examples

Example 1: Single-Line Comments

Calculate total marks

```
maths = 85
science = 90
total = maths + science # Add both subjects
print(total)
```

Example 2: Multi-Line Comments

```
"""
Program: Student Grade Calculator
Purpose: Calculate average marks
Author: Bikkad IT Institute
Date: February 2026
"""

marks1 = 75
marks2 = 80
marks3 = 85
average = (marks1 + marks2 + marks3) / 3
print("Average:", average)
```

Example 3: Commenting Out Code

This code is temporarily disabled

print("This will not run")

```
print("This will run")
```

---

5. Keywords & Identifiers

What are Keywords?

Keywords are reserved words in Python that have special meaning. You cannot use them as variable names.

Python Keywords (35 in total)

and	as	assert	break	class
continue	def	del	elif	else
except	False	finally	for	from
global	if	import	in	is
lambda	None	nonlocal	not	or
pass	raise	return	True	try
while	with	yield	async	await

Table 2: Python Keywords

What are Identifiers?

Identifiers are names given to variables, functions, classes, etc. They are user-defined names.

Identifier Rules

- Can contain letters (a-z, A-Z), digits (0-9), and underscore (_)
- Must start with a letter or underscore
- Cannot start with a digit
- Case-sensitive
- Cannot be a keyword
- No special characters allowed (!@#$%^&*)

Code Examples

Example 1: Checking Keywords
import keyword

Display all keywords

print(keyword.kwlist)

Check if a word is keyword

print(keyword.iskeyword("if")) # True
print(keyword.iskeyword("name")) # False

Example 2: Valid Identifiers
student_name = "Raj"
_age = 25
marks1 = 85
totalMarks = 450

Example 3: Invalid Identifiers

These will cause errors:

1student = "Error" # Starts with digit

my-name = "Error" # Contains hyphen

for = 10 # Keyword used

@price = 100 # Special character

---

6. User Input

What is User Input?

User input allows the program to receive data from the user during execution. We use the `input()` function.

Basic Syntax

```
variable_name = input("Enter your message: ")
```

Important Points

- `input()` always returns a string
- Use type conversion to get numbers
- Prompt message is optional but recommended
- User must press Enter to submit input
- Input is stored in a variable for later use

Code Examples

Example 1: Basic Input

```
name = input("Enter your name: ")
print("Hello,", name)
```

Example 2: Taking Number Input

Wrong way - input is string

```
age = input("Enter your age: ")
```

age + 5 will cause error

Right way - convert to integer

```
age = int(input("Enter your age: "))
next_year_age = age + 1
print("Next year you will be:", next_year_age)
```

Example 3: Multiple Inputs

```
name = input("Enter your name: ")
city = input("Enter your city: ")
age = int(input("Enter your age: "))

print("Name:", name)
print("City:", city)
print("Age:", age)
```

Example 4: Simple Calculator

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

sum_result = num1 + num2
print("Sum:", sum_result)
```

---

7. Type Conversion

What is Type Conversion?

Type conversion means changing data from one type to another. Also called Type Casting.

Types of Conversion

1. Implicit Conversion (Automatic)
Python automatically converts lower data type to higher data type.

2. Explicit Conversion (Manual)
Programmer manually converts using built-in functions.

Conversion Functions

Function	Purpose
int()	Convert to integer
float()	Convert to float
str()	Convert to string
bool()	Convert to boolean
list()	Convert to list
tuple()	Convert to tuple
set()	Convert to set

Table 3: Type Conversion Functions

Important Points

- Use int() to convert string/float to integer
- Use float() to convert string/int to decimal
- Use str() to convert number to string
- Not all conversions are possible
- Invalid conversions cause errors

Code Examples

Example 1: Implicit Conversion

```
a = 10 # int
b = 3.5 # float
c = a + b # Python converts int to float

print(c) # 13.5
print(type(c)) # <class 'float'>
```

Example 2: Explicit Conversion - Numbers

String to Integer

```
x = "100"
y = int(x)
print(y + 50) # 150
```

Integer to Float

```
a = 25
b = float(a)
print(b) # 25.0
```

Float to Integer (removes decimal)

```
c = 3.99
d = int(c)
print(d) # 3
```

Example 3: Explicit Conversion - String

Number to String

```
age = 25
age_str = str(age)
print("Age: " + age_str)
```

Concatenation

```
marks = 85
print("Marks: " + str(marks))
```

Example 4: Boolean Conversion

Non-zero numbers are True

```
print(bool(10)) # True
print(bool(0)) # False
```

Non-empty strings are True

```
print(bool("Hi")) # True
print(bool("")) # False
```

Example 5: Practical Use

Taking numeric input from user

```
marks1 = int(input("Enter Math marks: "))
marks2 = int(input("Enter Science marks: "))

total = marks1 + marks2
average = total / 2

print("Total:", total)
print("Average:", average)
```

---

8. Literals

What are Literals?

Literals are raw data values assigned to variables. They are fixed values that don't change.

Types of Literals

1. Numeric Literals

- Integer Literals: 10, -5, 0
- Float Literals: 3.14, -2.5, 0.0
- Complex Literals: 3+4j

2. String Literals

- Single quotes: 'Hello'
- Double quotes: "Python"
- Triple quotes: '''Multi-line'''

3. Boolean Literals

- True
- False

4. Special Literal

- None (represents absence of value)

Code Examples

Example 1: Numeric Literals

Integer literals

```
decimal_num = 100
binary_num = 0b1010 # Binary (10 in decimal)
octal_num = 0o12 # Octal (10 in decimal)
hexa_num = 0xA # Hexadecimal (10 in decimal)

print(decimal_num)
print(binary_num)
print(octal_num)
print(hexa_num)
```

Float literals

```
pi = 3.14
scientific = 1.5e2 # 1.5 x 10^2 = 150.0
print(pi)
print(scientific)
```

Example 2: String Literals

Single and double quotes

```
name1 = 'Python'
name2 = "Programming"
```

Escape sequences

```
text1 = "Hello\nWorld" # New line
text2 = "Name:\tRahul" # Tab
text3 = "She said "Hi"" # Quote inside

print(text1)
print(text2)
print(text3)
```

Multi-line string

```
message = """
This is a multi-line
string literal in Python.
Very useful for long text.
"""
print(message)
```

Example 3: Boolean and Special Literals

Boolean literals

```
is_active = True
is_deleted = False

print(is_active)
print(is_deleted)
```

None literal

```
result = None
print(result)
print(type(result)) # <class 'NoneType'>
```

Example 4: Collection Literals

List literal

```
numbers = [1, 2, 3, 4, 5]
```

Tuple literal

```
coordinates = (10, 20)
```

Dictionary literal

```
student = {"name": "Amit", "age": 20, "marks": 85}
```

Set literal

```
unique_nums = {1, 2, 3, 4, 5}
```

```
print(numbers)
print(coordinates)
print(student)
print(unique_nums)
```

Practice Exercises

Exercise 1: Variables and Output

Write a Python program that stores your name, age, and city in variables and prints them.

Exercise 2: User Input

Create a program that asks the user for their name and favorite subject, then prints a personalized message.

Exercise 3: Simple Calculator

Write a program that takes two numbers from the user and displays their sum, difference, product, and quotient.

Exercise 4: Type Conversion

Take a string input from the user, convert it to integer, add 10 to it, and display the result.

Exercise 5: Temperature Converter

Create a program that converts temperature from Celsius to Fahrenheit.
Formula: F = (C × 9/5) + 32

Quick Reference Summary

Concept	Key Point
Output	print() function
Data Types	int, float, str, bool, list, tuple, dict, set
Variables	Containers for storing data
Comments	# for single-line, """ """ for multi-line
Keywords	35 reserved words in Python
Input	input() function (returns string)
Type Conversion	int(), float(), str(), bool()
Literals	Fixed values: 10, 3.14, "text", True

Table 4: Python Fundamentals Quick Reference

Important Tips for Students

1. Always practice code by typing yourself
2. Use meaningful variable names
3. Comment your code for clarity
4. Test code with different inputs
5. Understand error messages
6. Use `type()` to check data types
7. Remember: `input()` returns string
8. Convert input to correct type before calculations
9. Start with simple programs and gradually increase complexity
10. Practice regularly to build confidence

End of Notes

Prepared by: Bikkad IT Institute
For: Python Beginner Level Students
Date: February 2026