Welcome to ACIT4420

Scripting with Python

Nikolai Bjørnestøl Hansen
Roza Abolghasemi
OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Photo: Ronny Østnes / OsloMet

Who are we?

- **Roza Abolghasemi** (rozaabol@oslomet.no)
  - PhD-candidate in Artificial Intelligence
- **Nikolai Bjørnestøl Hansen** (nikobjo@oslomet.no)
  - Assistant Professor at Mathematical Modelling

OSLOMET

Course Description

- The course is divided into two parts
  - The first part (7 weeks) will consist of lectures from 10:30 to 12:15, and lab from 12:30 to 16:15
  - The second part (6 weeks) will consist of lab from 10:30 to 16:15
- During the first part we will learn how to use Python, and you will work through exercises during the lab.
- During the second part you will work on completing a programming project.
- Each student will deliver a individually coded project, and an individual project report.
- This project will determine your grade.

OSLOMET

Why Python?

- Python is a high-level, interpreted, untyped programming language

High-Level We do not need to care about the internal workings of the machine.

Interpreted Most programming languages need to be translated into machine code before they can run. Python does this "on the fly", giving slower programs but saving time when error-checking.

Untyped Python will try its best to figure out if you are working with numbers, spreadsheets, text, or other things. You don't have to tell Python what everything is beforehand

OSLOMET

Why Python?

- Python will give us a simple way of solving low-to-medium complexity tasks
- The programs will run slower than a program designed to do the same written in a low-level language
- But we will write the program faster
- And machine labor is cheaper than human labor

OSLOMET

Example Python

- This is what Python code looks like.

```python
import math

def show_area(shape):
    print("Area =", shape.area())

def width_of_square(area):
    return math.sqrt(area)
```

```python
class Rectangle:

    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

# Main Program
r = Rectangle(10, 20)

show_area(r)
```

OSLOMET

Follow along

- We will install python on our computer later, but for today, it should be enough with an online interpreter
- Here is one such interpreter:

  https://repl.it/languages/python

OSLOMET

Printing

- In Python, the function `print` sends the answers to the screen.
- If you don't `print` the answers, they will be computed, but you will not see them.
- As an example, if I want to print the text 'Hello, World!', I would write

```
print("Hello, World!")
```

- Notice the quotation marks around the text.
- You do not need quotation marks if you are printing a number.
- Try these:
  - Print your name.
  - Print a sentence.
  - Print some numbers.

OSLOMET

Arithmetic Operations

- You can use python as a calculator, using $+$, $-$, $*$, and $/$.
- Try to compute these:
  - $275 + 312$
  - $714 * 1218$
  - $1987 - 15 * 3$
  - $785/3$
- Python also has exponentiation, but 2^3 is written as 2**3.
- Try to compute:
  - 7^2
  - $4^{0.5}$
  - 2^{10}
- Python also has % and //, which is used for modulo and integer division.
- Try out: 17% 7 and 15 // 4

OSLOMET

Variables

- A variable is a name for information that you can re-use.
- You create variables like this:

```
pi = 3.14
x = 73
some_text = "Hello, there!"
x = 15
```

- After this code
  - the variable `pi` is equal to 3.14,
  - the variable `some_text` is equal to 'Hello, there!'
  - the variable `x` is equal to 15.
- Note how the original value of `x` was overwritten.
- Names for varables can only contain letters, numbers, and underscore.
- They cannot start with a number, and are case-sensitive.

OSLOMET

Data types

Python has many types of data. These are some of them

Boolean A boolean is the value `True` or the value `False`. It represents something being true or false.

Integer An integer is just any whole number. You can change a variable `x` into an integer by using the function `int(x)`.

Float A float is a decimal number. You can change a variable `x` into a float by using the function `float(x)`.

String A string is some text surrounded by (single or double) quotation marks.

Arrays and tuples Arrays and tuples are lists of elements, like `[3, 4, 5]` or `(3, 4, 5)`. They surrounded by brackets (arrays) or parentheses (tuples).

Dictionary A dictionary, like `d = {"pi": 3.14, "e": 2.72, "phi": 1.62}` lets us look up elements by names.

We will talk more about arrays, tuples, and dictionaries later.

OSLOMET

Examples

- For integers and floats, we just write the number to define the element.
- Strings can be defined either with single or double quotation marks.
  - If a string is surrounded by single quotation marks, you can use double quotation marks inside the string, and vice versa.
  - As an example, you could write `'Hello, "friend".'` or `"Hello, 'friend'."`, but not `"Hello, "friend"."`.
- If you want multiple lines of text in a string, you can use three double quotation marks:

```
"""
This is a lot
of text.
I need multiple lines.
"""
```

- You could also use `\n` as a newline-symbol. `"This \nwill be \nover multiple lines."`

User input

- If you write `x = input()`, the program will ask you for input, and save what you write in the variable `x`.
- If we write `x = input("What is x?  ")` the program will print `What is x?` in front of the input.
- This saves the input as a string.
- Try these:
  - Take an input, and print it out.
  - Take an input, multiply it by two, and print out the answer.
- What did you get for the last example?
- We need to use `int(x)` or `float(x)` to tell Python that the input was a number.
- An example solution:

```
x = int(input())
print(2*x)
```

Comparisons

We can compare two elements by using the symbols

- `==` Checks if two elements are equal. Note the double equal sign. We use single equal sign to define variables.
- `>`, `<` Checks if one element is greater or smaller than the other.
- `>=`, `<=` Checks if one element is greater than or equal, respectively smaller than or equal to the other.
- `!=` Checks if two elements are not equal.

All of these will give us either `True` or `False` as an answer. We will use `True` and `False` in the next slide.

The operators `not`, `and`, and `or`

If we have something that is either `True` or `False`, we can combine them with the operators

`not` The operator `not` inverts the value of `x`. If `x` is `True`, then `not x` is `False`, and if `x` is `False`, then `not x` is `True`.

`and` The operator `and` combines two boolean values, and returns `True` only if both were `True`. So `x and y` is true if and only if both `x` and `y` are `True`.

`or` The operator `or` combines two boolean values, and returns `False` only if both were `False`. So `x or y` is `True` if at least one of `x` and `y` is `True`.

The `if`-statement

- An `if`-statement is a way to run some code only if something is true.
- It looks like this:

```
if condition:
    body
```

- This code will check if `condition` is `True`, and if it is, we will run the code in `body`.
- Note that the `body` is indented. Python uses indented code to figure out what is part of the body, and what is no longer part of the `if`-statement.
- We will see indentation used to separate code more in the future.
- In other programming languages, code is indented to increase readability. In Python, it is required.

OSLOMET

Example of `if`-statement

```
x = int(input("What is x? "))
y = int(input("What is y? "))
if x == 3 and y < 5:
    z = x*y
    print("Wow, x is three, and y is smaller than five. Their product is", z)
print("Wow, I'm done. Phew!")
```

- This program asks for two integers, checks if the first is equal to three, and the second is smaller than five.
- If both statements are true, the program will compute the product, and print the first `print`-statement.
- The last `print`-statement is not indented, and will therefore always run.
- Copy this code and try it out! Remember to check the indentation!
- Try using the variable `z` after the `if`-statement. What happens?

OSLOMET

The `else`-statements

- Sometimes we want to run one thing if something is true, and another if it is false.
- For this we have the `else`-statement.

```
if condition:
    true_body
else:
    false_body
```

- This code will run `true_body` if `condition` is `True`, otherwise it will run `false_body`.
- Try to write a program where you input a number, and the program prints "Hooray!" if the number is larger than 100, and "Boo!" if it is not.

OSLOMET

The `elif`-statement

If we want to check multiple conditions, we can use `elif` (else-if) statements.

```
if condition1:
    body1
elif condition2:
    body2
elif condition3:
    body3
else:
    body4
```

- This would run `body1` if `condition1` is true.
- If `condition1` is false and `condition2` is true, we run `body2`.
- If both previous conditions are false, and `condition3` is true, we run `body3`.
- If all previous conditions are false, we run `body4`.
- We can have as many `elif`-statements as we want.

OSLOMET

Loops

- Loops are used when we want to repeat code.
- There are two types of loops used in python, the `while`-loop and the `for`-loop.
- We will learn the `while`-loop today, and the `for`-loop later.
- A `while`-loop looks like this:

```
while condition:
    body
```

- The program will check if `condition` is `True`, and if it is, the `body` wil run.
- It will then again check if `condition` is `True`, and if it is, run `body` again.
- This continues until `condition` is no longer true.
- Possibly the program runs forever.

OSLOMET

Example `while`-loop

- Here is an example program using a `while`-loop

```
print("The program will add up all numbers from 0 up to x.")
x = int(input("What is x? "))
s = 0
i = 0
while i <= x:
    s = s + i
    i = i + 1
print("The sum is", s)
```

- Note how we increase the counter `i` inside of the loop. If we did not, it would run forever.
- We keep summing as long as the counter is smaller than or equal to `x`.
- Note that if the last `print`-statement had been indented, the program would print the intermediary sum at each step.

OSLOMET