Basic Programming

# Introductie Python

## Achtergrond

* Guido van Rossum (NL)
* Gebaseerd op ABC
* Free software
* Genoemd naar tv-serie

## Key features

* Easy to learn
* Easy to read
* Easy to maintain
* Open-source
* Grote standard library: cross-platform
* Portable: werkt op alle hardware platformen
* Uitbreidbaarheid: eenvoudig modules toevoegen

## Python vs andere

In python:

* Geen ‘edit-compile-run’ ontwikkelingscyclus zoals in andere talen
* “rapid application development language”
* Prototyping language

## Github Copilot

* 2021 lancering
* Eigendom van Microsoft
* Gratis voor studenten en docenten
* AI-tool dat in staat is codevoorstellen voor je te genereren

# Python – basis syntax

## Python identifier

DEFINITIE: Een naam om een variabele, methode, klasse, module, … aan te spreken

* De naam:
* Start met letter of underscore
* Daarna 0 of meerdere letters, cijfers, underscore
* Opgelet: hoofdlettergevoelig
* Afspraak: start met kleine letter bv.: klasgroep\_1

**Gereserveerde namen:**

Table

Description automatically generated

## Python kenmerken

* Geen teken achter code lijn
* Commentaar: #
* Inspringen van meerdere lijnen: samenhorigheid
* Meerdere statements op 1 lijn

## Python variabelen

DEFINITIE: Een variabele is een plaats in het geheugen om een bepaalde waarde in te bewaren.

* Declaratie van een variabele: plaats in code waar de variabele eerste maal vermeld en dus aangemaakt wordt.

Bv.: mijn\_naam = “Arno”

A picture containing timeline

Description automatically generated

### Datatype

DEFINITIE: Een datatype is representatief voor een groep van mogelijke waarden waarbij telkens ook voor-gedefinieerde bewerkingen mogelijk zijn.

Diagram

Description automatically generated

* Datatype bepaald wat we met een variabele kunnen doen
* Omschakelen van datatype = converteren
  + Int()
  + Float()
  + Complex()
  + Str()
  + Hex()
  + Oct()

### Printen van variabelen

* Gebruik van formatted String literals
  + Bv.:

aantal = 12

opleiding = “MCT”

score = 18.4

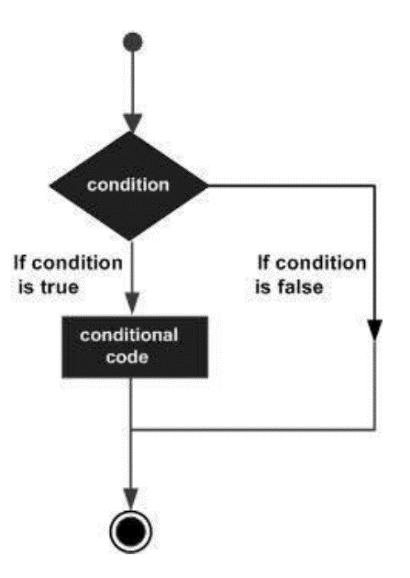
print(f”{aantal} studenten in {opleiding} haalden de score van {score:.2f} op 20”)

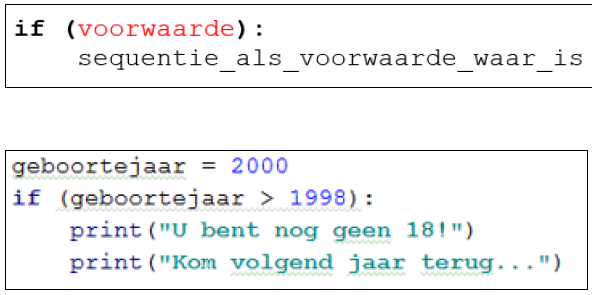
# Selectiestructuren

DEFINITIE: Een sequentie is een opéénvolging van lijnen code die 1 voor 1 uitgevoerd worden.

* Als niet alles automatisch uitvoeren
  + Voorwaarde / conditie toevoegen
  + Wordt pas uitgevoerd nadat voorwaarde gecontroleerd werd

## Enkelvoudige selectie

* Controle op voorwaarde:
  + Indien waar: sequentie *binnenin* wordt uitgevoerd
  + Indien niet waar: sequentie *binnenin* wordt niet uitgevoerd



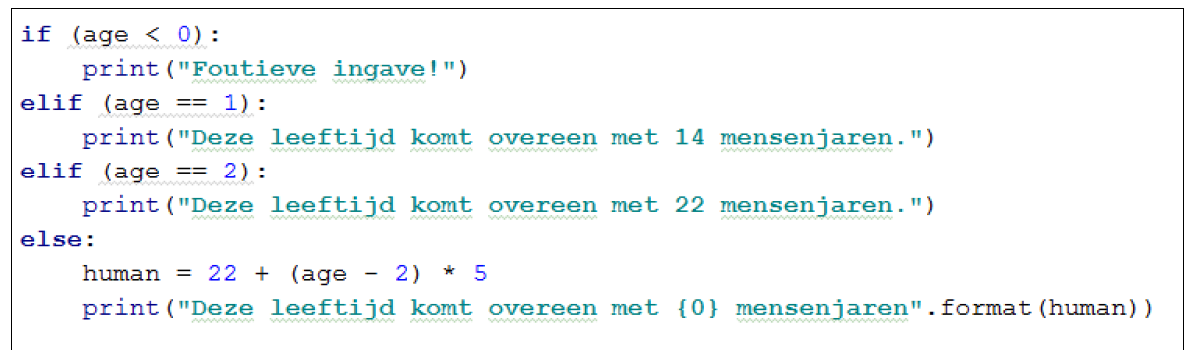
## Tweevoudige selectie

* Controle op voorwaarde
  + Indien waar: sequentie deel 1 uitvoeren
  + Indien onwaar: sequentie deel 2 uitvoeren

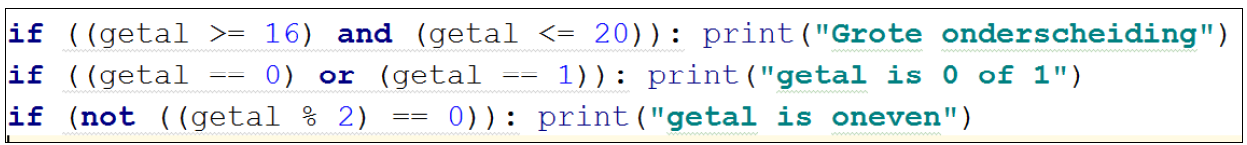
## Meervoudige selectie

* Meerdere voorwaarden in specifieke volgorde na

Elkaar

* Wanneer een voorwaarde waar is:
  + Bijhorende sequentie uitvoeren
  + Resterende voorwaarden worden niet meer gecontroleerd

## Bij meerdere voorwaarden

* Gebruik logische operatoren:
  + And
  + Or
  + Not

# Naming conventions

* Doel:
  + Consistente naamgeving van variabelen, functies, methodes, klassenamen
  + Leesbare code

## Variabelenamen

* Starten steeds met een leter; gevolgd door 0 of meerdere tekens
* Indien meerdere woorden
* SnakeCase:
  + hoogte\_kast

## Functienaam

* voorkeur
  + kleine letters met underscores tussen de verschillende woorden

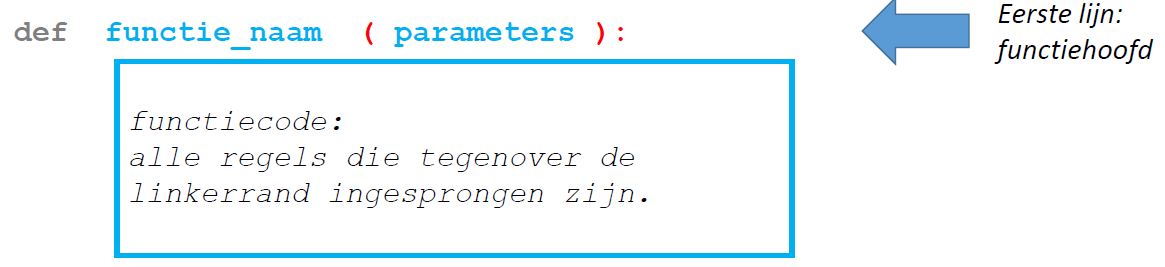
# Functies

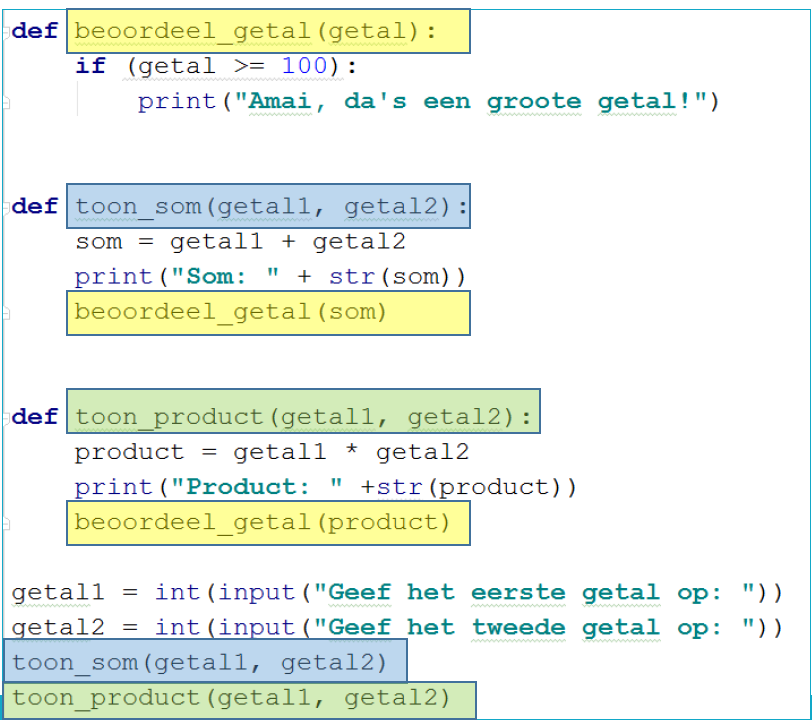
## Doel functies



* Onoverzichtelijk
* Moeilijk leesbaar
* Herhalende code
* Niet bruikbaar
* overzichtelijke code / verhoog leesbaarheid
  + voor jezelf
  + voor teamleden
* Herbruikbare code stimuleren
* Herhaling vermijden
* Top down structure: opsplitsen in kleinere, meer beheersbare stukken
* Stukken code verbergen / beschermen

## Syntax function





## Naamgeving functies

* Respecteer afspraken:
  + Start met kleine letter
  + Tussen elk woord “\_”
  + Kies logische naam

## Parameters van een functie

* Via de “parameters” worden waarden of “argumenten” aan een methode doorgegeven

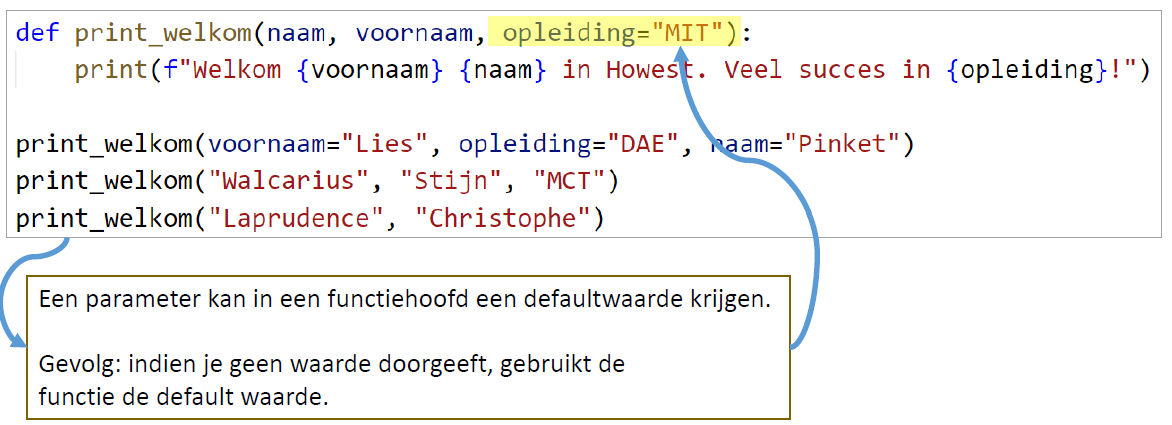
### Doel parameters

* Autonome functies
  + Krijgt alles netjes binnen
  + Geen extra info nodig
  + Trekt zich niet waar info vandaan komt
  + Kan zo dezelfde code uitvoeren met telkens andere waardes
* Verhoogt herbruikbaarheid
* Globale variabelen vermijden

### Gebruik parameters

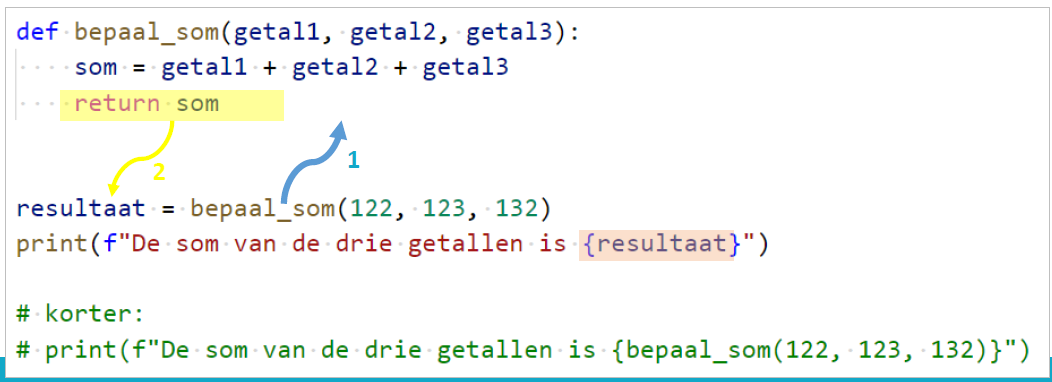






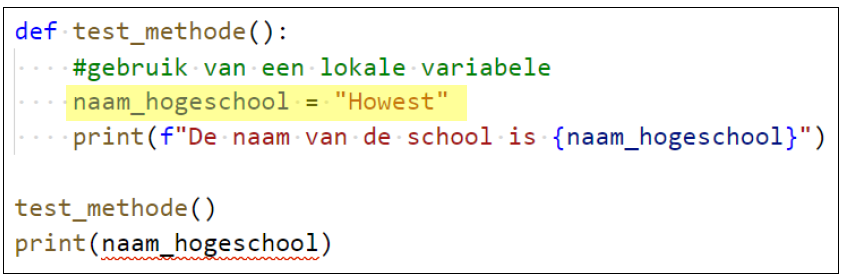
### Return commando

* Functies kunnen iets teruggeven

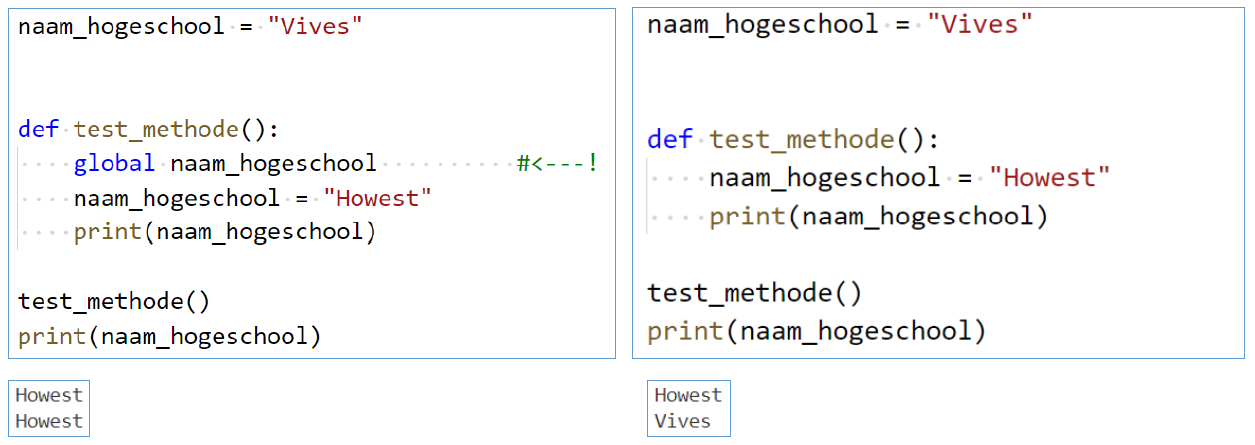


## Scope van variabelen

* Lokale variabele:
  + Variabele wordt binnen de functie aangemaakt
  + Dus enkel in de functie gekend



* Globale variabele:
  + Buiten functie aangemaakt
  + Overal aanspreekbaar (en wijzigbaar)

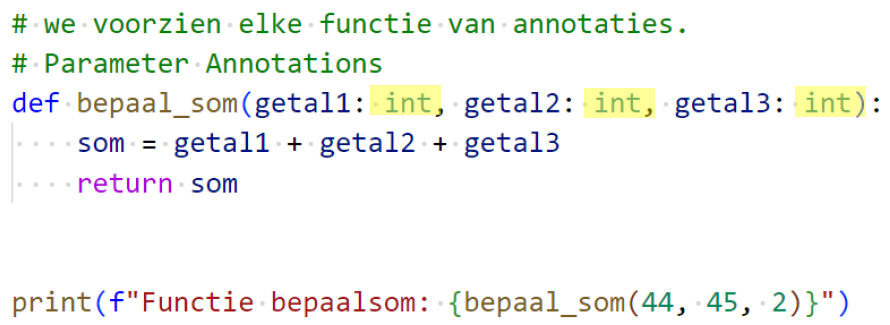


## Annotaties in functies

* ‘extra aantekening’
* Doel:
  + Leesbaarheid
  + Linter controleert mee

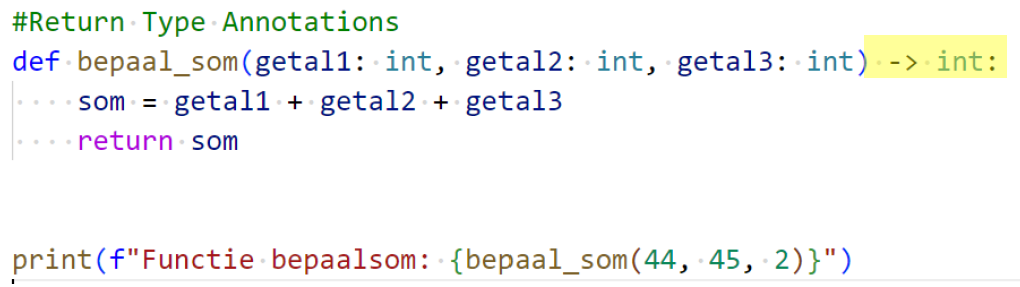
### Parameter annotations

* Datatype vermelden bij parameters



### Return type annotations

* Achteraan functiehoofd vermelden wat functie teruggeeft



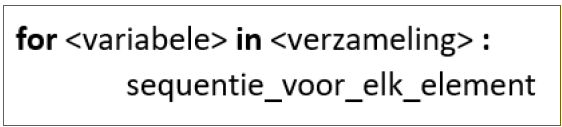
## Autonome functies

* Streef naar “autonome” functies:
  + Maakt niet uit waar data vandaan komt
    - Functie vertelt enkel wat hij nodig heeft
    - Via parameters
  + Maakt niet uit waar resultaat naartoe moet
    - Geeft gewoon resultaat terug
    - Oproeper bepaalt wat ermee gebeurt
  + Gebruikt geen globale variabelen
* Herbruikbaarheid

# Lusstructuren

## For-lus

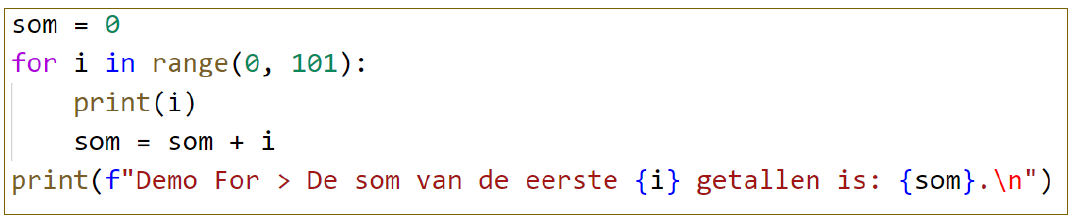
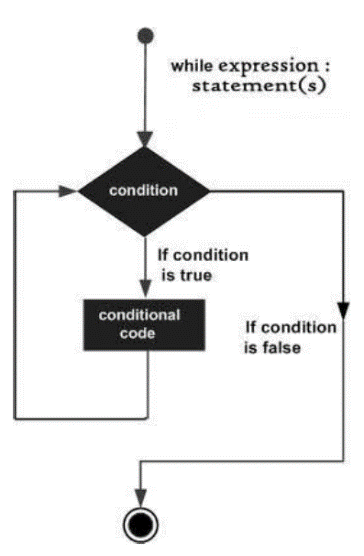
* Elementen één voor één overlopen
* Exacte aantal gekend



* Range-commando laat toe een

verzameling te definiëren

* + Range(n)
  + Range(min,max)
  + Range(min,max,step)



## While-lus

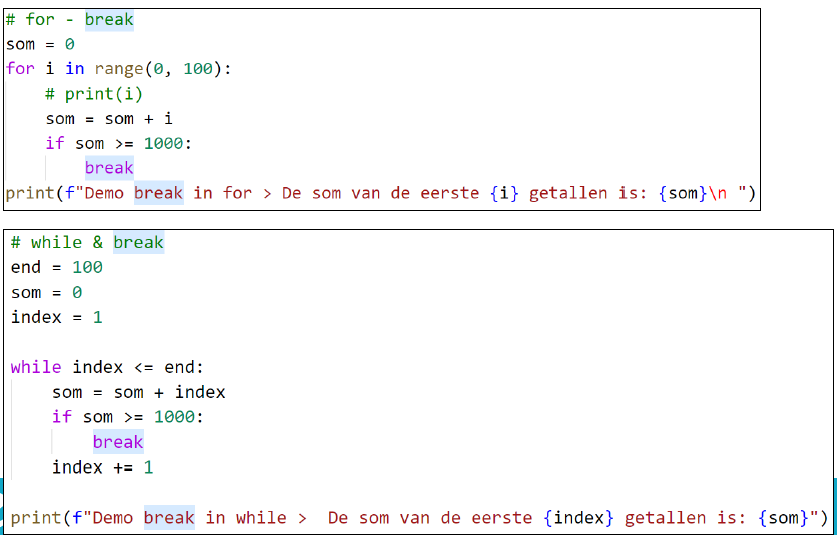
* Herhaling hangt af van voorwaarde
* Controle op voorwaarde:
  + Indien waar: sequentie binnenin wordt herhaald
  + Indien niet waar: herhaling wordt stopgezet

Opmerking:

* Voorwaarde minimum 1 maal gecontroleerd
* Indien geen voorwaarde: onmiddellijk false
* Gevaar voor infinite loop

## Break

* Met break kan je lus vroegtijdig verlaten



# String & string functies

## Datatype string

DEFINITIE: Een string is een datatype waarvan een variabele 0 of meerdere karakters kan bijhouden

Bv.: string\_1 = “Hello, how are you”

string\_2 = “1+1”

string\_3 = “00#”{ff@”

* In python is een string een geordende rij van karakters waarbij elk karakter een specifieke positie heeft
* Text

  Description automatically generatedOpvragen van een karakter kan via de index
  + Vooraan startend vanaf positie 0
  + Achteraan startend vanaf positie -1

Table

Description automatically generated with low confidence

## String operatoren

* String concatenation: via +

Text

Description automatically generated resultaat: Howest rules

* String repetition: via \*

Shape

Description automatically generated with medium confidence resultaat: HowestHowestHowestHowest

* Lengte: via len()

A screenshot of a computer

Description automatically generated with medium confidence resultaat: 6

* Indexing: via []

A screenshot of a computer

Description automatically generated with medium confidence resultaat: H

* Slicing: via [:]

A picture containing diagram

Description automatically generated

**Enkele stringfuncties:**

|  |  |  |
| --- | --- | --- |
| Stringfunctie | Beschrijving | Voorbeeld |
| Find(substring, beg=0, end=len(string)) | Gaat de positie na waar de substring in de string voorkomt; eventueel wordt extra zoekbereik opgegeven (default vanaf start tot einde string). Indien niet aanwezig wordt “-1” teruggegeven/ | str = “this is string example….wow!!!”  print(str.find(‘is))  print(str.find(‘is’,10))  resultaat:  2  -1 |
| Index(substring, beg=0, end=len(string)) | Idem als find, maar indien substring niet voorkomt: foutmelding | str = “this is string example….wow!!!”  print(str.find(‘is))  print(str.find(‘is’,10))  resultaat:  2  error |
| isalnum() | Geeft waar terug als string min 1 karakter bevat EN enkel bestaat uit cijfers/letters | str = “this is string example….wow!!!”  print(str.isalnum())  resultaat:  False |

# Debugging

## Compileerfouten (syntax errors)

* Wordt rood onderlijnd
* Mogelijke oorzaken
  + Onbekende variabele
  + Methode bestaad niet, te weinig parameters
  + …

## Uitvoeringsfouten (runtime errors)

* Syntax code is correct
* Tijdens uitvoering crasht applicatie
  + Regel wordt aangeduid
  + Aangepaste foutmelding

## Logische fouten

* Compiler geeft géén fouten = code compileert
* Programma crasht niet, werkt gewoon verder
* De uitkomst / resultaat / werking is niet zoals verwacht

## Oplossingen

### Syntaxfouten

* Pylint
  + Geeft warnings en errors

### Uitvoeringsfouten

* Leer type fouten herkennen
* Lees boodschap aandachtig

### Logische fouten

* Plaats breakpoint + debug
* Doorloop regel per regel

# Datastructuren

DEFINITIE: Een datastructuur (in programeercontext) is een manier waarop elementen (= ‘componenten’ of ‘items’) aan elkaar hangen en met 1 variabele in groep aanspreekbaar zijn.

Vb.: ‘studenten\_1MCT’, ‘weekdagen’, ‘rijksregisternummers’, …

* Het soort datastructuur bepaalt hoe de elementen kunnen opgevraagd, toegevoegd of gewijzigd worden.

## Sequentie

* Meest eenvoudige datastructuur
* Elk element heeft een positie (index)
* Eerste element start op positie 0
* String is ook een sequentie

### Sequentie ‘List’

* List is het veelzijdige datatype binnen python
* Elementen binnen een listen hoeven niet hetzelfde datatype te zijn

Graphical user interface, text, application

Description automatically generated

* Elementen uit list opvragen
  + 1 element opvragen: index
    - print(list1[3])
    - 2000
  + Meerdere elementen: slicing
    - print(list2[0:2])
    - [1, 2]
* Elementen aan list toevoegen
  + Append
    - list2.append(6)
    - [1, 2, 3, 4, 5, 6]
  + Insert
    - list2.insert(3, 55)
    - [1, 2, 3, 55, 4, 5]
* Elementen uit list updaten

A picture containing chart

Description automatically generated

* Elementen uit list verwijderen
  + del()
    - del(list2[3])
    - [1, 2, 3, 5]

#### Basic list operatoren

Table

Description automatically generated

#### List overlopen met lusstructuur

##### For-lus

Text

Description automatically generated

##### While-lus

Text

Description automatically generated

##### Handige ingebouwde list-methodes

|  |  |
| --- | --- |
| list1.append(element) | Element achteraan toevoegen |
| list1.count(element) | Telt hoeveel keer element 1 voorkomt |
| list1.extend(sequentie) | Voegt andere sequentie (bv list2) toe |
| list1.index(element) | Geeft positie van 1st voorkomende element |
| list1.insert(index, element) | Voegt element toe op positie index |
| list1.pop() | Verwijdert laatste element |
| list1.remove(element) | Verwijdert opgegeven element |
| list1.reverse() | Keert de list om |
| list1.sort() | Sorteert de list |

* Lists kunnen zowel parameters als returnwaardes zijn voor functies

### Sequentie ‘Tuple’

* Hetzelfde als een lsit maar kan niet meer gewijzigd worden
* Voordelen
  + Sneller
  + Veilig in gebruik
  + Ideaal voor het vastleggen van een beperkt aantal keuzemogelijkheden voor de gebruiker

#### Basic tuble operatoren

Table

Description automatically generated

#### Standaard bewerkingen voor elke sequentie: dus ook voor tuple

Text

Description automatically generated

* Tuple overlopen = idem list
* Kan parameter voor functie zijn
* Kan element van een list zijn EXAMENVRAAG

Text

Description automatically generated with low confidence

## Dictionary

DEFINITIE: ongeordende verzameling van elementen waarbij elk element bestaat uit een key en een value. Binnen de dictionary is elke key uniek.

* helemaal verschillend dan een list of tuple: de elementen hebben geen positie meer, maar wel een unieke sleutel waarmee ze geïdentificeerd kunnen worden
* syntax:
  + mijn\_dict = {“a”: “alpha”, “o”: “omega”, “g”: “gamma”}

### Opvragen van elementen

* get() functie
  + parameters: 1. Naam van de key

2. optioneel: default-waarde indien key niet gevonden wordt

Graphical user interface, text

Description automatically generated with medium confidence

* via [naam\_key]µ

Text

Description automatically generated

Chart

Description automatically generated with low confidence

### Toevoegen van nieuwe elementen

* <naam\_dictionary>[“key”] = “value”
* Indien key nog niet bestaat
  + Key wordt met value toegevoegd
* Indien key reeds bestaat
  + Value voor key wordt aangepast

### Verwijderen van een element

* del(<naam\_dict>[“key”])

### controle aanwezigheid

Graphical user interface, application, Word

Description automatically generated

### Overlopen van elementen

* for-lus

Calendar

Description automatically generated

### Spelen met dictionaries

* Dictionary kan parameter van een methode/functie zijn
* Parameter annotaties bij dictionary: Dict[str, int] (=vb)
* Een matrix kun je op 2 manieren bijhouden:
  + List waarvan de elementen opnieuw lists zijn
  + Een dictionary waarvan de keys tuples zijn (rij en kolomnr) en de value het bijhorende getal uit de matrix

Text

Description automatically generated

### Datastructuur ‘Set’

DEFINITIE: Een set is een ongeordende verzameling die geen duplicaten toelaat.

* Syntax: <naam\_set> = {‘apple’, ‘orange’, ‘apple’, ‘pear’, ‘banana’, ‘pear’}

Print(naam\_set)

{‘pear’, ‘banana’, ‘apple, ‘orange’}