Simple Operations

Table of Contents

Arithmetic Operations

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?

Students, write your response!

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`
  - `area = ?`

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?

Students, write your response!

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?

Students, write your response!

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  <=>  x = x + 3
x *= 3  <=>  x = x * 3
x **= 3 <=>  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  ⇐⇒ x = x + 3
x *= 3  ⇐⇒ x = x * 3
x **= 3 ⇐⇒ x = x ** 3
```

Arithmetic Operations

1. parentheses : `()`
2. power : `**`
3. unary minus : `-3`
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!

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 = ?

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()` Function

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⇒ `sep` → Default value ⇒ space

End of the line parameter⇒ `end` → Default value ⇒ 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...

Students, write your response!

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
```

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

- **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...

Students, write your response!

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...

Students, write your response!

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

not

and

or

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

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

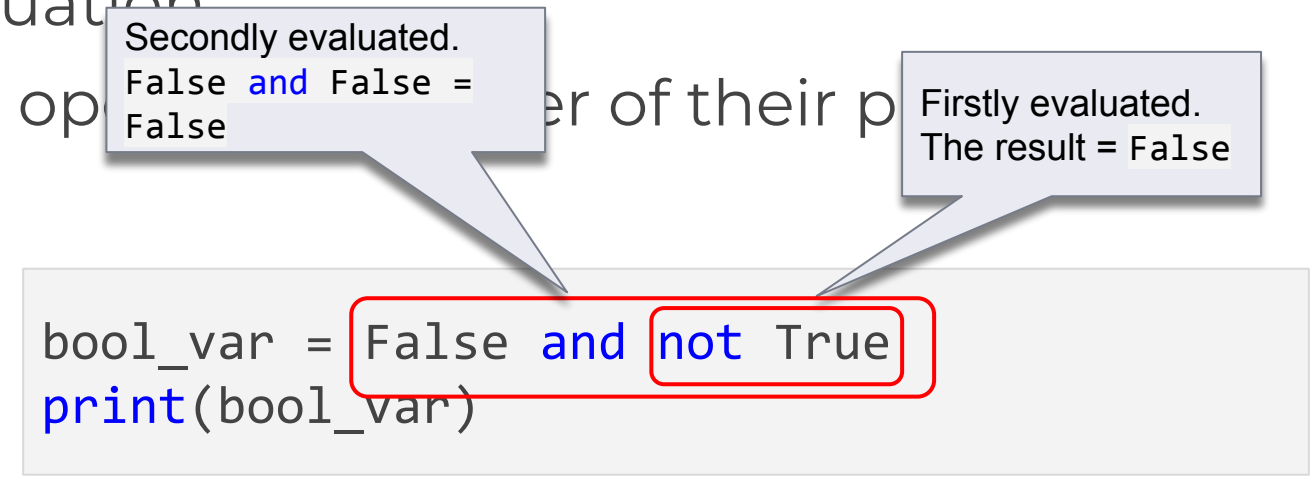
```
bool_var = False and not True
print(bool_var)
```

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 op er of their p

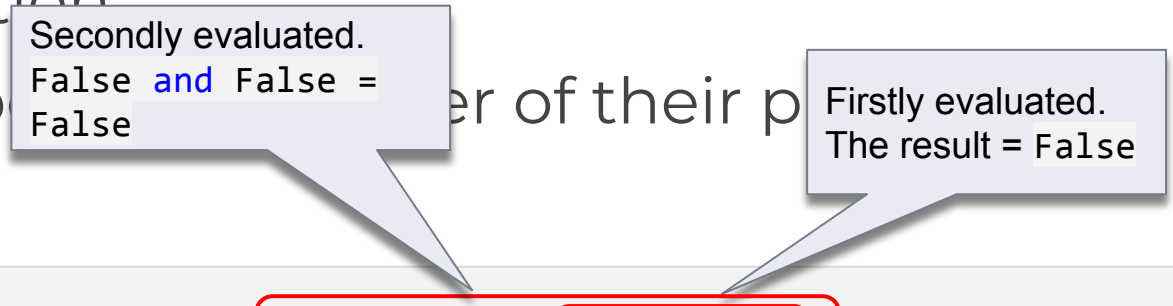
1. not
2. and
3. or



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 op er of their p

1. not
2. and
3. or

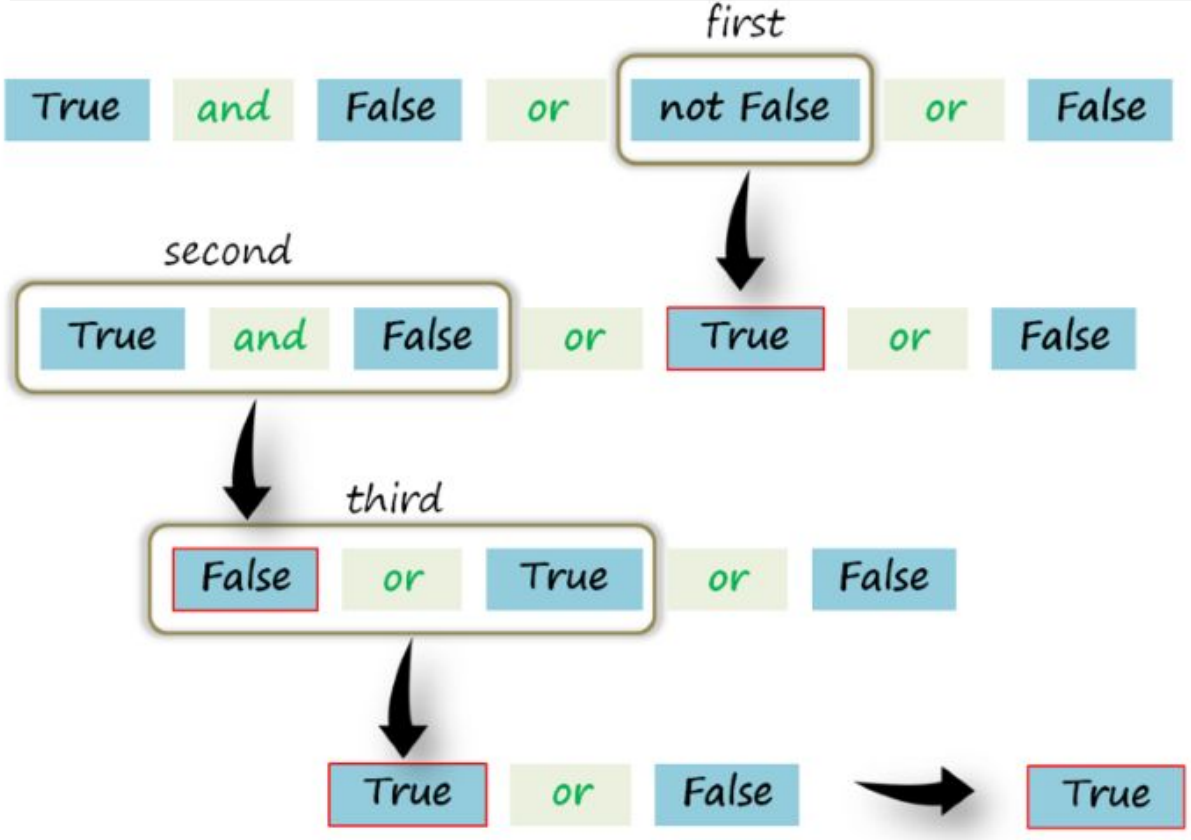



```
bool_var = False and not True
print(bool_var)
```

```
False
```

Order of Priority (review)

```
True and False or not False or False = ?
```



Truth Values of Logic Statements

Truth Values of Logic Statements

- **Falsy** values in Python:

- None
- Zero : `0, 0.0, 0j`
- Empty Seq. and collections : `'', [], {}`
- Any remaining value : `True`

Truth Values of Logic Statements

- Follow the `and` examples :

input :

```
1  print(2 and 3)
2
```

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

input :

```
1  print(1 and 0)
2
```

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

Truth Values of Logic Statements

- Follow the and examples :

input :

```
1  print(2 and 3)
2
```

output :

```
1  3
2
```

input :

```
1  print(1 and 0)
2
```

output :

```
1  0
2
```

Truth Values of Logic Statements

- Follow the `and` examples :

input :

```
1 print(2 and 3)
2
```

output :

```
1 3
2
```

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`.

input :

```
1 print(1 and 0)
2
```

output :

```
1 0
2
```

Truth Values of Logic Statements

```
print(2 and "hello world")
print([] and "be happy!")
print(None and ())
```

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

Students, write your response!

Truth Values of Logic Statements

```
print(2 and "hello world")
print([] and "be happy!")
print(None and ())
```

Output

```
hello world
[]
None
```

Truth Values of Logic Statements

- Follow the `or` examples :

input :

```
1  print(2 or 3)
2
```

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

input :

```
1  print(None or 1)
2
```

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

Truth Values of Logic Statements

- Follow the `or` examples :

input :

```
print(2 or 3)
```

output :

```
2
```

input :

```
print(None or 1)
```

output :

```
1
```

Truth Values of Logic Statements

- Follow the `or` examples :

input :

```
1 print(2 or 3)
2
```

output :

```
1 2
2
```

or

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

input :

```
1 print(None or 1)
2
```

output :

```
1 1
2
```

Truth Values of Logic Statements

```
print(2 or "hello world")
print([] or "be happy!")
print(None or ())
print({} or 0)
print({0} or False)
```

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

Students, write your response!

Truth Values of Logic Statements

```
print(2 or "hello world")
print([] or "be happy!")
print(None or ())
print({} or 0)
print({0} or False)
```

Output

```
2
be happy!
()
0
{0}
```

Truth Values of Logic Statements

- **Task**
  - First, Login to your LMS,
  - Then, click **here** to complete and submit the task.