Afbeelding met tekst, Graphics, tekenfilm, schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.

Dhr. Renaud Leroy – Mevr. Carmen Van Belle

Campus impuls | Wolfputstraat 42, 9041 Gent

Informaticawetenschappen

Inhoudsopgave

[Python essentials: datatypes en variabelen 5](#_Toc209412200)

[Wat is Python 5](#_Toc209412201)

[Variabelen 5](#_Toc209412202)

[Uitvoer (print) 6](#_Toc209412203)

[Invoer 6](#_Toc209412204)

[Data-types 7](#_Toc209412205)

[String 7](#_Toc209412206)

[Oefeningen: string 8](#_Toc209412207)

[Int 9](#_Toc209412208)

[Float 10](#_Toc209412209)

[Converties 10](#_Toc209412210)

[Oefeningen 11](#_Toc209412211)

[List 13](#_Toc209412212)

[ForEach 13](#_Toc209412213)

[For 14](#_Toc209412214)

[Oefeningen 15](#_Toc209412215)

[Bijlagen 18](#_Toc209412216)

[Bijlage 1 – titel 18](#_Toc209412217)

[Bibliografie 19](#_Toc209412218)

Introductie

Digitale systemen en algoritmes zijn in onze maatschappij nauwelijks nog weg te denken. Van het eten dat op jouw bord komt, tot de games die je speelt, in elk aspect van ons dagelijks leven is informatica aanwezig. Soms op de achtergrond, zonder dat je erbij stil staat.

In deze cursus zal je een inleiding krijgen tot programmeren en oplossen van algoritmes. Bij het programmeren kiezen we voor Python, aangezien dit één van de meest gebruikte programmeertalen is, zowel in het professionele leven als later in het hoger onderwijs. Algoritmes zijn (vaste) stappenplannen die een computer kan uitvoeren om een taak uit te voeren. Dit kan héél simpel het berekenen van een getal zijn, tot het sorteren van data of het vinden van routes in een doolhof.

Voor het programmeren, gaan we gebruik maken van Visual Studio Code (hierna: VS Code). Dit is een populaire IDE (ontwikkelomgeving), waarin je eenvoudig extra stukjes software kan toevoegen. Ook gaan we werken met GitHub. Dit is een platform waar je code kan opladen en delen met anderen. Dit is geïntegreerd met VS Code, zodat je héél snel en zonder al te veel tussenstappen je code kan krijgen. O.a. de oplossingen op de oefeningen in deze cursus zullen via GitHub worden aangeleverd. Meer info hierover verder in de cursus.

# Python essentials: datatypes en variabelen

## Wat is Python

Python is een programmeertaal, net als bijvoorbeeld C, C++, C#, Java en JavaScript. Met een programmeertaal kun je **instructies** geven aan een computer. Deze instructies worden uiteindelijk omgezet in machinetaal, zodat de computer ze kan uitvoeren.

Één groot voordeel van Python is de **toegankelijkheid**. Andere programmeertalen bevatten vaak veel *syntax*, specifieke tekens die je moet gebruiken om code te schrijven. Python leest en schrijft vaak als een Engelstalige tekst. Neem bijvoorbeeld onderstaand stukje code:

1. leeftijd = 15
2. if leeftijd < 18:
3. print(“Je bent minderjarig”)

Als je deze code leest, zou je al een idee moeten hebben wat er gebeurd. We kunnen dit iets meer concreet uitschrijven: De leeftijd is 15 jaar. Als de leeftijd kleiner is dan 18 jaar, dan moet je printen dat de persoon minderjarig is. Maar geen zorgen, alles wordt stap voor stap uitgelegd in de volgende hoofdstukken.

Een ander voordeel van Python is de **toepassingen**. Veel informaticasystemen draaien op Python code, of kunnen d.m.v. Python worden aangestuurd. In de beginjaren van de digitale revolutie zijn er veel programmeertalen ontstaan (en verdwenen), maar sinds enkele jaren lijkt de strijd gestreden te zijn en is Python sowieso een blijver.

## Variabelen

Elke programmeertaal, en dus ook Python, werkt met variabelen. Dit zijn stukjes computergeheugen die worden vrijgehouden om data naartoe te kunnen schrijven. Bij het programmeren onthoud je dan de locatie (adres) van het dit deeltje computergeheugen, zodat je dit kan uitlezen of iets naartoe kan schrijven.

Een voorbeeld van een geheugenadres is bijvoorbeeld *0x7ffe5367e044*. Good luck! Dit is natuurlijk niet te onthouden, en dus ook niet werkbaar. Daarom kan je met variabelen werken. Dit zijn verwijzingen naar een geheugenlocatie, maar je kan die een eigen naam geven. Als we het bovenstaande voorbeeld nemen, is *leeftijd* hier een variabele. Dit verwijst naar een geheugenlocatie waarin het getal 15 zit opgeslagen.

Bij de naamgeving van een variabele, zijn er wel enkele voorwaarden / informele regels:

* Begint met een letter of underscore ( \_ )
* Bevat enkel letters, cijfers en underscores
* Zijn hoofdlettergevoelig

In het onderstaande voorbeeld zie je een paar voorbeelden van correcte en foute variabelen.

1. 1enaam = “Marcel” begint niet met letter of underscore
2. voor.naam = “Marcel” bevat geen speciaal teken
3. leeftijd = 15
4. Leeftijd = 18
5. opgelet: leeftijd en Leeftijd zijn aparte variabelen

Tenslotte heb je ook nog verschillende manieren om namen van variabelen die uit meerdere woorden bestaan, leesbaar te kunnen opstellen. Bij *camelCase* en *PascalCase* schrijf je alle woorden aan elkaar, maar laat je ieder woord met een hoofdletter beginnen (*camelCase* begint met kleine letter). Bij *snake\_case* schrijf je de woorden van elkaar, maar gebruik je een underscore om ze aan elkaar te koppelen. Je bent vrij om eender welke stijl te gebruiken, maar gelieve deze niet te combineren.

## Uitvoer (print)

Om variabelen op het scherm af te drukken, gebruik je de functie *print*. Net zoals bij functies in MS Excel gebruik je haakjes bij functies om *parameters* mee te geven. Parameters zijn gegevens die een functie nodig heeft om te kunnen werken. De print functie schrijft iets op het scherm. Die heeft dus één parameter, nl. dat iets dat op het scherm moet komen.

1. leeftijd = 15
2. print(leeftijd)
3. print(3)

## Invoer

Hierboven hebben we gezien hoe je variabelen kan aanmaken en kan afdrukken op het scherm. We kunnen echter ook tekst inlezen. Dit kan met de functie *input*. Deze functie heeft, met als *print*, haakjes met hiertussen een parameter. Dit keer is de parameter welke vraag op het scherm moet verschijnen alvorens de gebruiker tekst kan intypen. Dit is echter optioneel. Dit wil zeggen dat je ook niets kan typen (dan verschijnt er ook geen vraag). De *input* leest hetgeen de gebruiker intypt in tot er op enter wordt geduwd en geeft dit terug.

1. naam = input('Wat is jouw naam? ')
2. print(naam)

## Data-types

Bij het programmeren werken we veel met data. Maar er zijn verschillende soorten data. Bekijk hieronder de voorstelling van een persoon:

Hallo, ik ben jullie leerkracht, *Renaud Leroy*. Ik ben *32* jaar, ben *1,70* meter groot en heb *3* kinderen. Mijn hobby’s zijn: *lezen, voetbal, gamen, fietsen en muziekspelen*. Ik speel *wel* dwarsfluit maar *geen* piano. Mijn lievelingsfilm is *The Lord of the Rings: The Fellowship of the Ring*.

In *cursief* stonden alle stukjes data die interessant zouden kunnen zijn voor een algoritme. Maar niet alle data is dus van hetzelfde type.

|  |  |  |
| --- | --- | --- |
| Data | Soort data | Datatype |
| *Renaud Leroy, LOTR: The Fellowship of the Ring* | Tekst | string |
| *32, 3* | Gehele getallen | int(eger) |
| 1,70 | Kommagetallen | float |
| *Lezen, voetbal, games, fietsen, muziekspelen* | Lijst | array |
| *wel, geen* | Waar/Vals | boolean |

Zo heb je bijvoorbeeld stukjes tekst, zoals een naam of de titel van een film. Je hebt ook getallen, zoals een leeftijd, een aantal kinderen of een grootte, maar dit kunnen dus gehele getallen of kommagetallen zijn. Je kan ook wel of niet iets hebben, en al deze soort data kan je ook nog eens in lijsten steken. Hieronder overlopen we per datatype wat je ermee kan doen.

## String

Een string is een stuk tekst. Eigenlijk is dit een lijst (array) van tekens (char (van character)), maar omdat tekst zoveel voorkomt bij het programmeren beschouwen we dit als een geheel.

Belangrijk is dat tekst steeds tussen aanhalingstekens moet worden geplaatst. Dit kunnen dubbele of enkele aanhalingstekens zijn.

1. eerste\_tekst = "Hallo, ik ben tekst"
2. tweede\_tekst = 'En ik ben ook tekst'

Maar je mag ze niet met elkaar combineren om een string te openen en af te sluiten.

1. tekst = 'Hallo, ik ben tekst"

Je kan ze wel met elkaar gebruiken, maar dan om deze als deel van de tekst te gebruiken:

1. zin = 'Ik zei: "Hallo!" '
2. tekst = "Ik ga 'direct' mijn huiswerk maken."

Als je ze allebei wil gebruiken, kan je gebruik maken van een \. Deze plaats je voor het aanhalingsteken, zodat dit niet als het einde van de string wordt beschouwd maar als een teken van de tekst.

1. zin = "Ik zei: \"Ik ga 'direct' mijn huiswerk maken.\""

Je kan ook teksten met elkaar combineren met een +-teken.

1. naam = "Obama"
2. voornaam = "Barack"
3. president = voornaam + naam ("BarackObama")

Het +-teken heeft echter het nadeel dat dit bij andere datatypes niet compatibel is. Beter is om een *format* te gebruiken. Hiervoor typ je een *f* voor de een string en gebruik je accolades als placeholder om variabelen in de string in te voegen.

1. leeftijd = 64
2. voornaam = "Barack"
3. naam = "Obama"
4. president = f"{voornaam} {naam} is {leeftijd} jaar."

## Oefeningen: string

1. Schrijf de tekst "Hello World!" op het scherm.
2. Maak een variabele (*naam)* en schrijf hier "John" in weg. Schrijf vervolgens op het scherm: "Hallo *John*"
3. Vraag aan de gebruiker diens naam (*"Wat is jouw naam? "*) en de titel van diens lievelingsgame (*"Wat is jouw favoriete game? "*). Schrijf vervolgens op het scherm: *naam* speelt graag *titel*!
4. Vraag naar een favoriete quote uit een film (*"Favoriete quote: "*) en schrijf op het scherm: Mijn favoriete quote: "*qoute*"
5. Schrijf de volgende tekst op het scherm, zonder gebruik te maken van variabelen: *Ik zei tegen hem: "Ik bel je 's avonds"*
6. Vraag aan de gebruiker de naam van diens favoriete huisdier ("Wat is jouw favoriete huisdier? ") en schrijf:Ik hou zo van *huisdier.*
7. Vraag aan de gebruikers eerst diens naam (*"Naam: "*), daarna diens favoriete film (*"Favoriete film: "*) en toon het volgende op het scherm:Hallo *naam*, ik kijk ook graag naar *film*.

## Int

Net als bij andere programeertalen is er een verschil tussen een kommagetal en een geheel getal. Een *int* is een geheel getal. Hiermee kan je de standaardbewerkingen +, -, \* (maal) en / doen. Let op: er zijn 2 soorten delingen. Met één enkele / voer je een gewone deling uit. Als je echter 2 / gebruikt, dan voer je een deling uit waarbij je afrond naar het dichtste gehele getal naar onder.

1. a = 3
2. b = a + 1
3. c = 9 / b
4. d = 9 // b

Hierboven is de waarde van b dus 4 (3 + 1), c is dan 2.25 en d is dan gewoon 2.

Je kan uiteraard ook haakjes gebruiken om voorrang te geven aan bepaalde bewerkingen.

Handig is ook het gebruik van \*\*. Hiermee kan je namelijk een macht uitrekenen. Ervoor schrijf je dan het grondtal, erachter de exponent. Zo is de waarde van a in het onderstaande 9.

1. a = 3 \*\* 2

Bij wiskunde heb je echter ook gezien hoe je wortels (bv. vierkantswortel) kan uitrekenen d.m.v. een macht. Door de inverse van de exponent te nemen, kan je dus ook nde-machtswortels berekenen.

1. a = 9 \*\* (1/2)

Tenslotte is er ook nog de modulo. Dit is de restberekening van een deling tussen twee getallen. Zo zie je in het onderstaande voorbeeld dat 5 % 3 de restwaarde is van 5 te delen door 3. 3 past 1 keer in 5, dan hou je nog 2 over, dus 5 % 3 is dus gelijk aan 2.

1. var restwaarde = 5 % 3;

## Float

*Float* is het representeren van een rationaal getal. Let op: zelfs ook bijvoorbeeld 3,0 is een kommagetal. De reden waarom *float* en *int* dan nog apart bestaan, heeft te maken met de geheugenallocatie van deze variabelen. Beiden gebruiken 32 bits. Dit wil zeggen dat wanneer je een variabele aanmaakt van het type *int*, deze 32 bits computergeheugen reserveren. Ook al gebruiken ze niet alle bits om hun getal te onthouden, toch worden deze 32 bits vastgelegd en kunnen die niet door een andere variabele worden gebruikt. Zo bestaat het getal 5 uit 3 bits (101), maar toch zullen de andere 29 bits (allemaal 0), toch niet beschikbaar blijven voor andere variabelen.

Zo heeft een *int* met de 32 beschikbare bits een bereik van -2 147 483 648 tot 2 147 483 647, en heeft *float* 'maar' een bereik van ±16 777 216. Uiteraard kan deze wel meer significant worden op het gedeelte na de komma, afhankelijk van hoeveel bits daarvoor resteren.

Let op: in Python is Engels de voortaal, dus schrijf je decimale getallen met een punt (bv. 3.0)

1. a = float("1,2")  
    ^^^^^^^^^^^^
2. ValueError: could not convert string to float: '1,2'

Met de functie *round()* kan je tenslotte ook getallen afronden. Je geeft hier 2 parameters mee. Eerst het getal dat je wenst af te ronden, tenslotte het aantal decimalen. Tip: je kan bij het aantal decimalen ook een negatief aantal gebruiken om zo op tientallen, hondertallen, etc. af te ronden.

1. getal = 172.238
2. getal\_172\_24 = round(getal,2)
3. getal\_200 = round(getal,-2)

## Converties

Misschien had je er al op gelet, maar *input()* geeft een *string* terug (tekst). Dit is echter een probleem voor getallen, want de *string* (lees: tekst) 5 – 3 (lees: getal 3), heeft geen enkele betekenis. Daarom krijg je ook een foutmelding als je het volgende probeert, met als invoer "3":

1. a = input("Geef een getal: ")
2. b = a + 1
3. b = a + 1
4. ~~^~~
5. TypeError: unsupported operand type(s) for +: 'str' and 'int'

Je moet dus de tekst (bv. "3") omzetten naar het getal (bv. 3). Dit kan met de functie *int()* (en *float()* voor kommagetallen). Wat er gebeurt wanneer we een ongeldige waarde ingeven, bv. "abc", zien we later.

Je kan ervoor kiezen om de conversie in één of twee stappen te doen. Ofwel schrijf je eerst de *string*-waarde van *input()* in een variabele weg, en wijs je in een tweede stap aan de variabele een nieuwe waarde toe, nl. het resultaat van de conversie-functie, die als enige parameter de oude waarde krijgt. Of je doet de twee stappen tegelijk, waarbij je de uitvoer van *input()* rechtstreeks naar de conversie-functie schrijft, die dan aan een variabele wordt toegewezen.

1. **a = int(input("a = "))**
2. b = input("b = ")
3. b = int(b)
4. c\_tekst = input("c = ")
5. c = int(c\_tekst)

## Oefeningen

1. Maak een programma dat 2 getallen aan de gebruiker vraagt ("a = ", "b = ") en van deze getallen het volgende toont: som, verschil, product, quotiënt, macht, wortel.
2. Maak een programma dat een invoer 'x' bevraagt ("x = ") en vervolgens de functiewaarde van de volgende functies berekent:
3. Maak een programma dat de lengte ("Lengte (m): ") en het gewicht ("Gewicht (kg): ") van iemand bevraagd. Bereken vervolgens het BMI van deze persoon ("BMI = "). Rond af tot een geheel getal. De berekening hiervoor is .
4. Maak een programma dat aan de gebruiker een aantal pizza's vraag die hij wenst te bestellen ("Aantal pizza's: "). Een pizza kost € 14,50 euro, en je hebt een speciale deal: 4 + 1 gratis! Laat de gebruiker het aantal pizza's ingeven en bereken de totale prijs.
5. Toon van een getal enkel het kommagedeelte. Bv. bij 14,245 toon je 0,245.

# List

Een variabele kan ook een *list* zijn. Een list is een datatype dat een lijst van verschillende waarden uitmaakt en wordt als volgt opgesteld:

1. nummers = []

Let op: lijsten zijn **zero-based**,   
dat wil zeggen dat ze vanaf 0 beginnen te tellen.   
nummers[0] wijst dus naar de eerste plaats in de lijst.

Enkele interessante functie om te beginnen met *List* zijn:

* *append(element)*: voegt een nieuw element toe aan het einde van de lijst.
* *insert(positie,element)*: voegt een nieuw element toe op een specifieke positie. Ook hier verwijst de positie 0 naar de voorste (eerste) positie in de lijst.
* *remove(element):* verwijdert een bepaalde waarde uit de lijst. Let op: *Remove* gaat de lijst van voor naar achter af en verwijdert het eerste element dat hij tegen komt. Komt het meegegeven element meerdere keren voor, dan wordt het slecht een eerste keer verwijderd.
* *del naam\_lijst[positie]*: verwijdert een bepaalde waarde op een bepaalde plaats uit de lijst.

Hieronder vind je een voorbeeld van een lijst met getallen. Er worden 3 getallen (nl. 1, 2, 3) toegevoegd en vervolgens het eerste getal (1) getoond op het scherm. Dan wordt eerst getal 1 verwijderd het tweede getal (3) verwijderd.

1. nummers = []
2. nummers.append(1)
3. nummers.append(2)
4. nummers.append(3)
5. print(nummers[0])
6. nummers.remove(1)
7. del nummers[1]

## ForEach

Om een collectie te overlopen, kan je een for each loop gebruiken. Deze gaat een bepaald stuk code voor elk element in de collectie uitvoeren. Zie onderstaand voorbeeld waarbij voor elk getal in de collectie deze op het scherm wordt getoond. De naam voor de variabele bij het overlopen van de variabelen (hier: nummer) mag je vrij kiezen.

1. nummers = [1,2,3]
2. for nummer in nummers:
3. print(nummer)

## For

Een *for-lus* wordt gebruikt om een bepaalde *int* (hieronder bv. *index*) vanaf een bepaalde startwaarde naar een eindwaarde te laten gaan, met een bepaalde verhoging/verlaging per keer. Hiervoor wordt de functie *range* gebruikt. Deze geeft een lijst van getallen terug die je dan over uit 3 delen:

1. Je stelt een variabele in met een startwaarde
2. Je bepaalt tot wanneer de herhaling wordt uitgevoerd (excl.)
3. Je verhoogt/verlaagt de variabele met een bepaalde waarde

In het onderstaande voorbeeld, start de variabele *i*  met de waarde 1, blijft de herhaling duren zolang *i* kleiner is dan 5 en verhoogt *i* per herhaling met *1*.

1. for i in range(1,5,1):
2. print(i)

In het onderstaande voorbeeld, start de variabele *i*  met de waarde 5, blijft de herhaling duren zolang *i* groter is dan 1 en verlaagt *i* per herhaling met *1*.

1. for i in range(5,1,-1):
2. print(i)

De start en eindpositie kan je dus vrij kiezen, al moet je natuurlijk ervoor zorgen dat de eindpositie ooit kan worden bereikt. In het onderstaande scenario verhoogt de variabele steeds meer, terwijl de eindpositie lager ligt (0) dan de startpositie (3).

1. for i in range(3,0,1):
2. print(i)

Een *for-lus* wordt vaak gebruikt in combinatie met de lengte van een array/list. Het nadeel van *foreach* is dat je, zolang je de *array/list* overloopt, moeilijk wijzigingen hierop kan uitvoeren.

Neem het onderstaande voorbeeld:

1. nummers = [1,2,3]
2. for nummer in nummers:
3. nummers.remove(nummer)
4. print(nummers)
5. print("Alles verwijderd")
6. print(nummers)

Hier gaat het getal 2 'onverwachts in blijven staan'. Dit komt omdat er alvorens er wordt overlopen een iterator wordt opgesteld, die wijzigingen aan de lijst niet actief meeneemt.

Beter is dus:

1. nummers = [1,2,3]
2. for index in range(len(nummers)):
3. del nummers[index]
4. print("Alles verwijderd")
5. print(nummers)

## Oefeningen

1. Maak een lijst en steek hier 5 getallen in. Overloop deze lijst en toon de getallen op het scherm.
2. Maak een lijst en steek hier 3 getallen in. Overloop deze lijst en tel alle getallen bij elkaar op.
3. Maak een lijst en steek hier 3 namen in van huisdieren ("Geef een naam van jouw huisdier op: "). Overloop deze lijst en toon deze strings op het scherm.
4. Maak een lijst van getallen. Laat automatisch hier alle getallen van 278 t.e.m. 300 in steken. Overloop vervolgens deze lijst en tel deze getallen bij elkaar op.
5. Laat de gebruiker een lijst van 3 ingrediënten opgeven. Overloop vervolgens deze lijst en plaats hier overal het nummer van de lijst voor. Bijvoorbeeld: 1. Eieren 2. Melk 3. Bloem
6. Laat de gebruiker een getal ingeven waarvan hij de tafel van 10 wil hebben ("Geef een getal in: "). Steek vervolgens in een lijst alle vermenigvuldigingen van 1 t.e.m. 10 in. Toon deze getallen op het scherm en vervang vervolgens deze getallen door hun kwadraat. Bijvoorbeeld: bij het ingeven van het getal 6, vul je eerst de lijst op met 6, 12, 18, … Vervolgens toon je 6, 12, 18, … op het scherm en vervang je de getallen in deze lijst met de getallen 36, 144, 324, …
7. Vraag de gebruiker hoeveel titels van films hij wil ingeven ("Aantal films: "). Vervolgens vraag je het aantal keer dat werd opgegeven naar een filmtitel, van slechtste naar beste ("Titel film (van slecht naar goed): "). Vervolgens toon je de films van goed naar slecht, met ervoor een nummer (1, 2, 3, …).

Bijvoorbeeld: Films: "Super slechte film", "Oké film", "Best film ooit".

Uitvoer:   
1. Beste film ooit   
2. Oké film  
3. Super slechte film

1. Laat een gebruiker 20 nummers ingeven. Bepaal vervolgens een boxplot, door de volgende data te tonen:

cijfers = [4, 5, 5.5, 6, 6, 6.5, 7, 7, 7.5, 8, 8, 9, 9.5, 10]

4|--|6--7--8|--|10

# Bijlagen

## Bijlage 1 – titel

# Bibliografie

**Ongeldige bron opgegeven.**