Lesson 4: Python Basics

Lesson 4: Python Basics 1

4.1. Introduction 2

4.2. Indentation and blocks 2

4.3. Comments 2

4.4. Variables and Constants 3

4.5. Data types and type casting 3

4.6. Operators 4

4.6.1. Arithmetic Operators 4

4.6.2. Assignment Operators 4

4.6.3. Comparison Operators 4

4.6.4. Logical Operators 4

4.7. Functions 5

4.8. Classes and Objects 5

Lesson 4: Review Questions 5

4.1. Introduction

Python can be executed by writing directly in the command line or y creating a python file on the server, using the .py file extension, and running it in the command line. You can also run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application / IDE on your system that supports Python.

4.2. Indentation and blocks

Indentation refers to the spaces at the beginning of a code line.
Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
Syntax Error:
if 5 > 2:
print("Five is greater than two!")
```

The number of spaces is up to you as a programmer, the most common use is four, but it has to be at least one.

Example

```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
      print("Five is greater than two!")
```

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

Example

```
Syntax Error:
if 5 > 2:
 print("Five is greater than two!")
      print("Five is greater than two!")
```

4.3. Comments

Python has commenting capability for the purpose of in-code documentation.
Comments start with a #, and Python will ignore them.
Comments can be used to explain Python code or to prevent Python from executing code.
Comments can be used to make the code more readable.
Comments can be used to prevent execution when testing code.
Python does not really have a syntax for multi-line comments. To add a multiline comment you could insert a # for each line

Example

Comments in Python:

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")#Another comment
```

4.4. Variables and Constants

Variables are containers for storing data values. Python has no command for declaring a variable. A variable is created the moment you first assign a value to it. Variables do not need to be declared with any particular type, and can even change type after they have been set.

Example

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

You can get the data type of a variable with the type()function-print(type(x)).

String variables can be declared either by using single or double quotes:

Example

```
x = "John"
# is the same as
x = 'John'
```

Variable names are case-sensitive.

Python allows you to assign a single value to several variables simultaneously.

```
x = y = 4
```

In Python, constants are usually declared and assigned in a module. Here, the module is a new file containing variables, functions, etc which is imported to the main file. Inside the module, constants are written in all capital letters and underscores separating the words.

4.5. Data types and type casting

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool

Binary Types:	bytes, bytearray, memoryview
None Type:	NoneType

If you want to specify the data type, it can be done with casting:

Example

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
w = float("4.2") # w will be 4.2
```

4.6. Operators

Operators are used to perform operations on variables and values. They can be grouped into: arithmetic, assignment, comparison and logical operators

4.6.1. Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations.

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor division

4.6.2. Assignment Operators

Assignment operators are used to assign values to variables.

Examples: x=5, x+=1.

4.6.3. Comparison Operators

Comparison operators are used to compare two values.

Operator	Name
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

4.6.4. Logical Operators

Logical operators are used to combine conditional statements i.e AND, OR, NOT.

4.7. Functions

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

A function is defined using the def keyword:

```
def my_function():
    print("Hello from a function")
```

To call a function, use the function name followed by parenthesis:

```
my_function()
```

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

Example

This function expects 2 arguments, and gets 2 arguments:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

If you try to call the function with 1 or 3 arguments, you will get an error.

4.8. Classes and Objects

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects

To create a class, use the keyword class.

Example: Create a class named MyClass, with a property named x:

```
class MyClass:
  x = 5
```

Now we can use the class named MyClass to create objects.

```
p1 = MyClass()
print(p1.x)
```

Lesson 4: Review Questions

1. State rules of naming variables in Python.
2. Write a function that accepts three values and return their product.
3. Explain the difference between = and = =. Give examples.
4. Create a class named Person, use the __init__() function to assign values for name and age.
5. Explain the use of keyword *pass.*