



Simple Operations



Instructor: Bulend Hoca



Table of Contents



- ▶ Arithmetic Operations
- ▶ Operations with `print()` Function
- ▶ Escape Sequences



Draw lines to match the operator to the answer:

//

/

addition

subtraction

exponentiation

division

modulus

floor division

%

-

+



Students, draw anywhere on this slide!



Arithmetic Operations

Operator	Description	Example
+	Addition operator	$100 + 45 = 145$
-	Subtraction operator	$500 - 65 = 435$
*	Multiplication operator	$25 * 4 = 100$
/	Float Division Operator	$10 / 2 = 5.0$
//	Integer Division Operator	$11 // 2 = 5$
**	Exponentiation Operator	$5 ** 3 = 125$
%	Remainder Operator	$10 \% 3 = 1$



Arithmetic Operations

- Interactive question :

```
1 print(11-7)
2 print(4 + 11.0)
3 print('11 - 7')
4 print('4' + 4)
5 |
```

What is the output?





Arithmetic Operations

- ▶ The output :

```
1 print(11-7)
2 print(4 + 11.0)
3 print('11 - 7')
4 print('4' + 4)
5 |
```

```
4
15.0
11 - 7
Traceback (most recent call last):
  File "code.py", line 5, in <module>
    print('4'+ 4)
TypeError: can only concatenate str (not "int") to str
```



Arithmetic Operations

- Interactive question :

```
1 num1, num2 = 81, 55
2 num3 = num1 - num2
3 print(num3)
4
5
```


What is the output?





▶ Arithmetic Operations

- ▶ Interactive question :




```
1 num1, num2 = 81, 55
2 num3 = num1 - num2
3 print(num3)
4
5
```



Arithmetic Operations

- The output :



```
1 num1, num2 = 81, 55
2 num3 = num1 - num2
3 print(num3)
4
5
```

Output

```
26
```



▶ Arithmetic Operations

▶ **Task:** Let's calculate the **area** of a **circle**:

▷ $r = 5$ # 'r' is the radius.

▷ $\text{area} = ?$

▷ Area of a circle can be calculated by using the formula:

▷ $\text{area} = \pi \times r^2$



▶ Arithmetic Operations

- ▶ Let's calculate the **area** of a **circle**:

```
pi = 3.14  
r = 5  
area = pi * r**2  
  
print(area)
```

78.5



Arithmetic Operations

- Interactive question :

```
1 print(11 % 2)  # remainder of this division is 1
2               # it means 11 is an odd number
3 print((4 * 5) / 2)  # parentheses are used as in normal math operations
4
```

What is the output?





Arithmetic Operations

- ▶ The output :

```
1 print(11 % 2) # remainder of this division is 1
2             # it means 11 is an odd number
3 print((4 * 5) / 2) # parentheses are used as in normal math operations
4
```

1

10.0



Arithmetic Operations

- Interactive question :

```
1 print(2 ** 3) # 2 to the power of 3
2 print(3 ** 2) # square of 3
3 a = 2
4 b = 8
5 print((a * b) ** 0.5) # square root
6 |
7
```

What is the output?





Arithmetic Operations

- ▶ The output :

```
1 print(2 ** 3) # 2 to the power of 3
2 print(3 ** 2) # square of 3
3 a = 2
4 b = 8
5 print((a * b) ** 0.5) # square root
6
7
```

```
8
9
4.0
```




Arithmetic Operations

💡 Tips:

- `Variable math operator = number` gives the same result as `Variable = Variable math operator number`.
- `Variable += number` gives the same result as `Variable = Variable + number`.

`x += 3` \Leftrightarrow `x = x + 3`

`x *= 3` \Leftrightarrow `x = x * 3`

`x **= 3` \Leftrightarrow `x = x ** 3`



Arithmetic Operations

💡 Tips:

- `Variable math operator = number` gives the same result as `Variable = Variable math operator number`.
- `Variable += number` gives the same result as `Variable = Variable + number`.

- `--` decrements the variable in place,
- `++` increment the variable in place,
- `*=` multiply the variable in place,
- `/=` divide the variable in place,
- `//=` floor divide the variable in place,
- `%=` returns the modulus of the variable in place,
- `**=` raise to power in place.

`x += 3` \Leftrightarrow `x = x + 3`

`x *= 3` \Leftrightarrow `x = x * 3`

`x **= 3` \Leftrightarrow `x = x ** 3`



Arithmetic Operations



1. parentheses : `()`
2. power : `**`
3. unary minus : `-`
4. multiplication and division : `*`, `/`
5. addition and subtraction : `+`, `-`



Arithmetic Operations

- ▶ Interactive question :



```
a = (1 + 3 ) ** (2 ** (1 * 2 / 2) / 2)
print(a)
```

What is the output?



Students, write your response!

REINVENT YOURSELF



▶ Arithmetic Operations

- ▶ The output :

```
a = (1 + 3 ) ** (2 ** (1 * 2 / 2) / 2)
print(a)
```

4.0



Arithmetic Operations

► **Task** : Let's calculate the **hypotenuse** of a **triangle**:

▷ $a = 3$

▷ $b = 4$

▷ $c = ?$

Hypotenuse formula : $c = \sqrt{a^2 + b^2}$.



► Arithmetic Operations

- Let's calculate the **hypotenuse** of a **triangle**:

```
a = 3
b = 4
c = (a ** 2 + b ** 2) ** 0.5

print(c)
```

5.0



Operations with `print()`

I'm the king of
the functions.



Operations with `print()` Function

- ▶ Printing the variables

```
number = 2021  
text = "we have reached"  
print(text, number)
```



Operations with `print()` Function

- ▶ The output :

```
number = 2020  
text = "we have reached"  
print(text, number)
```

```
we have reached 2020
```

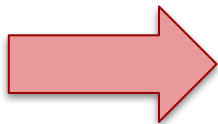


Operations with `print()` Function

- ▶ Let's take a look at the **inside** of `print()` function :

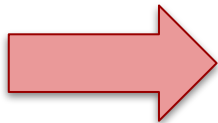
```
print(value, ..., sep=' ', end='\n')
```

Separation
parameter \Rightarrow `sep`



Default value \Rightarrow space

End of the line
parameter \Rightarrow `end`



Default value \Rightarrow newline



Operations with `print()` Function

```
text1 = "I bought"  
text2 = "kg. of apple this morning"  
amount = 6  
text3 = text1 + " " + str(amount) + " " + text2  
print(text1, amount, text2)  
print("I bought", 6, "kg. of apple this morning")  
print("I bought " + "6 " + "kg. of apple this morning")  
print(text3)
```

What is the output? Try to
guess in your mind...





Operations with `print()` Function

```
text1 = "I bought"  
text2 = "kg. of apple this morning"  
amount = 6  
text3 = text1 + " " + str(amount) + " " + text2  
print(text1, amount, text2)  
  
print("I bought", 6, "kg. of apple this morning")  
print("I bought " + "6 " + "kg. of apple this morning")  
print(text3)
```

```
I bought 6 kg. of apple this morning  
I bought 6 kg. of apple this morning  
I bought 6 kg. of apple this morning  
I bought 6 kg. of apple this morning
```



Operations with `print()` Function

```
text1 = "I bought"  
text2 = "kg. of apple this morning"  
amount = 6  
print(text1, amount, text2, sep="*")  
print("I bought", 6, "kg. of apple this morning", sep="")  
print("I bought " + "6 " + text2, end=" ")  
print("and came to home")
```

What is the output? Try to
guess in your mind...





Operations with `print()` Function

```
text1 = "I bought"  
text2 = "kg. of apple this morning"  
amount = 6  
print(text1, amount, text2, sep="*")  
print("I bought", 6, "kg. of apple this morning", sep="")  
print("I bought " + "6 " + text2, end=" ")  
print("and came to home")
```

```
I bought*6*kg. of apple this morning  
I bought6kg. of apple this morning  
I bought 6 kg. of apple this morning and came to home
```



Operations with `print()` Function

```
1 x = 5
2 print ('value of x      : ', x)
3
4 x += 2
5 print ("2 more of x      : ", x, "\n") # using string expression '\n',
6                                         # we produce extra line.
7                                         # So that we had empty line.
8 y = 10
9 print ('value of y      : ', y)
10
11 y -= 2
12 print ("2 minus y      : ", y, "\n")
13
14 z = 6
15 print ('value of z      : ', z)
16
17 z *= 2
18 print ("2 times z      : ", z, "\n")
19
```

What is the output? Use
your **Playground**...



Operations with `print()` Function

```
1 x = 5
2 print ('value of x      : ', x)
3
4 x += 2
5 print ("2 more of x      : ", x, "\n") # using string expression '\n',
6                                         # we produce extra line.
7                                         # So that we had empty line.
8 y = 10
9 print ('value of y      : ', y)
10
11 y -= 2
12 print ("2 minus y      : ", y, "\n")
13
14 z = 6
15 print ('value of z      : ', z)
16
17 z *= 2
18 print ("2 times z      : ", z, "\n")
```

1	value of x	:	5
2	2 more of x	:	7
3			
4	value of y	:	10
5	2 minus y	:	8
6			
7	value of z	:	6
8	2 times z	:	12
9			



Escape Sequences

`\n`

`\t`

`\b`



Escape Sequences (review)

Python ignores any character which comes immediately after `\`.

- `\n` : means new line,
- `\t` : means `tab` mark,
- `\b` : means backspace. It moves the cursor one character to the left.



Escape Sequences (review)

\'

Single quote

\"

Double quote

\\

backslash

\n

New line

\r

Carriage Return

\t

Horizontal tab

\b

Backspace

\N{name}

Unicode Character Database named Lookup

\uxxxxxxxx

Unicode Character with 16-bit hex value XXXX

\Uxxxxxxx

Unicode Character with 32-bit hex value XXXXXXXX

\ooo

Character with octal value 000

\xhh

Character with hex value HH

```
1 x = "\N{dog}"
2 print(x)
```

✓ 0.5s





Escape Sequences

- **Let's** take a closer look at the escape sequences through the examples.

```
print('C:\\north pole\\noise_penguins.txt')  
print('-----')  
print('first', 'second', 'third', sep='\\t')
```

What is the output? Try to guess in your mind...





Escape Sequences

- **Let's** take a closer look at the escape sequences through the examples.

```
print('C:\\north pole\\noise_penguins.txt')  
print('-----')  
print('first', 'second', 'third', sep='\\t')
```

```
C:\\north pole  
oise_penguins.txt  
-----  
first      second    third
```



Escape Sequences, Quiz

- ▶ **Let's** take a closer look at the escape sequences through the examples.

```
print('we are', '\boosting', 'our', '\brotherhood')  
print('it\'s essential to learn Python\'s libraries in IT World')
```

What is the output? Try to
guess in your mind...





Escape Sequences, Quiz

- **Let's** take a closer look at the escape sequences through the examples.

```
print('we are', '\boosting', 'our', '\brotherhood')  
print('it\'s essential to learn Python\'s libraries in IT World')
```

```
we areoosting ourrotherhood  
it's essential to learn Python's libraries in IT World
```


Escape Sequences, Quiz



► Task

- First, Login to your LMS,
- Then, click [here](#) to complete and submit the task.





Boolean Operations



Table of Contents



- ▶ Boolean Logic Expressions
- ▶ Order of Priority
- ▶ Truth Values of Logic Statements



not

Boolean Logic Expressions

and

or

Did you fully understand the **Boolean Logic**?



Students, drag the icon!

Pear Deck Interactive Slide
Do not remove this bar



Boolean Logic Expressions

- There are three built-in operators in Python :

and

It evaluates all expressions and returns the **last** expression if **all** expressions are evaluated **True**. Otherwise, it returns the **first** value that evaluated **False**.

or

It evaluates the expressions left to right and returns the first value that evaluated **True** or the last value (if none is **True**).

not

It evaluates the expression that follows it as the opposite of the truth. eg. **not True** means **False**



Boolean Logic Expressions

- ▶ Table of Logic Expressions in Python :

Value1	Logic	Value2	Returns
True	and	True	True
True	and	False	False
False	and	False	False
False	and	True	False
True	or	True	True
True	or	False	True
False	or	False	False
False	or	True	True

It's better to
keep this table
in mind.



Order of Priority



Order of Priority

- ▶ Here are the operators in order of their priorities :

1. not
2. and
3. or



Order of Priority

- ▶ It is important to remember that, logical operators have a different priority and it has an effect on the order of evaluation.
- ▶ Here are the operators in order of their priorities :

1. not

2. and

3. or

```
bool_var = False and not True  
print(bool_var)
```



Order of Priority

- ▶ It is important to remember that, logical operators have a different priority and it has an effect on the order of evaluation.
- ▶ Here are the operators in order of their p

1. not

2. and

3. or

```
bool_var = False and not True  
print(bool_var)
```

Firstly evaluated.
The result = False



Order of Priority

- It is important to remember that, logical operators have a different priority and it has an effect on the order of evaluation.
- Here are the operators and their priority:

1. not
2. and
3. or

```
bool_var = False and not True  
print(bool_var)
```

Secondly evaluated.
False and False =
False

Firstly evaluated.
The result = False



Order of Priority

- It is important to remember that, logical operators have a different priority and it has an effect on the order of evaluation.
- Here are the operators and their priority:

1. not
2. and
3. or

```
bool_var = False and not True  
print(bool_var)
```

Secondly evaluated.
False and False =
False

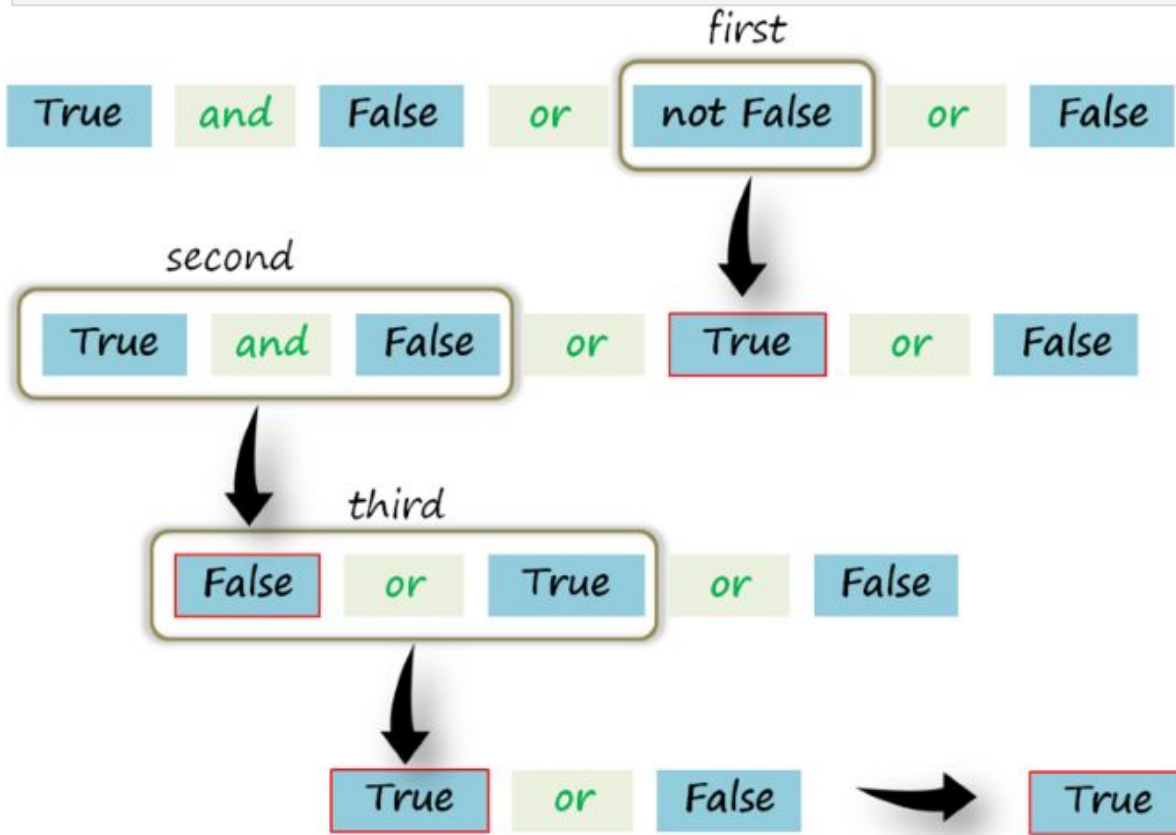
Firstly evaluated.
The result = False

False



Order of Priority (review)

True and False or not False or False = ?



Order of Priority



Tips:

- Note that `and` and `or` return one of its operands, not necessarily a `bool` type. But `not` always returns `bool` type.

```
print(1 and 0)
print(not 0)
```

What is the output? Try to guess in your mind...



Order of Priority



💡 Tips:

- Note that `and` and `or` return one of its operands, not necessarily a `bool` type. But `not` always returns `bool` type.

```
print(1 and 0)
print(not 0)
```

0

True



Truth Values of Logic Statements

The following values are considered False, in that they evaluate to **False** when applied to a boolean operator:

- **None**.
- Zero of any numeric type: `0, 0.0, 0j`
- Empty sequences and collections: `'', [], {}`.
- Other than above values, any remaining value is evaluated as **True**.

```
print(3 and "I am doing good!")  
print(3 or "I am doing good!")
```

I am doing good!

3



Identity operators

We use identity operators to compare the memory location of two objects.

Operator	Syntax	Description
is	x is y	This returns True if both variables are the same object
is not	x is not y	This returns True if both variables are not the same object