Python Programming - Beginner Level Notes

Prepared for: Bikkad IT Institute Students
Topic: Operators, If-Else, Modules & Loops
Date: February 2026

---

1. Operators in Python

What are Operators?

Operators are special symbols that perform operations on variables and values.

A. Arithmetic Operators

Perform mathematical operations.

Operator	Description	Example
+	Addition	5 + 3 = 8
-	Subtraction	5 - 3 = 2
*	Multiplication	5 * 3 = 15
/	Division (float)	5 / 2 = 2.5
%	Modulus (remainder)	5 % 2 = 1
//	Floor Division	5 // 2 = 2
**	Exponentiation	5 ** 2 = 25

Table 1: Arithmetic Operators

Code Example:

```
a = 10
b = 3

print("Addition:", a + b) # 13
print("Subtraction:", a - b) # 7
print("Multiplication:", a * b) # 30
print("Division:", a / b) # 3.333...
print("Modulus:", a % b) # 1
print("Floor Division:", a // b) # 3
print("Exponent:", a ** b) # 1000
```

B. Relational (Comparison) Operators

Compare two values and return True or False.

Operator	Description	Example
==	Equal to	5 == 5 → True
!=	Not equal to	5 != 3 → True
>	Greater than	5 > 3 → True
<	Less than	5 < 3 → False
>=	Greater or equal	5 >= 5 → True
<=	Less or equal	5 <= 3 → False

Table 2: Relational Operators

Code Example:

```
x = 10
y = 20

print(x == y) # False
print(x != y) # True
print(x > y) # False
print(x < y) # True
print(x >= 10) # True
print(y <= 15) # False
```

C. Logical Operators

Combine multiple conditions.

Operator	Description
and	True if both conditions are True
or	True if at least one is True
not	Reverses the result

Table 3: Logical Operators

Code Example:

```
a = 10
b = 20
c = 30
```

AND operator

print(a < b and b < c) # True

OR operator

print(a > b or b < c) # True

NOT operator

print(not(a > b)) # True

D. Bitwise Operators

Work on binary representation of numbers.

Operator	Description	Example
&	AND	5 & 3 = 1
|	OR	5 | 3 = 7
^{}	XOR	5 ^{} 3 = 6
~{}	NOT	~{}5 = -6
<<	Left Shift	5 << 1 = 10
>>	Right Shift	5 >> 1 = 2

Table 4: Bitwise Operators

Code Example:

```
a = 5 # Binary: 0101
b = 3 # Binary: 0011

print("AND:", a & b) # 1
print("OR:", a | b) # 7
print("XOR:", a ^ b) # 6
print("NOT:", ~a) # -6
print("Left Shift:", a << 1) # 10
print("Right Shift:", a >> 1) # 2
```

E. Assignment Operators

Assign values to variables.

Operator	Example	Equivalent
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Table 5: Assignment Operators

Code Example:

```
x = 10
print(x) # 10

x += 5 # x = x + 5
print(x) # 15

x *= 2 # x = x * 2
print(x) # 30

x //= 4 # x = x // 4
print(x) # 7
```

F. Membership Operators

Test if a value exists in a sequence.

Operator	Description
in	Returns True if value exists
not in	Returns True if value does NOT exist

Table 6: Membership Operators

Code Example:

```
fruits = ["Apple", "Banana", "Mango"]

print("Banana" in fruits) # True
print("Grapes" in fruits) # False
print("Grapes" not in fruits) # True
```

String membership

text = "Python Programming"
print("Python" in text) # True

---

2. If-Else Statements

What is If-Else?

Decision-making statement that executes code based on conditions.

Syntax

if condition:
# Code if True
else:
# Code if False

Important Points

- Indentation is mandatory (4 spaces or 1 tab)
- Condition must return True or False
- `elif` is short for "else if"
- `else` is optional

Code Examples

Example 1: Simple If-Else
age = 20

if age >= 18:
print("You are an adult")
else:
print("You are a minor")

Example 2: If-Elif-Else
marks = int(input("Enter marks: "))

if marks >= 90:
print("Grade: A+")
elif marks >= 80:
print("Grade: A")
elif marks >= 70:
print("Grade: B")
elif marks >= 60:
print("Grade: C")
elif marks >= 40:
print("Grade: D")
else:
print("Grade: F (Fail)")

Example 3: Nested If

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

if age >= 18:
marks = int(input("Enter your marks: "))
if marks >= 60:
print("Eligible for admission")
else:
print("Not eligible - Low marks")
else:
print("Not eligible - Age below 18")
```

Example 4: Ternary Operator

```
age = 20
```

One-line if-else

```
status = "Adult" if age >= 18 else "Minor"
print(status)
```

---

3. Modules in Python

What is a Module?

A module is a file containing Python code that can be imported and reused.

How to Import

```
import module_name
from module_name import function_name
import module_name as alias
```

A. Math Module

Provides mathematical functions.

Common Functions:

Function	Description
sqrt(x)	Square root
pow(x, y)	x raised to power y
ceil(x)	Round up
floor(x)	Round down
factorial(x)	Factorial
pi	Pi constant (3.14159...)

Table 7: Math Module Functions

Code Example:
import math

Square root

print(math.sqrt(25)) # 5.0

Power

print(math.pow(2, 3)) # 8.0

Rounding

print(math.ceil(3.2)) # 4
print(math.floor(3.8)) # 3

Factorial

print(math.factorial(5)) # 120

Constants

print(math.pi) # 3.141592653589793

Circle area

radius = 5
area = math.pi * radius ** 2
print("Area:", area)

B. Keyword Module

Work with Python keywords.

Code Example:
import keyword

Display all keywords

print(keyword.kwlist)

Check if keyword

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

Count total keywords

print("Total keywords:", len(keyword.kwlist))

C. Random Module

Generate random numbers and selections.

Common Functions:

Function	Description
random()	Random float between 0 and 1
randint(a, b)	Random integer between a and b
choice(seq)	Random element from sequence
shuffle(seq)	Shuffle sequence

Table 8: Random Module Functions

Code Example:
import random

Random float

print(random.random())

Random integer

print(random.randint(1, 10))

Random choice

fruits = ["Apple", "Banana", "Mango"]
print(random.choice(fruits))

Shuffle list

numbers = [1, 2, 3, 4, 5]
random.shuffle(numbers)
print(numbers)

Dice rolling

dice = random.randint(1, 6)
print("Dice:", dice)

Generate OTP

otp = random.randint(1000, 9999)
print("OTP:", otp)

D. Datetime Module

Work with dates and times.

Common Functions:

Function	Description
datetime.now()	Current date and time
date.today()	Current date
strftime()	Format datetime as string

Table 9: Datetime Module Functions

Code Example:
import datetime

Current date and time

now = datetime.datetime.now()
print(now)

Current date only

today = datetime.date.today()
print(today)

Extract components

print("Year:", now.year)
print("Month:", now.month)
print("Day:", now.day)
print("Hour:", now.hour)

Format date

print(now.strftime("%d/%m/%Y")) # 01/02/2026
print(now.strftime("%B %d, %Y")) # February 01, 2026

Calculate age

birth_year = 2000
current_year = datetime.date.today().year
age = current_year - birth_year
print("Age:", age)

---

4. Loops in Python

What are Loops?

Execute a block of code repeatedly.

A. While Loop

Executes as long as condition is True.

Syntax:
while condition:
# Code to execute
# Update condition

Important Points:

- Condition checked before each iteration
- Must update condition to avoid infinite loop
- Use `break` to exit early
- Use `continue` to skip iteration

Code Examples:

Example 1: Basic While Loop
count = 1

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

Output: 1 2 3 4 5

Example 2: Print Table

```
num = 5
i = 1

while i <= 10:
print(f"{num} x {i} = {num * i}")
i += 1
```

Example 3: Break Statement

```
count = 1

while True:
print(count)
count += 1
if count > 5:
break
```

Example 4: Continue Statement

```
num = 0

while num < 10:
num += 1
if num % 2 == 0:
continue # Skip even numbers
print(num)
```

Output: 1 3 5 7 9

B. While Loop with Else

`else` block executes when loop completes normally (not broken by `break`).

Code Example:

```
count = 1

while count <= 5:
print(count)
count += 1
else:
print("Loop completed successfully")
```

Example with Break (Else Not Executed):

```
count = 1

while count <= 10:
print(count)
if count == 5:
break
count += 1
```

else:
print("This will NOT print")

C. For Loop

Iterates over a sequence (list, tuple, string, range).

Syntax:
for variable in sequence:
# Code to execute

Important Points:

- Used when number of iterations is known
- Works with any iterable
- range() function commonly used
- More concise than while loop

Code Examples:

Example 1: Loop Through List
fruits = ["Apple", "Banana", "Mango"]

for fruit in fruits:
print(fruit)

Example 2: Loop Through String
name = "Python"

for char in name:
print(char)

Output: P y t h o n

Example 3: Using range()

Print 1 to 5

for i in range(1, 6):
print(i)

Print even numbers

for i in range(2, 11, 2):
print(i) # 2 4 6 8 10

Example 4: Print Table
num = 7

for i in range(1, 11):
print(f"{num} x {i} = {num * i}")

Example 5: Sum of List
numbers = [10, 20, 30, 40, 50]
total = 0

for num in numbers:
total += num

print("Total:", total) # 150

Example 6: Break and Continue

Break

for i in range(1, 11):
if i == 6:
break
print(i) # 1 2 3 4 5

Continue

for i in range(1, 11):
if i % 2 == 0:
continue
print(i) # 1 3 5 7 9

D. Nested Loops

A loop inside another loop.

Important Points:

- Inner loop completes fully for each outer loop iteration
- Used for patterns and 2D data
- Can nest while inside for or vice versa

Code Examples:

Example 1: Basic Nested Loop
for i in range(1, 4):
for j in range(1, 4):
print(f"i={i}, j={j}")

Example 2: Star Pattern
for i in range(1, 6):
for j in range(i):
print("*", end="")
print()
Output:
*
**

Example 3: Number Pattern

```
for i in range(1, 6):
for j in range(1, i + 1):
print(j, end=" ")
print()
```

Output:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Example 4: Multiplication Table

```
for i in range(1, 4):
print(f"Table of {i}:")
for j in range(1, 6):
print(f"{i} x {j} = {i * j}")
print()
```

Example 5: 2D List (Matrix)

```
matrix = [
[1, 2, 3],
[4, 5, 6],
[7, 8, 9]
]

for row in matrix:
for element in row:
print(element, end=" ")
print()
```

Output:

```
1 2 3
4 5 6
7 8 9
```

Practice Exercises

Exercise 1: Operators

Write a program that takes two numbers and displays results of all arithmetic operations.

Exercise 2: Grade Calculator

Create a program using if-elif-else that assigns grades based on marks.

Exercise 3: Even Numbers

Use a while loop to print all even numbers from 1 to 20.

Exercise 4: Factorial

Write a program using for loop to calculate factorial of a number.

Exercise 5: Pattern Printing

Print the following pattern using nested loops:
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

Exercise 6: Random Number Game

Create a number guessing game using random module.

Exercise 7: Age Calculator

Use datetime module to calculate your age in years.

Exercise 8: Prime Number

Check if a number is prime using loops.

Quick Reference

Concept	Key Point
Arithmetic	+, -, *, /, %, //, **
Relational	==, !=, >, <, >=, <=
Logical	and, or, not
Bitwise	&, |, ^{}, ~{}, <<, >>
Assignment	=, +=, -=, *=, /=, %=, //=, **=
Membership	in, not in
If-Else	Decision making with conditions
While Loop	Repeat while condition is True
For Loop	Iterate over sequences
Nested Loops	Loop inside loop
Math Module	sqrt(), pow(), ceil(), floor(), pi
Random Module	randint(), choice(), shuffle()
Datetime Module	now(), today(), strftime()

Table 10: Python Quick Reference

End of Notes

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