Chapter 2 - Operators

Operators

Lab - Operators

Lab - Arithmetic Operators

Tutorial Lab 1: Arithmetic Operators

Arithmetic operations in Python are mostly the same as what you learned in math class. However, the symbols used in Python can be different.

Operation	Symbol	Notes
Addition	+	
Subtraction	-	
Multiplication	*	
Division	/	Always returns a float
Floor Division	//	Truncates the answer to be an integer
Power (exponent)	**	The inverse of the exponent will return a root
Modulo	%	Returns the remainder after division is performed

Use the text editor open in the left pane, and enter the following code:

```
print(10 + 5)
print(10 - 5)
print(10 * 5)
print(10 / 5)
print(10 // 3)
print(10 % 3)
```

To see what is happening in these calculations, click here. Click on the `Forward >` button to see how the calculations progress step by step.
1) Addition works as expected.
2) Subtraction works as expected.
3) Multiplication works as expected.
4) Division will always return a float, even if you are dividing two integers.
5) Floor division returns `3` because everything after the decimal point is ignored. The result is not rounded up or down.
6) Modulo returns `1` because that is the remainder (not the decimal) after division is performed.

Lab - Strings

Tutorial Lab 2: Strings

You can use the `+` and `*` operators with strings, even though the result is not based on math. Using the `+` operator with strings is called concatenation.

Use the text editor open in the left pane, and enter the following code:

```
string_1 = "hip "
string_2 = string_1 * 2
string_3 = "hoo"
string_4 = "ray!"
string_5 = string_3 + string_4
print(string_2, end='')
print(string_5)
```

To see what is happening in these operations, click here. Click on the `Forward >` button to see how the calculations progress step by step.
1) Assign the value `"hip "` to the variable `string_1`. Note the inclusion of a space after the word `hip`.
2) The variable `string_2` will have the value `"hip hip "` because `string_1 * 2` repeats the value of `string_1` two times.
3) Declare `string_3` and assign it the value `hoo`.
4) Declare `string_4` and assign it the value `ray!`.
5) Declare `string_5` and assign it the value of `string_3` combined with the value of `string_4` (`hooray!`).
6) Print the value of `string_2` (`hip hip`) and also remove the newline character.
7) Print the value of `string_5` (`hooray!`) to the end of `string_2`.

Lab - Order of Operations

Tutorial Lab 3: Order of Operations

Python uses PEMDAS when determining the order of operations.

P **P**arentheses
E **E**xponents - powers & square roots
MD **M**ultiplication & **D**ivision - left to right
AS **A**ddition & **S**ubtraction - left to right

PEMDAS

▼ **Modulo and PEMDAS**

Since modulo is based on division, modulo operations happen at the time of multiplication and division, going from left to right.

Use the text editor open in the left pane, and enter the following code:

```
print(5 * 8 // 3 + (18 - 8) ** 2 - 25)
```

Unfortunately, the code visualizer is not very helpful. It executes an entire line of code, so you will not see the order of operations. Below are the steps that Python takes when evaluating the code above.

1) `5 * 8 // 3 + (18 - 8) ** 2 - 25`
2) `5 * 8 // 3 + 10 ** 2 - 25`
3) `5 * 8 // 3 + 100 - 25`
4) `40 // 3 + 100 - 25`
5) `13 + 100 - 25`
6) `113 - 25`
7) `88`

Lab - Boolean Operators

Tutorial Lab 4: Boolean Operators

Boolean operators are those operators which will return either true or false.

Operation	Symbol	Notes
Equal to	`==`	The `=` operator is assignment operator, not the equality operator
Not equal to	`!=`	
Less than	`<`	
Less than or equal to	`<=`	
Greater than	`>`	
Greater than or equal to	`>=`	
And	`and`	Compares two boolean expressions. Both must be true for the whole to be true. Everything else is false.
Or	`or`	Compares two boolean expressions. Both must be false for the whole to be false. Everything else is true.
Not	`not`	Returns the opposite of a boolean expression.

Use the text editor open in the left pane, and enter the following code:

```
print((5 > 7) and (False or 1 < 9) or 4 != 5 and not 2 >= 3)
```

Unfortunately, the code visualizer is not very helpful. It executes an entire line of code, so you will not see each of the boolean expressions being evaluated. Below are the steps that Python talks when evaluating the code above.

1) `(5 > 7) and (False or 1 < 9) or 4 != 5 and not 2 >= 3`
2) `False and (False or 1 < 9) or 4 != 5 and not 2 >= 3`
3) `False and (False or True) or 4 != 5 and not 2 >= 3`
4) `False and True or 4 != 5 and not 2 >= 3`
5) `False and True or True and not 2 >= 3`
6) `False and True or True and not False`
7) `False and True or True and True`
8) `False or True and True`
9) `True and True`
10) `True`

Lab Challenge - Operators

Operators Challenge

Write a boolean expression that incorporates one of the equality operators, one of the less than operators, one of the greater than operators, and two different logical operators. The result of your boolean expression must be `False`.

Equality	Less Than	Greater Than	Logical
`==`	`<`	`>`	`and`
`!=`	`<=`	`>=`	`or`
			`not`