Start to Program: Python, Les 5

type-bepaling

Door ***type()*** te gebruiken verkrijg je het data-type van wat er tussen de haken staat. Je kan dan via een conditie gaan controleren met welk data-type je te maken hebt.

Schrijf een conditie waarmee je het datatype van een var test.

- Maak een variabele met een string, float of integer
- Doorloop met een conditie alle data-types. Maak gebruik van de afkortingen die je zou gebruiken bij een omzetting: int, str, float
- Print met welk data-type je te maken hebt.

Dit werkt zeer goed met variabelen die reeds zijn opgegeven, maar hier zijn beperkingen aan. Die worden verder in de cursus bekeken.

Een tweede mogelijkheid is om de meerdere types tegelijk te testen. Dit gebruik je wanneer je nauwkeurig moet zijn.

```
isinstance(waarde, type)
OF
isinstance(waarde,(type1,type2))
```

Zo kan je controleren of iets een *number* is zonder specifiek voor een *int* of *float* te vragen.

input

Tot hiertoe heb je steeds vaste waarden toegekend aan je variabelen. Dat beperkt zeer hard de interactie van je progamma. Alles zal vanaf het begin moeten ingesteld zijn.

Je kan de gebruiker vragen om een bepaalde waarde op te geven. Deze wordt dan best opgeslagen in een variabele en kan zo verder gebruikt worden in het programma.

```
naam_var = input('vraag om input')
```

- eerst kies je een duidelijke naam voor de var waarin het antwoord opgeslagen wordt.
- Input() gaat het programma stoppen en invoer verwachten.
- Tussen de haakjes en tussen aanhalingstekens kan je een string die de invoer die je wilt krijgen omschrijft.

Schrijf een programma dat eerst om je naam vraagt.
Print deze variabele tekst dan af in bijvoorbeeld een begroeting.

String naar number

LET OP! Alle invoer zal worden opgeslagen met het data-type *string*. Als je een een *number* wil zal je deze moeten omzetten.

Nu kan je geen gebruik maken van *type()* of *isinstance()*. Je zal eerst de *string* naar een *int* of *float* moeten omzetten voor je kan controleren of ze wel tot dat type behoort.

Als je de invoer omzet naar het gewenste datatype en dit is niet mogelijk zal er een error verschijnen.
Hiervan kan je gebruik maken. In plaats van de error rechtstreeks aan de gebruiker te tonen kan je ook een specifieke boodschap meegeven.

Je gaat eerst ***proberen*** om het datatype naar een *integer* of *float* om te zetten. Als dit niet lukt zal deze omzetting niet doorgaan. Je krijgt een ***waarde error***. In plaats hiervan toon je je boodschap.

```
var = input('vraag')
try:
        var = int(var)
except ValueError:
        print('eigen error boodschap')
```

Opdracht:
Vraag om een getal en slaag deze op in een var.
Zet deze om naar een int OF float. Je kan hier met nesting werken.
Als dit niet lukt geef je een boodschap die weergeeft dat je geen getal opgaf.

Uitbreiding:
Hoe zou je door een conditie toe te voegen een hoger lager spelletje kunnen maken?

Lussen

While

Soms wil je in herhaling vallen. Bijvoorbeeld zolang de invoer niet het juiste data-type is wil je de vraag blijven herhalen tot de gebruiker wel de juiste invoer geeft.

Je herhaalt je code zolang aan een bepaalde conditie wordt voldaan. Pas dan zal er verdergegaan worden naar de code die erop volgt.

```
while conditie True is:
    voer code uit.
```

Hier maak je gebruik van dezelfde operatoren als bij condities (if-structuur)

operator	Wat doet de operator
==	Zijn de waarden gelijk?
!=	Zijn de waarden niet gelijk?
<	Is de waarde kleiner dan de andere?
>	Is de waarde groter dan de andere?
<=	Is de waarde kleiner of gelijk aan?
>=	Is de waarde groter of gelijk aan?

Opdracht:
Blijf om invoer van een wachtwoord vragen **zolang** die invoer niet gelijk is aan het vooropgestelde wachtwoord.

Denk eerst even na. Wat heb je allemaal nodig?
Welke var? Welke aanpak van operator? Waar vraag je de input? Wat doe je er dan mee? …

Hoe geef je een extra boodschap bij een foute poging?

HOGER – LAGER uitwerking

Eerst ga ja een random getal tussen 1 en 100 genereren. Dit doe je als volgt.
Bovenaan je code zal je eerst een module met extra methodes moeten importeren. Later in de cursus wordt er dieper ingegaan op modules.

```
import random
```

Vervolgens slaag je een random getal op in een variabele.

```
te_raden = random.randint(1,100)
```

TIP! Print vervolgens deze variabele. Anders ga je veel tijd verspelen met het gokken naar het juiste getal om te controleren of je code werkt. Als je toch vast 'loopt' gebruik dan ctrl+c om je programma te sluiten.

VERSIE 1:
Maak nu een eenvoudige versie van hoger lager. Je blijft naar een getal vragen tot het te_raden getal geraden wordt.
Alle invoer is toegestaan.

Wat gebeurt er als je een string invoert?
Hoe los je dit op?

VERSIE 2:
a) Voeg een variabele toe die het aantal gokken opslaat. Geef wanneer het getal geraden is weer hoeveel gokken je nodig had.

b) Beperk het aantal gokken tot 10.

TIP: Voeg een extra conditie toe aan de while-lus.

Klopt de eindboodschap nog? Hoe los je dit op?

VERSIE 3:
a) Verfijn je programma zodat er ook gemeld wordt of je hoger of lager moet gokken.

b) Geef weer hoeveel keer je nog kan gokken, NIET hoeveel je al gegokt hebt. Kan je dit variabel oplossen?

c) Als er een string ingegeven wordt verlies je geen beurt.

d) Als je in 1 gok het getal raadt klopt je boodschap dan nog?

VERFIJNING:
Zorg dat er alleen maar een getal geraden kan worden tussen 1 en 100.
Maak de vraag om een getal variabel. vb. te_raden = 50, geraden = 70. Dus LAGER. De volgende vraag zou dan een getal tussen 1 en 70 moeten zijn.
Pas de tekst aan en zorg ervoor dat er niet meer boven 70 geraden kan worden.