

Day11_BitwiseOperators_Packages_Functions_Input_Mini_Programs

May 29, 2025

1 Day 11 – Python Practice: Bitwise Operators, Packages, Functions, Input & Mini Programs

Today I revised many core concepts and learned how everything fits together in Python. I also wrote simple programs to practice what I've learned so far.

1.1 What I Learned Today

1.1.1 1. String Manipulation

- How to change characters in a string using indexing and slicing.
- Example: Changing 'fine' to 'dine'.

1.1.2 2. Swapping Numbers

- I learned 3 different ways to swap two numbers:
 1. Using a temporary variable
 2. Without a temporary variable (arithmetic)
 3. Pythonic way using tuple unpacking

1.1.3 3. Bitwise Operators

- Bitwise operations work on binary bits.
- $\&$ \rightarrow AND, $|$ \rightarrow OR, \wedge \rightarrow XOR
- Used with integers like 35 and 40

1.1.4 4. Packages, Modules & Functions

- A **package** is a folder of **modules**
- A **module** is a file that can have **functions**
- Functions can be **built-in** or **user-defined**

Example:

```
“python from dmart.food import pasta # dmart = package, food = module, pasta = function
```

1.1.5 5. Built-in Modules: math

I imported the math module to use functions like sqrt() and pow()

1.1.6 6. Taking Input from User

I used the input() function to take values from users and perform calculations.

2 Code Practice

2.1 Convert 'fine' to 'dine'

```
[1]: word = "fine"
new_word = 'd' + word[1:]
print(new_word) # Output: dine
```

dine

Swap Two Numbers (3 Ways)

```
[2]: # 1. Using a third variable
a = 10
b = 5
temp = a
a = b
b = temp
print(a, b) # 5 10

# 2. Without using third variable
a, b = 10, 5
a = a + b
b = a - b
a = a - b
print(a, b) # 5 10

# 3. Pythonic way
a, b = 10, 5
a, b = b, a
print(a, b) # 5 10
```

5 10

5 10

5 10

3 Bitwise Operators

3.0.1 AND (&): 1 only if both bits are 1

Rule: $1 \& 1 = 1$, $1 \& 0 = 0$, $0 \& 1 = 0$, $0 \& 0 = 0$ `print("5 & 3 =", 5 & 3) # 0101 & 0011 = 0001 → 1`

3.0.2 OR (|): 1 if at least one bit is 1

Rule: $1 \mid 1 = 1, 1 \mid 0 = 1, 0 \mid 1 = 1, 0 \mid 0 = 0$ `print("5 | 3 =", 5 | 3) # 0101 | 0011 = 0111`
 $\rightarrow 7$

3.0.3 XOR (^): 1 if bits are different

Rule: $1 \wedge 1 = 0, 1 \wedge 0 = 1, 0 \wedge 1 = 1, 0 \wedge 0 = 0$ `print("5 ^ 3 =", 5 ^ 3) # 0101 ^ 0011 = 0110`
 $\rightarrow 6$

3.0.4 NOT (~): flips all bits (inverts 1 \rightarrow 0 and 0 \rightarrow 1), result is negative in Python

`print("~5 =", ~5) # ~00000101 = 11111010 \rightarrow -6`

3.0.5 Left Shift («): shifts bits left, like multiplying by 2

`print("5 « 1 =", 5 « 1) # 0101 \rightarrow 1010 = 10`

3.0.6 Right Shift (»): shifts bits right, like dividing by 2

`print("5 » 1 =", 5 » 1) # 0101 \rightarrow 0010 = 2`

```
[3]: a = 35  # 00100011
     b = 40  # 00101000

     print(a & b)  # 32
     print(a | b)  # 43
     print(a ^ b)  # 11
```

32

43

11

Importing a Built-in Module (math)

```
[4]: import math
     print(math.sqrt(25))  # 5.0
     print(math.pow(2, 4))  # 16.0
```

5.0

16.0

4 Or

```
[5]: import math as m
     print(m.sqrt(25))  # 5.0
     print(m.pow(2, 4))  # 16.0
```

5.0

16.0

5 Understanding input() and eval() in Python

5.1 1. input() Returns a String by Default

Even if the user types 10, it's stored as a string, not a number.

```
[6]: x = input("Enter a number: ")
```

Enter a number: 10

```
[8]: x = input("Enter 1st Number :")
y = input("Enter 2nd Number :")
z = x + y
print(z)
```

Enter 1st Number : 10

Enter 2nd Number : 20

1020

5.2 Because it concatenates strings ('10' + '20' → '1020')

6 2. Use int() to Convert to Integer

To perform actual arithmetic (addition, subtraction, etc.), convert input to an integer:

```
[9]: x1 = int(input("Enter the 1st number: "))
y1 = int(input("Enter the 2nd number: "))
z1 = x1 + y1
print("Sum is:", z1)
```

Enter the 1st number: 10

Enter the 2nd number: 20

Sum is: 30

7 3. Use eval() to Auto-Detect Number Types

```
[13]: x = eval(input("Enter 1st number: "))
y = eval(input("Enter 2nd number: "))
print("Ans is:", 2 - x + 5 - y )
```

Enter 1st number: 8

Enter 2nd number: 2

Ans is: -3