Python

Python lessen voor tweedejaars AO leerlingen: An introduction to Python 3.

- Lesplan
- Les 1: Python - Introductie, installatie & operatoren
- Week 5: Opgaven Python functions and lists
- LEGENDA
- Toetsvoorbereiding week 5
- Uitwerkingen opgaven ter toetsvoorbereiding (week 4)

Lesplan

Introductie Python les

In de Python lessen wordt kennis opgedaan over de basis van Python 3.

Python is een object-georiënteerd programmeertaal en wordt geïntroduceerd in de eerste les van week 1. Tijdens deze les wordt gedemonstreerd hoe Python geïnstalleerd kan worden op Windows en hoe de Python shell gebruikt kan worden.
Van de leerling wordt verwacht dat hij/zij (inter)actief meedoet in de les en het huiswerk van de voorgaande week af - en ingeleverd heeft bij de docent.

Het ingeleverde huiswerk wordt aan het begin van de les behandeld en de leerlingen krijgen de ruimte om aan te geven waar ze tegenaan zijn gelopen en wat ze lastig vonden. Voor het oplossen van het probleem wordt de leerling gevraagd wat hij/zij zelf al heeft gevonden met betrekking tot het probleem en wat een mogelijke oplossing ervoor zou kunnen zijn.

Na het bespreken van het huiswerk wordt een nieuw stukje theorie behandeld, en krijgen de leerlingen de gelegenheid om aan de les-opdrachten **[4]** en huiswerk te werken.

Een overzicht van de lesstof is te vinden in de studiewijzer hieronder.
Voor meer informatie kun je de individuele lessen raadplegen.

> Het kan zijn dat er een begrippenlijst is toegevoegd aan het einde van de pagina. Deze lijst bevat begrippen die je kunnen helpen bij het begrijpen van de lesstof.
>
> Zie je toch een onbekend begrip? Geef het aan bij de docent!

> **LET OP!!** Neem de LEGENDA door om de pagina's optimaal te gebruiken.

Studiewijzer

Week	Doel	Beschrijving

1	De leerling is in staat Python 3.8 te installeren, de Python shell op te starten en is bekend met de Python operatoren.	• Instaptoets **[1], [2]**
		• Introductie Python 3.8 **[3]**
		• Installeren Python shell
		• Python operatoren:
		  ◦ Vergelijkende operatoren
		  ◦ Rekenkundige operatoren
		  ◦ Logische operatoren

2	De leerling kan verschillende datatypes herkennen en is in staat om deze datatypes op te slaan in variabelen en te gebruiken.	• Bespreken huiswerk week 1 **[4]**
		• Datatypes:
		  ◦ Int
		  ◦ boolean
		  ◦ strings (escape filepaths)
		  ◦ float
		• Variabelen

3	De leerling kent basic python build-in functies en kan deze gebruiken in if-statements (vergelijkingen).	• Bespreken huiswerk week 2 • Build-in functions als print(), len(), insert(), range() • If-statements & ternary conditional operator
4	Toetsing opgedane kennis van voorgaande weken. Bespreken van de toets en vragen beantwoorden.	• Bespreken huiswerk week 3 • Eerste toets over de kennis die is opgedaan in de afgelopen weken. • Bespreken van de toets • Vragen beantwoorden

5	De leerling kent datastructuren en kan deze gebruiken.	• Bespreken van de toets • Bespreken van datastructuren: arrays & dictionaries, tuples (immutable)
6	De leerling kan loops schrijven en is in staat de kennis van de voorgaande weken toe te passen in een loop.	• Bespreken huiswerk week 5 • Introductie in loops
7	De leerling kan functies (methods) schrijven en kent de begrippen parameter en argument.	• Bespreken huiswerk week 6 • Introductie tot functies programmeren • Functies met parameters • Samenvatting van voorgaande weken
8	Eindtoets Python	• Bespreken huiswerk week 7 • Eindtoets Python

Afhankelijk van het tempo wordt/worden (er) extra-curriculaire theorie/opdrachten toegevoegd aan de Python lessen.

Bronnen

[1] Instaptoets: https://b.socrative.com/login/student/
Roomname: ROCVAAO

[2] De instaptoets telt niet mee voor het eindcijfer.

[3] Installeren van Python 3: https://www.python.org/downloads/

[3] Eventuele vragen/problemen worden ook besproken tijdens het behandelen van het huiswerk.

[4] Les-opdrachten zijn anders dan het huiswerk en worden tijdens de les gemaakt. Van de leerling wordt verwacht dat hij/zij
vóór iedere les zowel de les-opdrachten als het huiswerk heeft gemaakt en ingeleverd bij de docent.

Les 1: Python - Introductie, installatie & operatoren

Python

Introductie - Wat is Python?

Wat is Python?

Python is een object-georiënteerde programmeertaal die begin jaren 90 ontworpen en ontwikkeld werd door Guido van Rossum.
Hoewel Python tegenwoordig veel wordt gebruikt als back-end programmeertaal voor de Data Science, kan het ook gebruikt worden als front-end taal.

Iedereen denkt dat Python alleen gebruikt kan worden voor webdevelopment. Toch is het ook mogelijk om Python te gebruiken bij het ontwikkelen van desktop applicaties en spelletjes.

Fun fact: Python heeft zijn naam te danken aan het favoriete televisieprogramma van Guido van Rossum, namelijk: Monty Python's Flying Circus **[2].**

Installeer Python 3

Tijdens de lessen gebruiken wij Windows. Uiteraard zijn andere Operating Systems toegestaan, maar het kan zijn dat de docent niet (altijd) kan helpen met het troubleshooten van problemen die zich voordoen.

Hieronder staat hoe je Python 3 kunt installeren op verschillende Operating Systems.

Windows:

Stap	Toelichting
Download Python 3	Ga naar https://www.python.org/downloads/ en selecteer Python-3.8.1. Er wordt een download gestart, klik op het bestand om Python 3 te installeren.

Ubuntu:
Open de Konsole **[3]**, deze kan via de Software Explorer of via de keyboard shortcut: `CTRL + ALT + T` .

Stap	Toelichting
Installeer Python 3	Volg de stappen op de website https://linuxize.com/post/how-to-install-python-3-7-on-ubuntu-18-04/ om Python 3 te installeren.

Mac OS:

Stap	Toelichting
Installeer Python 3	Volg de stappen op de website https://installpython3.com/mac/ om Python 3 te installeren.

Operatoren

Net als andere programmeertalen, maakt ook Python gebruik van operatoren (operators).
In python kunnen we deze operatoren in drie categorieën indelen, namelijk:

- Vergelijkende operatoren
- Rekenkundige operatoren
- Logische operatoren

Alle bovenstaande categorieën zullen we hieronder behandelen.

Vergelijkende operatoren

Vergelijkende operatoren worden gebruikt om waardes met elkaar te vergelijken en returnen een True/False. Deze waardes zijn boolean **[B3]** waardes. Het onderstaande tabel - Tabel 1 - beschrijft vergelijkende operatoren in Python:

Operator	Beschrijving
==	Gelijk aan
!=	Niet gelijk aan
<	Kleiner dan
<=	Kleiner dan of gelijk aan
>	Groter dan
>=	Groter dan of gelijk aan

Tabel 1: Overzicht van vergelijkende operatoren

Rekenkundige operatoren

De rekenkundige operatoren kunnen gebruikt worden voor berekeningen. Voorbeelden van deze berekeningen zijn bijvoorbeeld optellen, aftrekken, vermenigvuldigen, delen, rest berekeningen en exponenten.

Tabel 1 toont verschillende rekenkundige operatoren die je kunt gebruiken wanneer je programmeert in Python. Er zijn twee getallen als voorbeeld genomen om de syntax **[B1]** van een operator uit te leggen, de getallen 4 en 2.

Operator	Beschrijving	Syntax
+	Plus: telt twee waarden bij elkaar op.	4 + 2
-	Min: trekt twee waarden van elkaar af.	4 - 2
*	Vermenigvuldig: telt twee waarden herhaaldelijk bij elkaar op.	4 * 2
/	Delen: trekt twee waarden herhaaldelijk van elkaar af.	4 / 2
%	Modulo: berekend de restwaarde van een deelsom.	4 % 2
//	Delen met afronden naar beneden.	4 // 2
**	Exponent: herhaaldelijk vermenigvuldigen van twee waarden.	4 ** 2

Tabel 2: Overzicht van rekenkundige operatoren

Lesopdracht: vergelijkende - en rekenkundige operatoren:

Lesopdracht 1: Wat is de uitkomst van `5 % 2` ?

Lesopdracht 2: Wat is de uitkomst van `2 + 4 > 10 % 2` ?

Lesopdracht 3: Wat is de betekenis van het woord "exponent" en hoe kun je rekenen met exponenten in Python?

Lesopdracht 4: Klopt deze vergelijking `"1" == 1` ? Leg uit waarom wel/niet.

Lesopdracht 5: Een "product" is een wiskundige term **[B2]**. Leg uit hoe je een product kunt uitrekenen in Python en laat zien hoe je het product van 2 en 8 hebt uitgerekend in je Python shell.

Lesopdracht 6: We hebben de volgende vergelijking `4 + 25 * 2 = 58`. Klopt deze berekening? Leg uit waarom wel/niet.

Lesopdracht 7: Wat is de uitkomst van `5 * 5 * 5` en hoe kun je dit anders berekenen in Python?
Tip: `5 * 5 * 5 = 5 ^ 3`.

Lesopdracht 8: Wat is de uitkomst van `16 + 4 * 8` ? Leg uit welke volgorde wordt aangehouden tijdens deze berekening.

Lesopdracht 9: De volgorde waarin rekenkundige operatoren worden toegepast kan beïnvloed worden door haakjes. Hoe zorg je er in je Python shell voor dat de uitkomst van `16 + 4 * 8` gelijk is aan 160?

Logische operatoren

In Python zijn er 3 logische operatoren. Deze zijn:

- and (en)
- or (of)
- not (niet)

De bovenstaande - logische operatoren - worden gebruikt voor vergelijken van boolean waarden. Afhankelijk van de operator geeft deze een True/False terug. Hieronder zijn drie tabellen die de logische operatoren uitwerken.

Logische operator - AND

De uitkomst van een vergelijking met de logische operator "and" geeft alleen True als beide kanten waar zijn. Voor alle andere vergelijkingen returned het False:

Vergelijking	Uitkomst
True *and* True	True

True *and* False	False
False *and* True	False
False *and* False	False

Tabel 3: Vergelijking en uitkomst van logische operator "and".

Logische operator - OR

De uitkomst van een vergelijking met de logische operator "or" geeft altijd True als minstens een kant van de vergelijking waar is:

Vergelijking	Uitkomst
True *or* True	True
True *or* False	True
False *or* True	True
False *or* False	False

Tabel 3: Vergelijking en uitkomst van logische operator "or".

Logische operator - NOT

De logische operator "not" returned het tegenovergestelde. Dit betekend dat not True altijd False wordt en visa versa:

Vergelijking	Uitkomst
not True	False
not False	True

Tabel 4: Vergelijking en uitkomst van logische operator "not"

Lesopdrachten: logische operatoren

Lesopdracht 10: Wat is de uitkomst van `10 < 4 and 5 > 3`? Licht je antwoord toe.

Lesopdracht 11: Wat is de uitkomst van `5 < 4 or 4 < 2`? Licht je antwoord toe.

Lesopdrachten en huiswerk

> De lesopdrachten en het huiswerk kun je vinden op Microsoft Teams: Team "Python-OITAOO8A".
> De deadline voor het huiswerk staat in de additionele informatie van het genoemde team.

Begrippen

[B1] Syntax = De manier waarop iets, bijvoorbeeld een operator, gebruikt/geschreven wordt.
[B2] Term = Woord; Benaming.
[B2] Boolean = True of False.

Bronnen

[1] Bronvermelding: https://docs.python.org/2/faq/general.html#why-is-it-called-python
[2] De Ubuntu terminal wordt "Konsole" genoemd.

Week 5: Opgaven Python functions and lists

1. Schrijf een functie die een list accepteert als argument. Deze functie moet checken of de list leeg is.

2. Schrijf een functie die een list accepteert als argument. Deze functie moet de items in een list bij elkaar optellen.

3. Schrijf een functie die een list accepteert als argument. Deze functie moet het product van de items printen.

4. Maak een variabele listie aan en sla hier een empty list in op. Maak vervolgens een functie. Deze functie moet een for-loop bevatten en deze for-loop moet gemaakt worden met een range(100). Tijdens iedere iteratie moet het getal aan listie toegevoegd worden.

5. Gebruik listie uit opgave 4 voor deze opdracht. Schrijf een functie die een list accepteert als argument. Bovenin de functie moet je ervoor zorgen dat de items in de list ge-shuffeld worden. Zorg dat je een nieuwe variabele maakt die de geshuffelde list bevat.

   Schrijf vervolgens een for-loop die de items in de list looped. Alle getallen die groter zijn dan 50 moeten uit de list verwijderd worden.

   Na de for-loop moet je je list printen (zonder getallen boven de 50).

LEGENDA

Deze pagina dient als een legenda bij het begrijpen van de lessen. Hieronder zie je een overzicht van mogelijkheden:

Belangrijke informatie:

> Belangrijke informatie kun je vinden in deze blauwe ballon.

Lesopdrachten:

> Lesopdrachten moeten uitgewerkt en ingeleverd worden bij de docent en kun je herkennen aan deze oranje ballon.

Inline code:
Wanneer er in een zin een stukje code wordt getoond, kun je het herkennen aan de inline code ballon: `"Hello" + "world" + 1` .

Code snippet:
Wanneer een code snippet **[B1]** wordt gebruikt, kun je deze herkennen aan de grijze wol met een getal langs de kantlijn. Bijvoorbeeld:

```
print('Hallo!')
```

Begrippen:
Het kan zijn dat er woorden worden gebruikt die mogelijk extra uitleg nodig hebben. Deze worden aangeduid met een rechte haar, gevolgd door een B en dan een nummer. Het nummer bepaalt de volgorde van begrippen. Een voorbeeld hiervan is hierboven te zien, naast het woord "code snippet". Dit is het eerste begrip, daarom zie je een 1 naar de B.
De uitleg van "code snippet" kun je op dezelfde pagina onder titel "Begrippen" terug vinden:

Begrippen

[B1] Code snippet = een stukje code (van een groter geheel).

Bronnen:

Bronnen kun je herkennen aan getallen tussen rechte haken (**[1]**). Deze verwijzen naar bronnen in de bronnenlijst. De bronnenlijst is te vinden aan het einde van de pagina, onder het kopje 'Bronnen'.

Bronnen

[1] www.google.com (dit is een slecht voorbeeld!)

Toetsvoorbereiding week 5

Wat je moet kunnen voor de Python toets

- De leerling weet wat operatoren zijn en kan gebruik maken van vergelijkende -, rekenkundige - en logische operatoren.
- De leerling is bekend met het begrip concatenation en kan met behulp van string concatenation woorden/zinnen aan elkaar toevoegen.
- De leerling weet wat een datatype is, kan enkele voorbeelden opnoemen en werken met (verschillende) datatypen.
- De leerling kan de ingebouwde type-functie toepassen om de datatype van bijvoorbeeld een variabele hiermee achterhalen.
- De leerling heeft kennis van variabelen en kan hiermee werken.
- De leerling weet wat een Python for-loop is, is bekend met de syntax en kan deze schrijven.
- De leerling weet wat een if-statement is, is bekend met de syntax en kan deze schrijven.
- De leerling weet wat een break - en continue statement is en kan deze toepassen in een for-loop.
- De leerling weet wat een datastructuur is, kan enkele voorbeelden noemen en kan werken met Python lists en dictionaries.
- De leerling is bekend met de volgende ingebouwde Python functies:
  - print()
  - len()
  - range()

De bovenstaande punten worden in de volgende secties toegelicht en uitgewerkt.

> **LET OP! Deze pagina bevat informatie voor de toets in week 5. Echter, van de leerling wordt ook verwacht dat hij/zij in staat is om bijvoorbeeld for-loops te schrijven die een list of dictionary loopen. Hierbij is het belangrijk om te weten hoe je 1) Loopt door een list en 2) Loopt door een dictionary.**
>
> **Van de leerling wordt verwacht dat hij zijn de key-value pairs, keys en values kan loopen wanneer het om een dictionary gaat.**

Het loopen van lists/dictionaries is in de les gedemonstreerd. Voorbeelden zijn ook te vinden in de uitwerkingen van de toetsstof van week 4.

Operatoren (operators)

de leerling weet wat operatoren zijn en kan gebruik maken van vergelijkende -, rekenkundige - en logische operatoren.

Vergelijkende operatoren

Vergelijkende operatoren resulteren altijd in een Boolean waarde.

Syntax	Beschrijving	Voorbeeld
<	Kleiner dan	5 < 7 = True
<=	Kleiner dan of gelijk aan	5 <= 5 = True
>	Groter dan	7 > 5 = True
>=	Groter dan of gelijk aan	7 >= 5 = True
==	Gelijk aan (gebruikt voor een vergelijking. Niet hetzelfde als =)	7 == 5 = False
!=	Niet gelijk aan	7 != 5 = True

Rekenkundige operatoren

Rekenkundige operatoren worden gebruikt voor calculaties met numerieke waarden.

Syntax	Beschrijving	Voorbeeld
+	Plus	4 + 2 = 6
-	Min	4 - 2 = 2
*	Keer (vermenigvuldiging)	4 * 2 = 8
/	Delen	4 / 2 = 2

%	Modulus (rest van een deelsom)	4 % 2 = 0 (geen rest, want 4 is een even getal) 7 % 2 = 1 (rest 1, want 7 is een oneven getal)
**	Exponent (herhaaldelijk vermenigvuldigen)	4 ** 2 = 16 (4 ^2 = 4 * 4 = 16)
//	Geeft een geheel getal na deling (afgerond naar beneden). Bijvoorbeeld 4.5 = 4.	7 // 2 = 3 (7/2 is 3.5, maar een dubbele forward-slash zorgt voor een afronding naar beneden)

Logische operatoren

Logische operatoren resulteren in een Boolean waarde.

Syntax	Beschrijving	Voorbeeld
or	Of	True or False = True
and	En	True and False = False
not	Niet	not True = False

De leerling is bekend met het begrip concatenation en kan met behulp van string concatenation woorden aan elkaar toevoegen.

String concatenation

Een datatype is een bepaalde type data, bijvoorbeeld tekst. In Python - en andere programmeertalen - wordt dit String genoemd.
Text zullen we voortaan beschrijven als een string, en een string wordt aangeduid door enkele - of dubbele aanhalingstekens. Data van het type string kun je samenvoegen. Over het algemeen gebruik je hier een plus teken voor, bijvoorbeeld:

```
>>> 'hello '+'world'
'hello world'
```

Dit samenvoegen wordt in het Engels *concatenation* genoemd. Je kunt dit proces dus beschrijven als het samenvoegen van twee or meer aparte stukken data van het type string.

Datatype

Zoals het woord *datatype* aangeeft, gaat het bij datatypes om het type gegeven. Voorbeelden van datatypen zijn:

Datatype	Python (syntax)
Teskt type	str
Numeriek type	int, float
Reekstype	list, range
Map type	dict
Boolean type	bool

Uiteraard zijn er meer datatypen. De bovenstaande datatypen hebben we in de les behandeld.

Let op! List en dict (dictionary) zijn datastructuren, maar behoren in Python ook tot datatypen omdat je o.a. met de type-functie kan achterhalen tot welke datatype class ze behoren.

De type-functie is een ingebouwde Python functie en wordt gebruikt om een datatype van een bepaald gegeven te achterhalen. Hieronder zie je enkele voorbeelden:

```
>>> type([])
<class 'list'>
>>> type('hello world')
<class 'str'>
>>> type(1)
<class 'int'>
>>> type(True)
<class 'bool'>
>>> type({})
<class 'dict'>
>>> type(.5)
<class 'float'>
```

Variabelen

Met behulp van variabelen kun je data uit het computergeheugen ophalen en wijzigen. Python is, in vergelijking tot bijvoorbeeld Java, er makkelijk met het aanmaken van variabelen. In Python hoef je bijvoorbeeld geen datatype aan te geven wanneer je een variabele declareert. Bijvoorbeeld:

```
x = "ik ben een variebele


y = 5
```

```
variebel_in_python = "In Python wordt ieder woord gescheiden met een underscore(_)"


mijn_mooie_list = []
```

For-loop

Een for-loop kun je het beste gebruiken wanneer je een bepaalde handeling herhaaldelijk wilt uitvoeren. De logica werkt als volgt:

- Voer de volgende handeling(en) uit voor ieder getal binnen een bepaald interval.

of:

- Voer de volgende handeling(en) uit voor ieder element in een list.

range()

Het volgende voorbeeld demonstreert de for-loop aan de hand van het interval 0 tot en met 9. Dit interval zetten we vast met behulp van de range functie, dit is een ingebouwde Python functie. De syntax voor de range functie is: `range()`. Omdat we over een functie spreken, is het noodzakelijk dat we de haakjes gebruiken om deze aan te roepen. Tussen de haakjes zetten we het getal, bijvoorbeeld 6: `range(6)`. Dit zal ervoor zorgen dat we straks loopen, startend bij 0 tot en met 6 - 1 = 5.

Syntax

De for-loop heeft altijd dezelfde opbouw. Het begrip met het woordje *for* gevolgd door één of meerdere variabelen. Vervolgens komt het woordje *in*, gevolgd door hetgeen waar we doorheen willen itereren. Denk hierbij een een interval, een list, een dictionary enzovoort.

Voor dit voorbeeld werken we, zoals eerder vermeldt, met een range functie met de waarde 6 en loopen we door ieder getal binnen het interval 0 tot en met 5. Tijdens iedere iteratie printen we het getal:

```
>>> for i in range(6):
...     print(i)
...
0
1
2
3
4
```

```
5
```

Let op! De i na het woordje *for* is de naam van mijn variabele. Het had evengoed een andere naam kunnen hebben.

If-statement

Wanneer een if-statement binnen, bijvoorbeeld een for-loop staat, is het mogelijk de break - of continue statement te gebruiken. Met behulp van deze statements kun je een loop laten stoppen of ervoor zorgen dat deze blijft itereren zonder een opdracht uit te voeren. Beide statements worden aan de hand van een voorbeeld gedemonstreerd.

Break statement

Stel dat we een for-loop hebben die 10 keer looped. Deze for-loop stoppen we zodra onze variabele *i* de waarde 7 heeft. Dit doen we met behulp van de break statement:

```
>>> for i in range(10):
...     if i == 7:
...             break
...     else:
...             print('De waarde van i is ', i)
...
De waarde van i is  0
De waarde van i is  1
De waarde van i is  2
De waarde van i is  3
De waarde van i is  4
De waarde van i is  5
De waarde van i is  6
```

Merk op dat *De waarde van i is 7* niet is geprint. Dit komt omdat de *break* statement ervoor zorgt dat we uit de for-loop stappen.

Continue statement

De continue statement wordt gebruikt om delen van de code binnen een iteratie over te slaan. We zouden deze statement bijvoorbeeld kunnen gebruiken in een for-loop die alle even getallen print en geen code uitvoert als het getal oneven is. Bijvoorbeeld:

```
>>> for i in range(10):
...     if i % 2 == 0:
```

```
...             print(i, ' is een even getal')
...     else:
...             continue
...
0  is een even getal
2  is een even getal
4  is een even getal
6  is een even getal
8  is een even getal
```

Merk op dat de print is overgeslagen voor alle oneven getallen.

Datastructuur

Een datastructuur is een methode/manier om data op te slaan. Deze sectie behandeld de volgende datatstructuren:

- Lists
- Dictionaries

List

Een list is een datastructuur die wordt aangeduid door rechte haken (`[]`). Lists kunnen items bevatten. Deze worden door komma's van elkaar gescheiden. List items kunnen bestaan uit String(s), Integer(s) of andere list(s). De items binnen een list zijn gealloceerd aan een index, deze start bij 0.

Dictionary

Een dictionary is een datastructuur die word aangeduid door accolades (`{}`). Dictionary items bestaan uit paren, namelijk een key-value pair hebt. Deze worden van elkaar gescheiden middels een dubbele punt (`:`). Hieronder is een voorbeeld van een lege dictionary en een dictionary met key-value pairs:

```
>>> {} # lege dictionary
{}
>>> {1: 'hello', 2: 'Welcome', 3: 'to', 4: 'Python'} # dictionary met key-value pair
{1: 'hello', 2: 'Welcome', 3: 'to', 4: 'Python'}
```

Ingebouwde Python functies

Zoals alle programmeertalen, heeft ook Python ingebouwde functies. Enkele functies die je moet kennen voor de toets zijn:

- print()
- range()
- len()

Deze functies worden in de volgende secties uitgelegd en gedemonstreerd.

print()

De print-functie is een krachtige functie die ingezet kan worden door beginnende programmeurs, zodat het duidelijk wordt wat de waarde van een gegeven is. Het is echter ook een tool om je logs bij te houden aan de serverside.

Print() is een functie en functies moeten aangeroepen worden met haakjes. Tussen de haakjes komt hetgeen te staan wat je wil printen, bijvoorbeeld;

```
>>> print("Hello world")
Hello world
```

range()

De range-functie wordt gebruikt voor loops en is een functie die je op drie manieren kunt gebruiken. Het is mogelijk om de functie minimaal 1 - en maximaal 3 waarden te geven.

range() met 1 waarde

Wanneer je 1 waarde meegeeft aan de range functie begin je te itereren bij 0 tot het ingevoerde getal-1. Stel als we `range(10)` gebruiken, dan zal de loop beginnen te tellen bij 0 tot en met 9:

```
>>> for i in range(10):
...     print('i is ',i)
...
i is  0
i is  1
i is  2
i is  3
i is  4
i is  5
i is  6
i is  7
```

```
i is  8
i is  9
```

range() met 2 waarden

De range functie accepteert ook twee waarden, waarbij te ingevoerde getallen dienen als een interval. Stel als we `range(2, 12)` gebruiken, dan zal de loop beginnen te tellen bij 2 en tellen tot en met 12-1=11:

```
>>> for i in range(2, 12):
...     print('i is ',i)
...
i is  2
i is  3
i is  4
i is  5
i is  6
i is  7
i is  8
i is  9
i is  10
i is  11
```

range() met 3 waarden

Door een derde waarde mee te geven aan de range functie, kun je de toenamefactor aangeven, `range(0, 20, 2)` zal dus beginnen met het tellen vanaf 0 tot en met 20-1=19 in stappen van 2:

```
>>> for i in range(0, 20, 2):
...     print('i is ', i)
...
i is  0
i is  2
i is  4
i is  6
i is  8
i is  10
i is  12
i is  14
i is  16
i is  18
```

len()

De length functie is een ingebouwde Python functie die je kunt gebruiken om de inhoud van een list en/of een dictionary te achterhalen. Deze functie geeft altijd de hoeveelheid items in een list/dictionary.

len() met lists

Wanneer je de length functie gebruikt om de items uit een list te tellen, resulteert dit altijd in het aantal elementen dat zich in een list bevindt. Als een list leeg is, krijg je een 0 terug. Als een list waarden bevat, dan krijg je de hoeveelheid elementen als cijfer terug. Bijvoorbeeld:

```
>>>len([])
0


>>> a = [1, 2, 3, 4]
>>> len(a)
4
```

len() met dictionary

In de vorige sectie hebben we gezien hoe de length functie toegepast kan worden om het aantal elementen in een list te achterhalen. Deze functie werkt exact hetzelfde bij dictionaries. Bij een lege dictionary krijg je een length 0 terug. Anders het aantal key-value pairs. Bijvoorbeeld:

```
>>> len({})
0


>>> d = {1: 'hello', 2: 'there', 3: 'you'}
>>> len(d)
3
```

Uitwerkingen opgaven ter toetsvoorbereiding (week 4)

Lists (=array)

Gegeven is list

```
 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

. Gebruik deze list om de volgende vragen te beantwoorden.

1. Geef de waarde van de 5e index van a.

```
a[5]
```

2. Geef de waardes van de 0e - tot en met de 3e index van a.

```
a[:3]
```

3. Geef de waardes van de 3e - tot en met de 5e index van a.

```
a[3:6]
```

4. Geef de waarde van de laatste index van a.

```
a[-1]
```

5. Geef de waarde van de eerste index van a.

```
a[0]
```

Dictionary

Gegeven is de volgende dictionary `d={1: 'hello', 2: 'there', 3: 'you'}`. Gebruik deze dictionary om de volgende vragen te beantwoorden.

1. Schrijf een for-loop die alle keys van dictionary d loopt en print.

```
>>> d = {1: 'hello', 2: 'there', 3: 'you'}
>>> for key in d.keys():
...     print('key is ', key)
...
key is  1
key is  2
key is  3
```

2. Schrijf een for-loop die alle values van dictionary d loopt en print.

```
>>> d = {1: 'hello', 2: 'there', 3: 'you'}
>>> for value in d.values():
...     print('value is ', value)
...
value is  hello
value is  there
value is  you
```

3. Schrijf een for-loop die de key-value pairs van dictionary d loopt en print.

Er zijn twee manieren om deze vraag te beantwoorden. De eerste methode is het opvragen van de keys en middels de keys de value te achterhalen:

Dictionaries:

D = {1: “hello”, 2: “hello”, 3: “you”}

1. Schrijf een for-loop die alle keys van dictionary D loopt
2. Schrijf een for-loop die alle values van dictionary D loopt
3. Schrijf een for-loop die de key, value-pairs van dictionary D loopt

Loop:

1. Schrijf een for loop die 10 keer loopt. Deze loop moet bij iedere iteratie het getal printen.
2. Schrijf een for loop die array a loopt. Deze loop moet iedere index van de array printen.

Datatypes:

1. Maak een variabele x aan. Sla hier een waarde 5 in op. Bepaal met behulp van een ingebouwde Python functie de datatype van variabele x
2. Maak een variabele y aan. Sla hier een waarde .5 in op. Bepaal met behulp van een ingebouwde Python functie de datatype van variabele y.
3. Maak een variabele z aan. Sla hier een waarde “Hi there” in op. Bepaal met behulp van een ingebouwde Python functie de datatype van variabele z.

Loop:

1. We hebben een array a=[5,6,7,8,9,9,10,12,24,5]. Loop deze array en print alleen de even getallen. Als een getal oneven is ga je door met je loop.