Quickguide in Python

# Syntax, variables & data types

### Syntax & whitespace

* Whitespace is heel belangrijk in python. Plaats geen whitespaces waar dit niet nodig is, aangezien whitespace een deel van de syntax is.
* Indentatie wordt gedaan met 4 spaties.

### Variables & data types

* Je hoeft **geen** data type te geven aan een variable, dit gebeurt automatisch.
* Een value van een variable kan gereassigned worden door de variable opnieuw te vullen met een nieuwe waarde.

|  |  |
| --- | --- |
| **Code** | **Result** |
| print X | Prints X |
| variable\_name = X | Plaatst X in variable\_name  Dit kan eender welk data type zijn |
| #Single line comment | Alles na de # wordt gecomment |
| “””Multi-Line Comments  komen tussen  tripple quotations  “”” | Alles tussen de “”” (tripple quotations) wordt gecomment |
| my\_string = “dit is een string” | Data type: string  Value: dit is een string |
| my\_int = 23 | Data type: integer  Value: 23 |
| my\_float = 1.25 | Data type: float  Value: 1.25 |
| my\_bool = True | Data type: boolean  Value: True  Note: “True & False horen met een hoofdletter geschreven te worden” |

Voorbeeldcode:

melk = “Volle melk”

hoeveelheid = 3

print melk

print hoeveelheid

Result:

Volle Melk

3

# Math

### Math & operators

* Wiskunde in programmeren wordt gedaan met operators. Operators zijn “+ - \* /”. Hierop zijn nog enkele variaties zoals modulo en macht.
* Gebruik een of meerdere wiskundige values, zet er een operator tussen en je krijgt een wiskundige bewerking.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Result** | **Syntax** | **Result** |
| + | Adition (optellen) | addition = 158 + 322 | 488 |
| - | Subtraction (aftrekken) | subtraction = 680 - 144 | 536 |
| \* | Multiplication (vermenigvuldiging) | multiplication = 51 \* 86 | 4386 |
| / | Division (deling) | division = 125 / 25 | 5 |
| \*\* | Power (macht) | power = 2 \*\* 3 | 8 |
| % | Modulo (modulo, rest) | modulo = 3%2 | 1 |

Voorbeeldcode:

meal = 44.50

tax = 0.0675

tip = 0.15

meal = meal + meal \* tax

total = meal + meal \* tip

print total

Result:

54.63

# Strings

### Basics

Een string is de letterlijke waarde van de inhoud. Een string “17” is exact dat, de letters “1” en “7”. Strings hebben geen wiskundige inhoud en hier kan men dan als gevolg ook niet mee rekenen. Bijvoorbeeld: “17” + “17” wordt “1717” en niet “34”.

### Concatinatie

Met dit voorbeeld hebben we ook ineens concatenatie gezien. Concatenatie is het samenvoegen van verschillende waarde. Dit kan met alle datatypes. Dit kan gedaan worden door reeds bestaande variables en nieuwe strings.

Voorbeeldcode:

aantal\_glazen = 25

print “Aantal glazen op de kast: “ + aantal\_glazen + “ stuks.”

Result:

Aantal glazen op de kast: 25 stuks.

### Escape

Maar wat nu als we het volgende willen uitschrijven: Ik wil het boek genaamd “Lord of the Rings” kopen.

Het probleem dat we hier hebben is dat we de string sluiten net voor het wordt Lord.

Om dit op te lossen gebruiken we escapes. Het escape symbool in Python is “\”. De eerste letter of het eerste symbool dat je hiervoor zet wordt letterlijk geschreven en niet gezien als syntax. De code ziet er dan zo uit.

Voorbeeldcode:

print “Ik wil het boek genaamd \”Lord of the Rings\” kopen.

### Access by Index

Elke letter in een string heeft een nummer. Dit noemen we de index. De index begint te tellen vanaf 0. Als we bijvoorbeeld de 5e letter willen hebben van een string gaan we als volgt tewerk, rekening houdende dat we bij 0 beginnen te tellen.

Voorbeeldcode:

fifth\_letter = “Een string”

print fifth\_letter[4]

Result:

S

### String methods

String methods laten je bepaalde acties uitvoeren op een string.

|  |  |
| --- | --- |
| String Method | Function |
| len() | Geeft de lengte van een string terug |
| lower() | Zet heel de string om in lower case |
| upper() | Zet heel de string om in upper case |
| str() | Maakt strings van niet string variables  (dus maakt bijvoorbeeld een string van een float) |

Voorbeeldcode 1:

parrot = "Norwegian Blue"

print len(parrot)

Result:

14

Voorbeeldcode 2 (andere schrijfwijze, doet hetzelfde):

parrot = "Norwegian Blue"

print parrot.len()

Result:

14

### String formatting

Je kan placeholders gebruiken die vervangen kunnen worden door strings. We gebruiken de “%” operator om de “%s” placeholders te vervangen door variables. Je moet dezelfde hoeveelheid “%s” placeholders hebben als het aantal variables. De syntax ziet er als volgt uit:

Voorbeeldcode:

name = raw\_input("What is your name?")

quest = raw\_input("What is your quest?")

color = raw\_input("What is your favorite color?")

print "Ah, so your name is %s, your quest is %s, " \

"and your favorite color is %s." % (name, quest, color)

Note:

raw\_input() vraagt de user om input via een prompt met een vraag en plaats de gegeven value in een variable.

# Date & Time

Soms is het nodig om te weten wanneer we iets gedaan hebben of om andere bewerkingen te doen met tijd. Om te kunnen werken met tijd moeten we eerst de module inladen. Dat gaat met de volgende lijn code “from datetime import datetime”. Als we dit bovenaan van onze code zetten zal de module ingeladen worden en kunnen we ermee beginnen werken.

|  |  |
| --- | --- |
| **Code** | **Result** |
| from datetime import date | Import datetime module |
| datetime.now() | De datum en tijd op het punt van declaratie |

Voorbeeldcode:

from datetime import datetime

now = datetime.now()

Eens we de datetime in een variable hebben gestoken kunnen we details eruit vissen.

|  |  |
| --- | --- |
| Code | Result |
| now.year | Geeft het jaar terug van de datetime variable now |
| now.month | Geeft de maand terug van de datetime variable now |
| now.day | Geeft de dag terug van de datetime variable now |
| now.hour | Geeft het uur terug van de datetime variable now |
| now.minute | Geeft de minuut terug van de datetime variable now |
| now.second | Geeft de second terug van de datetime variable now |