

Python basis

herhaling
3e jaar

Volledig in te halen!

Werkboek 'Python' p. 6-39

ook antwoorden beschikbaar op Pelckmansportaal
(eerst bestellen, dan gebruiken)

+

bijhorende video's: "video 1" tot en met "video 18"

<https://www.youtube.com/@UHasseltTutorials/videos>

+

deze cursus

deel 1

Wat is programmeren?

Waarom Python?

Werkomgeving: repl.it of Thonny app.

Decompositie

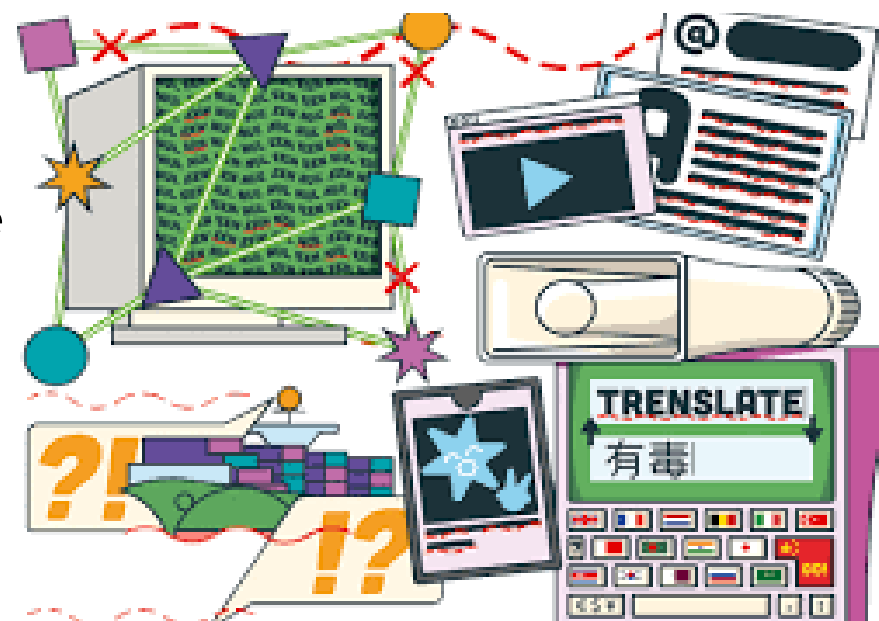
Abstractie

Patroonherkening

Algoritme

programmeertaal

Een programmeertaal is een formele taal waarmee je **instructies aan een computer** kunt geven. Het is een manier om de computer te vertellen wat hij moet doen, vergelijkbaar met hoe we natuurlijke talen gebruiken om met elkaar te communiceren. Programmeertalen hebben hun **eigen syntaxis** (grammatica) en regels, zodat computers de instructies **kunnen begrijpen en uitvoeren**.



programmeren

is een instructie schrijven voor een computer. De computer voert deze instructie vervolgens uit. Deze instructie wordt geschreven in een **programmeertaal**, bijvoorbeeld Python. Het is belangrijk dat je de stapjes in de juiste volgorde door de computer laat uitvoeren. Dit kan vaak heel lastig zijn. Als je iets verkeerd ingeeft, voert de computer dit ook fout uit of doet zelfs helemaal niets.

Een script is een reeks instructies die door een computer kan worden uitgevoerd. **Een script** is een veelvoorkomend type computerprogramma, met als kenmerk dat het niet vooraf hoeft te worden **gecompileerd**. Het wordt in **realtime** geïnterpreteerd en uitgevoerd.

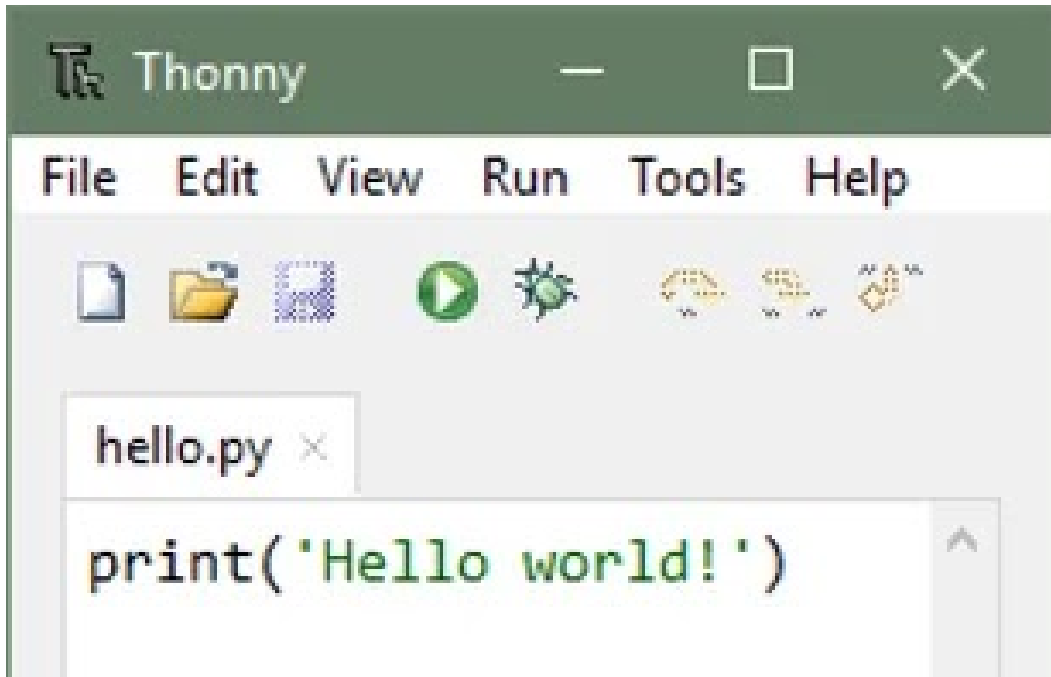
```
31
32 self.file = None
33 self.fingerprints = set()
34 self.logdups = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'fingerprint.log'), 'a')
39     self.file.seek(0)
40     self.fingerprints.update(set(self.logger.handlers))
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('debug')
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)
```

Waarom Python?

Python is niet alleen een **krachtige taal** die wordt gebruikt door professionals in de industrie, maar ook een uitstekende keuze voor beginners vanwege zijn leesbaarheid en eenvoudige syntax (schrijfwijze).

Hier zal je stap voor stap door de basisconcepten van programmeren gaan, zoals variabelen, datatypes, functies e.a.

Ons doel is niet alleen om je vertrouwd te maken met **Python als programmeertaal**, maar ook om je te helpen bij **het ontwikkelen van het analytisch denken en probleemoplossend vermogen** dat essentieel is binnen de informaticawetenschappen. Elk hoofdstuk bevat praktische voorbeelden en oefeningen die zijn afgestemd op de uitdagingen die je tegenkomt in je studie.



Thonny (app op je laptop)



repl.it (online platform)

▶ **Decompositie**

Problemen opsplitsen in verschillende delen zodat we ons op elk deel afzonderlijk kunnen concentreren.

▶ **Abstractie**

Het weglaten of negeren van bepaalde details zodat we ons op het basisprobleem kunnen concentreren.

▶ **Patroonherkenning**

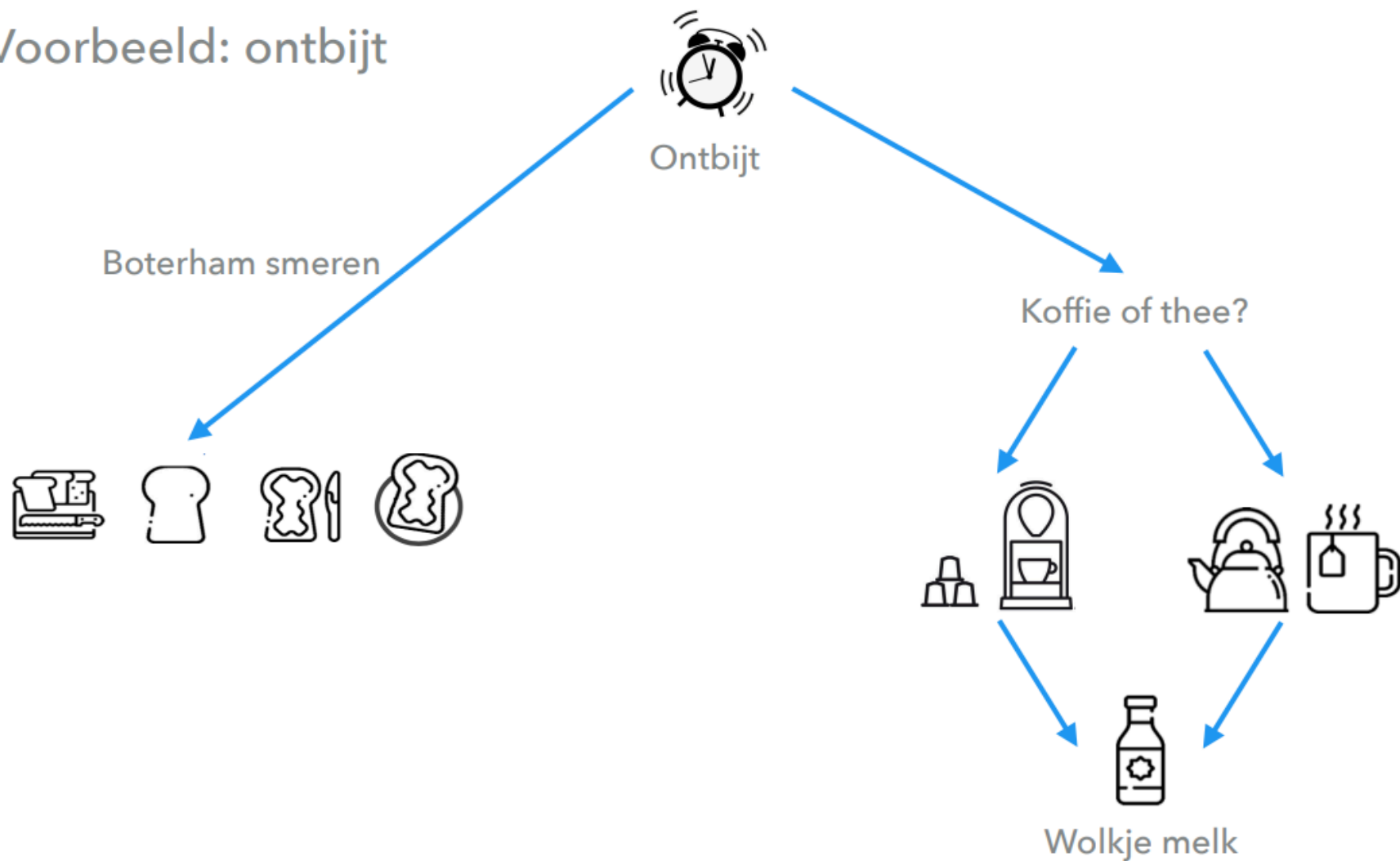
Gelijkenissen of patronen herkennen tussen gegevens of verschillende problemen.

▶ **Algoritme**

Reeks van opeenvolgende instructies die stapsgewijs kunnen uitgevoerd worden om een bepaalde taak uit te voeren.

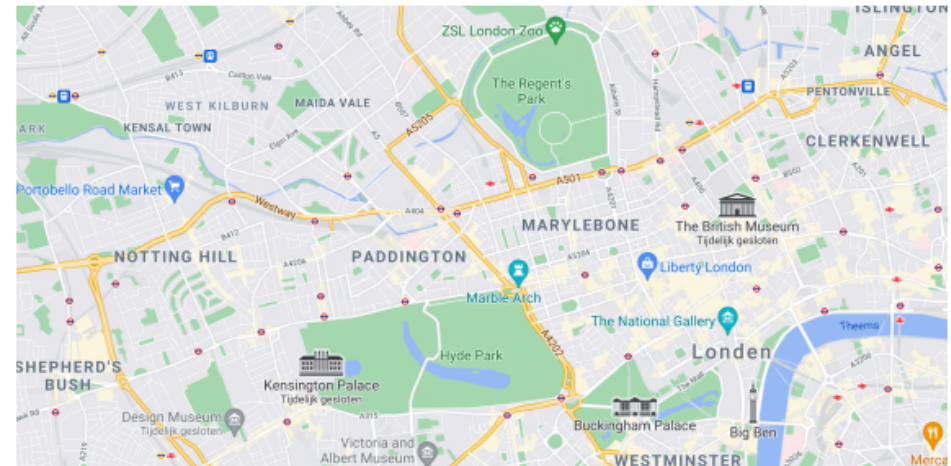
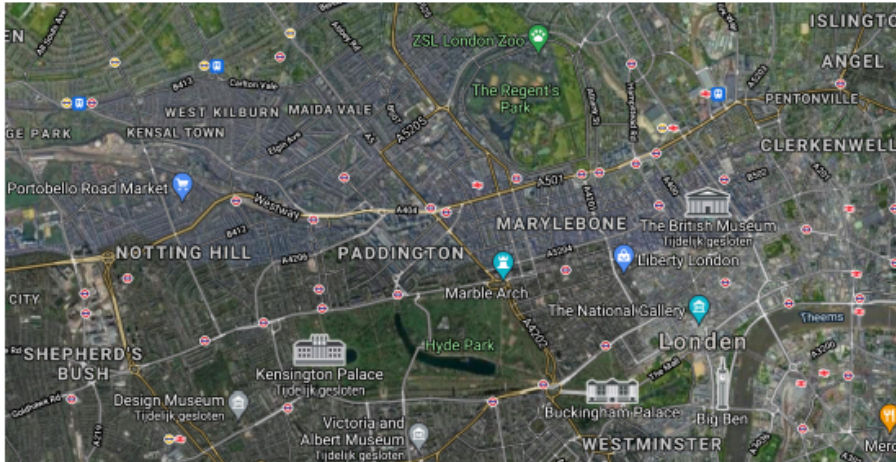
DECOMPOSITIE

► Voorbeeld: ontbijt



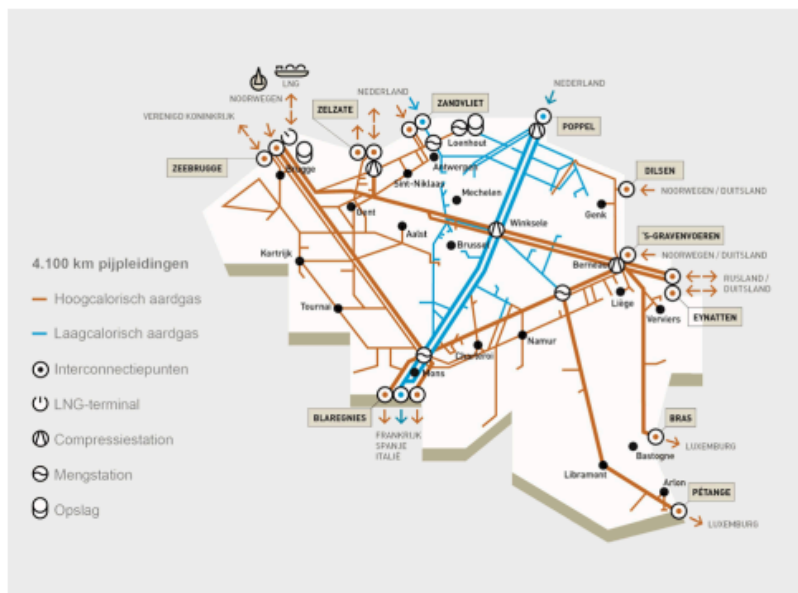
ABSTRACTIE

- ▶ Verbergen van onnodige complexiteit.
- ▶ Extraheren van de nodige informatie.
- ▶ Vinden van een goede data-representatie.
- ▶ Relaties / overeenkomsten vinden met andere problemen.



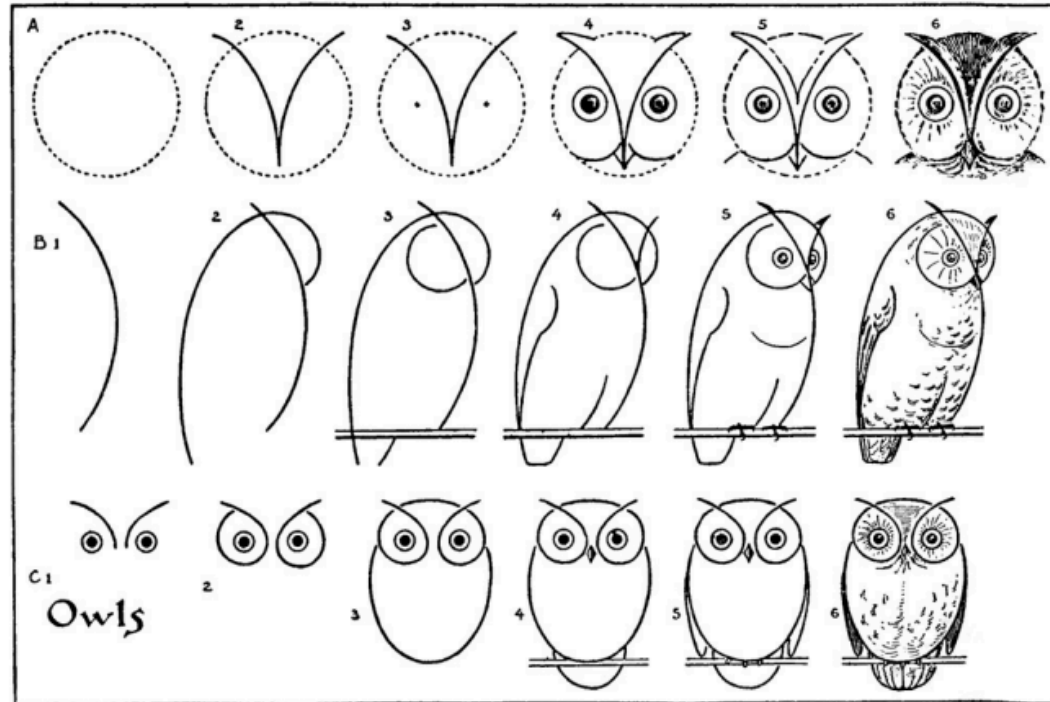
PATROONHERKENNING

- ▶ Oplossingsstrategie
 - ▶ toepassen op gelijkaardige problemen;
 - ▶ aanpassen aan nieuwe maar gelijkaardige contexten.



ALGORITME

- Leren formuleren van instructies die door de mens en computer kunnen begrepen worden.



kennen en kunnen

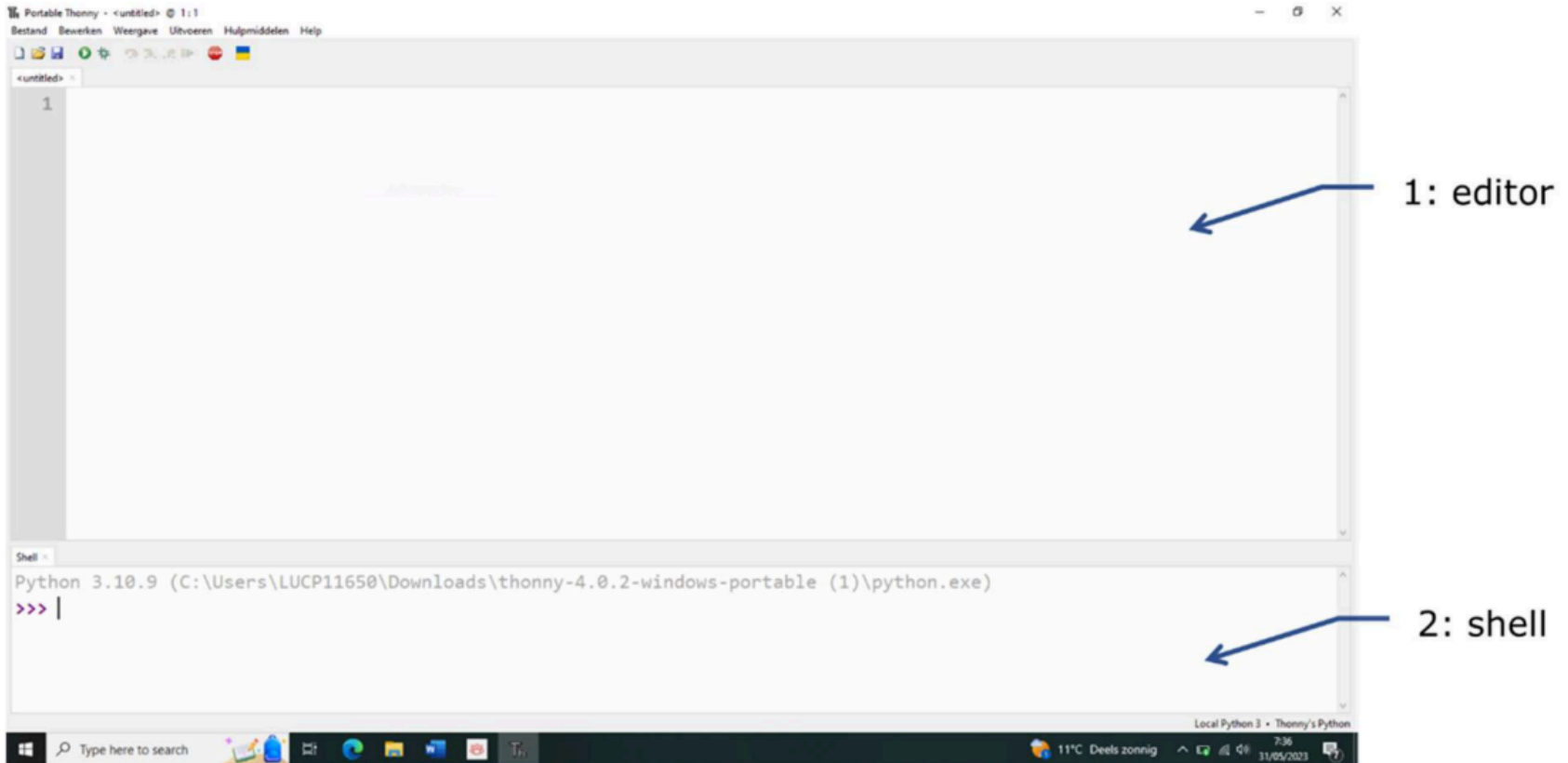
- De werkomgeving kennen (bv. Thonny). Je leert de interface en basisfuncties van een programmeeromgeving gebruiken.
- Een probleem opdelen in deelproblemen (decompositie). Je verdeelt een groter probleem in kleinere, beter oplosbare stukjes.
- Overbodige informatie wegfilteren (abstractie). Je focust je alleen op de relevante gegevens die je nodig hebt om het probleem op te lossen.
- Een samenhang of patroon herkennen (patroonherkenning). Je ontdekt herhalende structuren of gelijkaardige stappen in problemen of oplossingen.
- **Een oplossing omzetten naar duidelijke instructies (algoritme).** Je schrijft een reeks logische stappen die een computer kan uitvoeren om een probleem op te lossen.

Thonny

We gebruiken Thonny IDE (is een programma dat je helpt om code te schrijven, testen en uitvoeren op één plek.).

Thonny is een gebruiksvriendelijke programmeeromgeving die speciaal is ontworpen voor beginners. De interface is overzichtelijk en biedt een minimalistische benadering van programmeren. Je vindt alle benodigde tools en functies op één plek, waardoor je snel aan de slag kunt.

De eerste keer dat je de Thonny programmeer omgeving opent, staan er twee schermen open.



Editor: Dit is het primaire scherm van Thonny **waar je je Python-code kunt schrijven**, bewerken en opmaken.

Shell (CLI): is een interactieve Python-omgeving **waarin je Python-code direct kunt uitvoeren** en de resultaten kunt zien. herkenbaar door '>>>'

deel 2

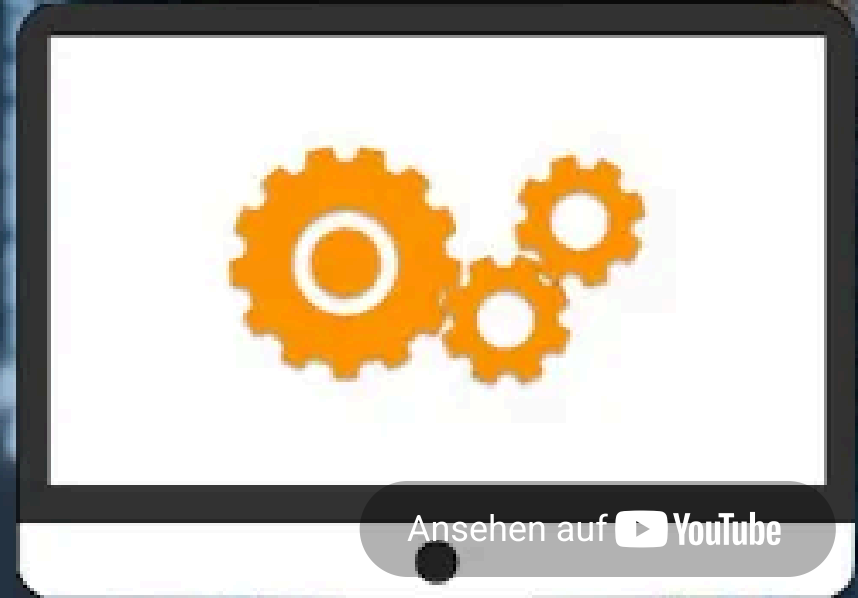
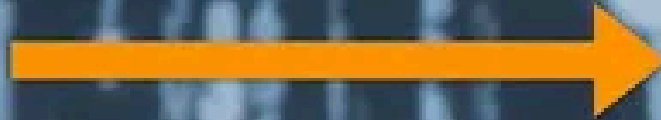
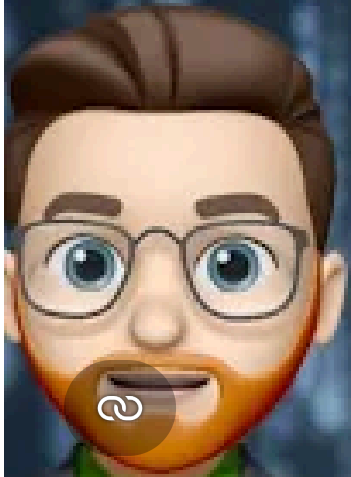
invoer >>> verwerking >>> uitvoer

operatoren

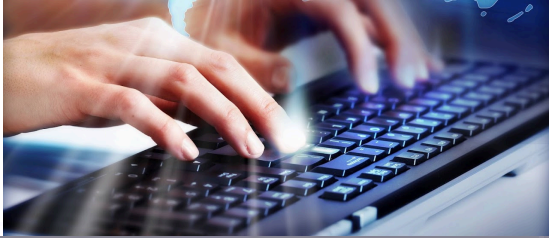
sequentie

algoritme

GEGEVENSVERWERKEN PROCES



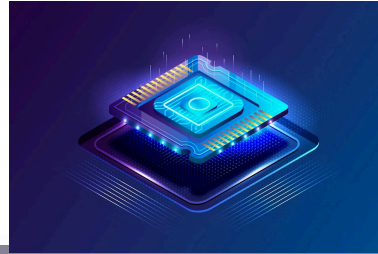
herhaling:
invoer >>> verwerking >>> uitvoer



Invoer

tekens
(toetsen drukken op klavier)

input()



Verwerking

samenvoegen
(CPU: tekst maken, berekeningen)

**het manipuleren of
berekenen van gegevens**



Uitvoer

zichtbaar maken
op het scherm / afdrukken

print()

vergelijkende operatoren in Python

A groter dan B: **a > b**

A kleiner dan B: **a < b**

A gelijk aan B: **a == b**

A niet gelijk aan B: **a != b**

A groter of gelijk aan B: **a >= b**

A kleiner of gelijk aan B: **a <= b**

rekenkundige operatoren in Python

A vermenigvuldigen met B: **a * b**

A optellen/samenvoegen met B: **a + b**

A verminderen met B: **a - b**

A delen door B: **a / b**

A tot de macht van B: **a ** b** (7^3 in Python is `7 ** 3`)

A delen door B, enkel gehele deling: **a // b** (`7 // 3 >>> 2`)

A delen door B, enkel restdeling: **a % b** (`7 % 3 >>> 1`)

sequentie

Een sequentie is een onderdeel van een algoritme waarin de stappen in een vaste volgorde worden uitgevoerd — van boven naar beneden.

□ Voorbeeld (met analogie): Eerst neem je een boterham, dán pas smeer je er choco op — niet omgekeerd.

In een programma is dat net zo: eerst `input()`, dan `print()`.

algoritme

Algoritme-opdracht – Boterham met choco

Stel dat je iemand moet programmeren alsof het een robot is die een boterham met choco moet smeren.

Schrijf in je eigen woorden een stappenplan om deze taak tot een goed einde te brengen.

Bekijk daarna het filmpje: [Leer programmeren met boterham met choco.](#)

Evalueer je stappenplan:

Zou jouw plan werken als je het letterlijk uitvoert?

Wat liep er eventueel mis?

Hoe kan je het verbeteren?

□ Dit verbeteren heet **debuggen**:

fouten of onduidelijkheden in je algoritme opsporen en aanpassen.



algoritme

Algoritme-opdracht – Boompjes

Schrijf eens in je eigen woorden op **welke stappen je moet ondernemen om het aantal boompjes te berekenen** dat de boer moet aankopen.

Gegeven:

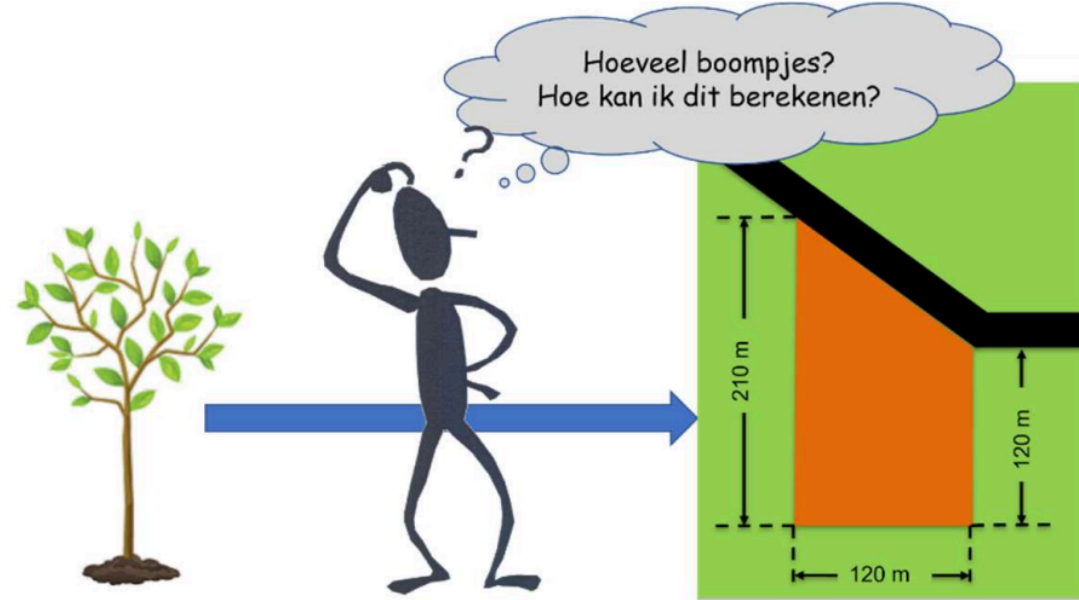
Een akker in de vorm van een rechthoekig trapezium.

De afmetingen van de akker zijn aangeduid op de figuur hiernaast.

Een appelboom heeft een oppervlakte van 8×8 m nodig om goed te groeien.

Gevraagd:

Hoeveel bomen moet de boer aankopen?



algoritme

Als ik een tekst in het Nederlands zou schrijven met fouten erin, zou je nog steeds in staat zijn om de tekst te begrijpen. Maar als we AI even buiten beschouwing laten, kan een computer dat niet.

Daarom is de **syntaxis** (schrijfwijze) van een computertaal van groot belang.

Gelukkig is de syntaxis van Python over het algemeen vrij eenvoudig en levert het leesbare **script's** (programma in tekstvorm) op. Dit betekent dat je meer aandacht kunt besteden aan het denkproces achter het programmeren.

Controleer je algoritme van opdracht 1.3 door het stap voor stap uit te voeren in de Shell.

Je geeft één voor één de bewerkingen in. Sluit de bewerkingen af door op ENTER te drukken.

Waarschijnlijk kom je als resultaat 156,25 bomen uit. Dit komt omdat je `/` gebruikt hebt als wiskundige operator. Verander nu het symbool `/` met `///` (twee maal het deelteken) en controleer wat je nu krijgt.

Het symbool `///` wordt gebruikt voor de gehele deling. De gehele deling geeft als uitkomst het gehele getal dat kleiner of gelijk is aan de deling.

in de volgende dia zie je het verschil tussen beide bewerkingen:

Oplossing: Algoritme-opdracht – Boompjes

Een akker in de vorm van een rechthoekig trapezium met afmetingen $a=120$ m, $b=80$ m en hoogte $h=100$ m.

Elke appelboom heeft een oppervlakte van $(8 \times 8 =) 64 \text{ m}^2$ nodig

▮ Stap 1: Begrijpen van het probleem

We moeten de oppervlakte van de akker berekenen, en dan uitrekenen hoeveel bomen daarin passen als elke boom 64 m^2 nodig heeft.

De oppervlakte A van een trapezium wordt berekend met:

$$A = \frac{(a + b)}{2} \times h$$

waarbij:

a = lengte van de ene basis

b = lengte van de andere basis

h = hoogte van het trapezium

▮ Stap 2: Algoritme in stappen (in gewone taal)

1. Bepaal de afmetingen van de akker.
2. Bereken de oppervlakte van het trapezium.
3. Bereken de oppervlakte die één boom nodig heeft.
4. Deel de oppervlakte van het trapezium door de oppervlakte per boom.
5. Rond het resultaat af naar beneden (want je kan geen halve boom planten).
6. Druk het aantal bomen af.

$$\text{Oppervlakte akker} = \frac{(120 + 80)}{2} \times 100 = 10000 \text{ m}^2$$

$$\text{Aantal bomen} = 10000 // 64 = 156$$

De boer moet dus 156 bomen aankopen.

oplossing opdracht 'Boompjes'

Aantal bomen nodig
$b \leftarrow 120$
$h \leftarrow 120$
$H \leftarrow 210$
$oppervlakte_boom \leftarrow 8 \cdot 8$
$A \leftarrow \frac{h + H}{2} \cdot b$
$aantal_bomen \leftarrow \frac{A}{oppervlakte_boom}$
Output: $aantal_bomen$
EINDE

```
Shell x
Python 3.10.9 (C:\Users\LUCP11650\Downloads\thonny-4.0
>>> (210 + 120)/2*120
19800.0
>>> 19800/(8*8)
309.375
>>> 19800//(8*8)
309
>>> |
```

fout: onjuiste operator

deel 3

syntax

variabelen

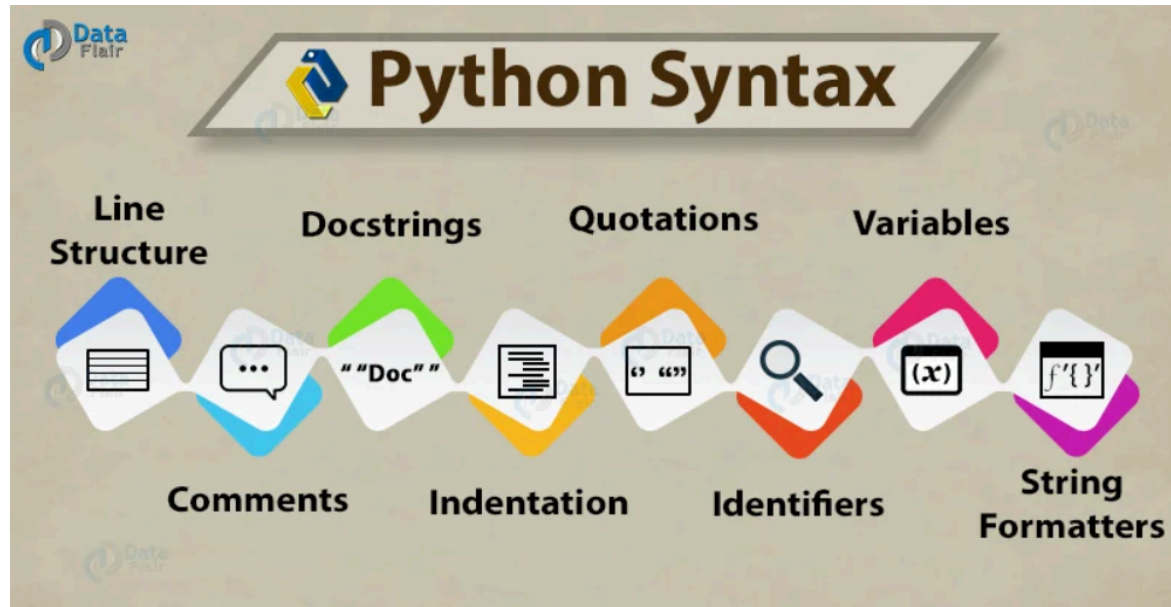
toekening

assignment statement

expressie

syntax

De juiste manier waarop code geschreven moet worden in een programmeertaal



Syntax error:

Foutmelding wanneer de code niet volgens de juiste regels is geschreven.

variabele



In Python is een variabele een naam die je koppelt aan een waarde. Je kunt het zien als een vakje in een kast: de naam van het vakje is de variabele, en wat je erin stopt is de waarde.

Variabelen gebruik je om gegevens op te slaan die je later in je programma nodig hebt. Python is dynamisch getypeerd: dat betekent dat je variabelen kunt gebruiken voor allerlei soorten gegevens.

Voorbeeld:

```
naam = "Sara"
```

```
leeftijd = 14
```

Hier krijgt variabele *naam* de waarde "Sara" en *leeftijd* de waarde 14.

Let op: in één variabele kan maar één waarde tegelijk bewaard worden. Als je een nieuwe waarde toekent, vervangt die de vorige.

variabele namen

Bij het oplossen van Fysica vraagstukken maak je ook gebruik van variabelen, meestal bestaande uit één letter of symbool. Passen we dezelfde methode toe in een script, dan is het later moeilijk te achterhalen waarvoor deze variabele gebruikt werd.

We geven daarom onze variabele een **betekenisvolle naam**.
Bekijk vorig voorbeeld.

Er zijn enkele **regels** die we moeten volgen!

Een variabele naam:

- mag enkel bestaan uit letters, cijfers en "underscores" (_).
- mag geen spatie bevatten
- mag niet beginnen met een cijfer.
- bevat enkel kleine letters
- mag geen gereserveerd woord (keyword) van Python zijn.



koekjes = "naaigerief" **X**

toekenning =

Toekenning met =

Met het toekenningsteken = kun je in Python een waarde toekennen aan een variabele. Dit lijkt op het gelijkheidsteken uit de wiskunde, maar betekent iets anders. In programmeertaal hoort = tot de groep van 'assignment operators'.

Onderstaand voorbeeld lees je dus **niet** als: "lange_zijde is **gelijk aan** 210", **maar** als: "de variabele *lange_zijde* **krijgt de waarde** 210".

```
lange_zijde = 210
```

```
Lange_zijde = 210
```

toekenning / assignment

Onderstaand voorbeeld toont het verschil tussen het gelijkheidsteken en het toekenningsteken.

Python "leest" van rechts naar links!

```
aantal = 5
```

```
aantal = aantal + 1
```

De variabele *aantal* kan (wiskundig) nooit gelijk zijn aan *aantal* +1.

eerste regel: In onze code de waarde 5 wordt toegekend aan de variabele *aantal* .

tweede regel: gaat het script de waarde van variabele *aantal* ophalen, en gaat hierbij 1 optellen en het resultaat van deze bewerking opnieuw aan de variabele *aantal* toekennen.

Bij een toewijzing wordt de bewerking rechts van het toewijzingsteken eerst uitgevoerd, en het resultaat van deze bewerking wordt dan toegekend aan de variabele links van het toewijzingsteken.

assignment statement

Een assignment statement is een **volledige opdracht waarbij een variabele een waarde krijgt toegewezen**.

Het bestaat uit een variabele, gevolgd door een toekenningsteken (=), gevolgd door een **expressie** (20 + 5) of **waarde** (maandag).

Voorbeeld **assignment statement**:

```
temp_celsius = 20 + 5
```

```
eerste_dag = "maandag"
```

In dit voorbeeld is `temp_celsius = 20 + 5` een assignment statement. De expressie `20 + 5` wordt eerst uitgerekend (geeft 25), van rechts naar links, en deze waarde wordt dan toegekend aan de variabele `temp_celsius`.

expressie

Een expressie (Engels: expression) is een stukje code dat een waarde oplevert.

Dit kan bijvoorbeeld:

- een berekening zijn: $3 + 5$
- een vergelijking: $10 > 3$
- of een combinatie van variabelen en operatoren: `leeftijd * 2`, `"Hallo" + " wereld"`

Expressies worden vaak gebruikt in combinatie met opdrachten zoals een toekenning:

`aantal = 3 + 5`

deel 4

datatypes:

string

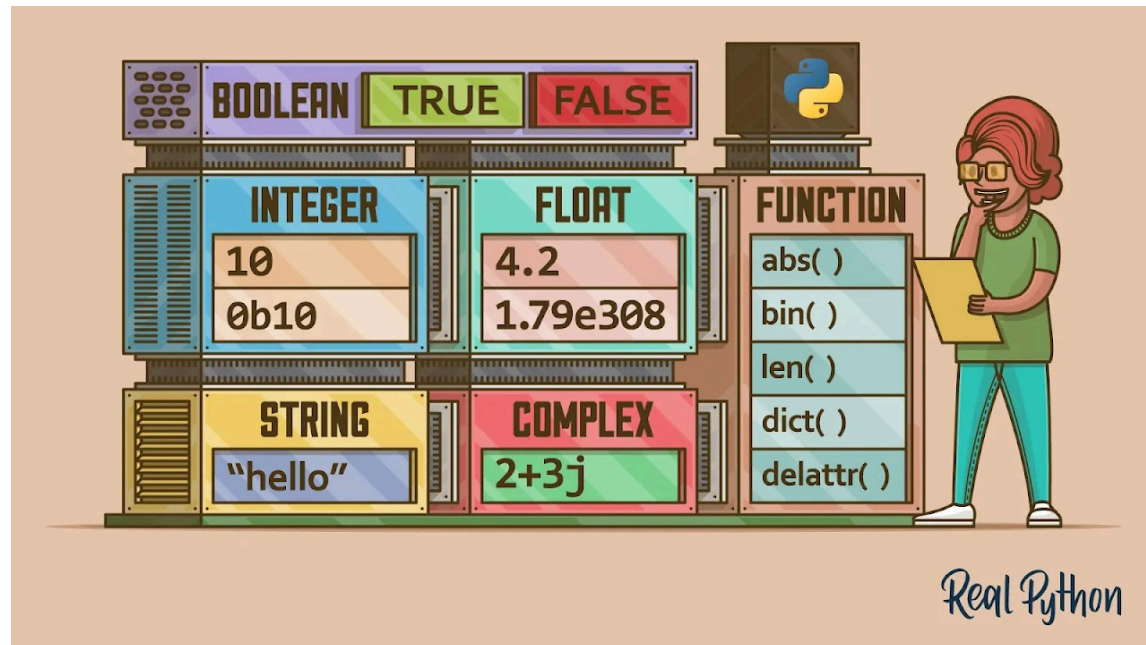
integer

float

datatype

In Python is een variabele een naam die aan een waarde is gekoppeld, wat betekent dat je variabelen kunt gebruiken om verschillende **soorten gegevens** op te slaan.

Hier gaan we enkel over de drie datatypes spreken: string, integer en float.



string

Wat is een string (str)?

Een string is een gegevenstype dat een **reeks tekens** voorstelt, zoals woorden, zinnen of zelfs cijfers als tekst. Strings gebruik je om tekstuele informatie op te slaan.

Voorbeelden van strings zijn: "Hallo", 'Python', of "123" (als tekst, niet als getal).

Strings **duid je aan door de tekst tussen dubbele aanhalingstekens** ("...") of enkele aanhalingstekens ('...') te plaatsen. Let op: je moet het begin en einde van je string met hetzelfde type aanhalingsteken afsluiten.

Correcte voorbeelden:

```
bericht = "Welkom!"
```

```
naam = 'Anna'
```

Fout voorbeeld:

```
fout = "Hallo'
```

Dit geeft een syntax error omdat de string niet correct is afgesloten.

string in de Shell

probeer nu zelf in de Shell van Thonny app
hier gebruiken we functie print(), later meer erover.

```
Python 3.10.9 (C:\Users\LUCP11650\  
>>> print("Hello World!")  
Hello World!  
>>> print('Hello World')  
Hello World  
>>> print("'s avonds laat")  
's avonds laat  
>>> |
```

concatinatie

```
tekst1 = "Hallo,"
```

```
tekst2 = " Sara."
```

```
groet = tekst1 + tekst2
```

Wat je hier ziet in het Engels “concatenate” genoemd, in het Nederlands is dit “aaneenschakeling”. Je kan met het “+”-teken twee strings aan elkaar schakelen...

Met de code `tekst1 + tekst2`

schakel je dus de twee strings aan elkaar, eerst tekst1, gevolgd door tekst2 zonder spaties!

integer

Dit is een geheel getal, bijvoorbeeld 5, -3, 0.

Het wordt gebruikt voor het opslaan van numerieke waarden zonder decimalen.

Je gebruikt integers voor tellingen, berekeningen, of om iets op een vaste positie te plaatsen.

Voorbeelden:

leeftijd = 15

punten = -3

sterren = 5

float

Dit is een drijvendekommagetal, bijvoorbeeld 3.14, -0.5, 2.0. Het wordt gebruikt voor het opslaan van numerieke waarden met decimalen.

Floats gebruik je wanneer je te maken hebt met breuken, percentages of andere metingen die geen hele getallen zijn.

✓ Voorbeelden:

prijs = 3.99

gemiddelde = $(8 + 7 + 6) / 3$

In het laatste voorbeeld zorgt de deling / ervoor dat het resultaat automatisch een float wordt.

□ Regels voor floats:

Je gebruikt een punt (.) als decimaalscheiding, geen komma.

oefening

Float-getallen kunnen ook het resultaat zijn van wiskundige bewerkingen zoals delen of worteltrekken. Je kunt floats combineren met integers in berekeningen. Het resultaat is dan meestal een float.

probeer dit in Editor van Thonny:

```
aantal_boeken = 3
prijs_per_boek = 12.5
totaal = aantal_boeken * prijs_per_boek
print(totaal)
```

klik daarna op het groene bolletje bovenaan en zie resultaat in de Shell:

```
>>> 37.5
```

deel 5

functies:

print()

input()

int()

float()

print()

output geven met de print-opdracht

Wil je het resultaat kennen of de waarde van een variabele weergeven, dan kan dit met de print()-functie. Wanneer je de print-opdracht gebruikt, dan vraag je **om uitvoer te geven naar het scherm**.

Wil je dus het resultaat, het aantal bomen kennen (zie oefening bomen), dan kan dit met `print(aantal_bomen)`

vertaling: maak de waarde van de variabele `aantal_bomen` zichtbaar op het scherm

Je moet niet steeds gebruik maken van een variabele, maar kan ook rechtstreeks de bewerking tussen de haakjes plaatsen:

`print((210+120)/2*120)`

Of je kan verschillende gegevens (datatypes) combineren door ze te scheiden met een komma. Hieronder een combinatie van een string en expressie met integers

`print("Oppervlakte veld:",(210+120)/2*120)`

Belangrijk is op te merken dat na een print-opdracht er steeds een nieuwe lijn gestart wordt.

script met print()

hier **vertalen** we ons eerste **algoritme naar Python**, naar script (programma in tekstvorm). Voor een commentaar gebruiken we #. Test dit programma in Thonny:

Aantal bomen nodig
$b \leftarrow 120$
$h \leftarrow 120$
$H \leftarrow 210$
$oppervlakte_boom \leftarrow 8 \cdot 8$
$A \leftarrow \frac{h + H}{2} \cdot b$
$aantal_bomen \leftarrow \frac{A}{oppervlakte_boom}$
Output: <i>aantal_bomen</i>
EINDE

```
#aantal boompjes
breedte = 120
korte_zijde = 120
lange_zijde = 210
oppervlakte_boom = 8*8
oppervlakte_veld = (lange_zijde + korte_zijde)/2*breedte

aantal_bomen = oppervlakte_veld/oppervlakte_boom
print("Aantal bomen:",aantal_bomen)
```

Je kan je script starten door de sneltoets F5 te drukken of te klikken op de startknop.



werken als een pro

Documenteer je script.

Het is belangrijk dat je script leesbaar is en daarom van voldoende commentaar voorzien wordt.

Op die manier kan iemand de gedachtegang van de programmeur volgen. Dit maakt het makkelijker om later aanpassingen aan het script aan te brengen.

Je kan opmerkingen toevoegen door deze vooraf te laten gaan door de hashtag (#).

Alles na # wordt genegeerd en is louter ter info.

werken als een pro

Gebruiksvriendelijke interface

Je interactie met de gebruiker van je script moet ook gebruiksvriendelijk zijn.

Dit kan je bereiken door meer print-opdrachten toe te voegen.

Script met verzorgde output en extra uitleg voor/opmerkingen:

```
# Aantal bomen berekenen
```

```
lange_zijde = 210
```

```
korte_zijde = 120
```

```
breedte = 120
```

```
oppervlakte_boom = 8*8
```

```
oppervlakte_veld = (lange_zijde+korte_zijde)/2*breedte # Oppervlakte veld berekenen
```

```
aantal_bomen = oppervlakte_veld/oppervlakte_boom # Aantal bomen berekenen
```

```
# Output
```

```
print("Gegevens:")
```

```
print("Lange zijde:",lange_zijde)
```

```
print("Korte zijde:",korte_zijde)
```

```
print("Breedte:",breedte)
```

```
print()
```

```
print("Resultaat:")
```

```
print("Aantal bomen aan te kopen:",aantal_bomen)
```

opdracht "bussen"

Schrijf een programma in Thonny dat berekent hoeveel volle bussen nodig zijn om een groep personen te vervoeren, en hoeveel personen er nog overblijven nadat de bussen gevuld zijn.

Voer in je programma de volgende gegevens in:

Aantal personen : 137

Capaciteit van één bus: 52

Je programma moet het aantal volle bussen dat nodig is en het aantal personen dat overblijft berekenen.

Gebruik hiervoor:

De operator // voor gehele deling (voor het aantal bussen)

De operator % voor de rest (personen die overblijven)

mogelijke oplossing "bussen"

```
# Voorbeeldcode "bussen":  
aantal_personen = 137  
buscapaciteit = 52  
volle_bussen = aantal_personen // buscapaciteit  
overblijvend = aantal_personen % buscapaciteit  
print("Aantal volle bussen:", volle_bussen)  
print("Personen zonder plaats in een bus:", overblijvend)
```

Shell:

```
>>> Aantal volle bussen: 2  
>>> Personen zonder plaats in een bus: 33
```

input()

In deze verkenningsfase ga je onderzoeken hoe je programma (script) **kan vragen** om via het toetsenbord **data in te geven**.

In deze verkenning ga je:

- de input-instructie onderzoeken
- je kennis verdiepen van drie belangrijke data types
- leren de data van het ene type om te vormen naar het andere type.

input() oefening

Python is een zeer leesbare taal. Wanneer je alle Engelse instructies zou vertalen naar het Nederlands, dan kan je al een goed idee krijgen wat het script gaat doen.

Bestudeer onderstaande scripts.

Wat denk je dat er gaat gebeuren?

Typ de code over in Thonny en controleer.

#Script 1

```
getal = input("Geef getal in:")  
print("Het ingegeven getal was", getal)
```

#Script 2

```
getal = input("Geef getal in:")  
dubbel_getal = 2 * getal  
print("Het ingegeven getal was", getal)  
print("Het dubbele van het getal is", dubbel_getal)
```

Foutje?

```
<untitled> * x
1  getal=input("Geef getal in:")
2  dubbel_getal=2*getal
3  print("Het ingegeven getal was",getal)
4  print("Het dubbele van het getal is", dubbel_getal)

Shell x
>>> %Run -c $EDITOR_CONTENT
Geef getal in:5
Het ingegeven getal was 5
Het dubbele van het getal is 55
>>>
```

In Python wordt **alles** wat je invoert **via input()** automatisch behandeld als **een string (tekst)**, ook al typ je een cijfer in.

Bv: als je 5 invoert, dan wordt dit opgeslagen als de tekst "5". Wanneer je dan `2 * getal` doet, krijg je niet de wiskundige vermenigvuldiging (10), maar de string wordt 2 keer na elkaar geplakt: "5" wordt "55". Dit noemt men stringherhaling.

ja, foutje...wat nu?

Om dit te vermijden, moet je **de invoer eerst omzetten (casten)** naar een getal met **int()** of **float()**:

voorbeeld:

```
getal = int(input("Geef een getal in:"))
```

```
dubbel_getal = 2 * getal
```

```
print("Het ingegeven getal was", getal)
```

```
print("Het dubbele van het getal is", dubbel_getal)
```

Opgelet: als je een decimaal getal wilt invoeren, gebruik dan **float()** in plaats van **int()**.

casten (omzetten, converteren)

De instructie die je gebruikt om te converteren hangt enkel af van het type naar waar je converteert, en niet van het oorspronkelijke variabel type:

Converteren naar een integer:

```
geheel_getal = int(gegeven)
```

Converteren naar een float:

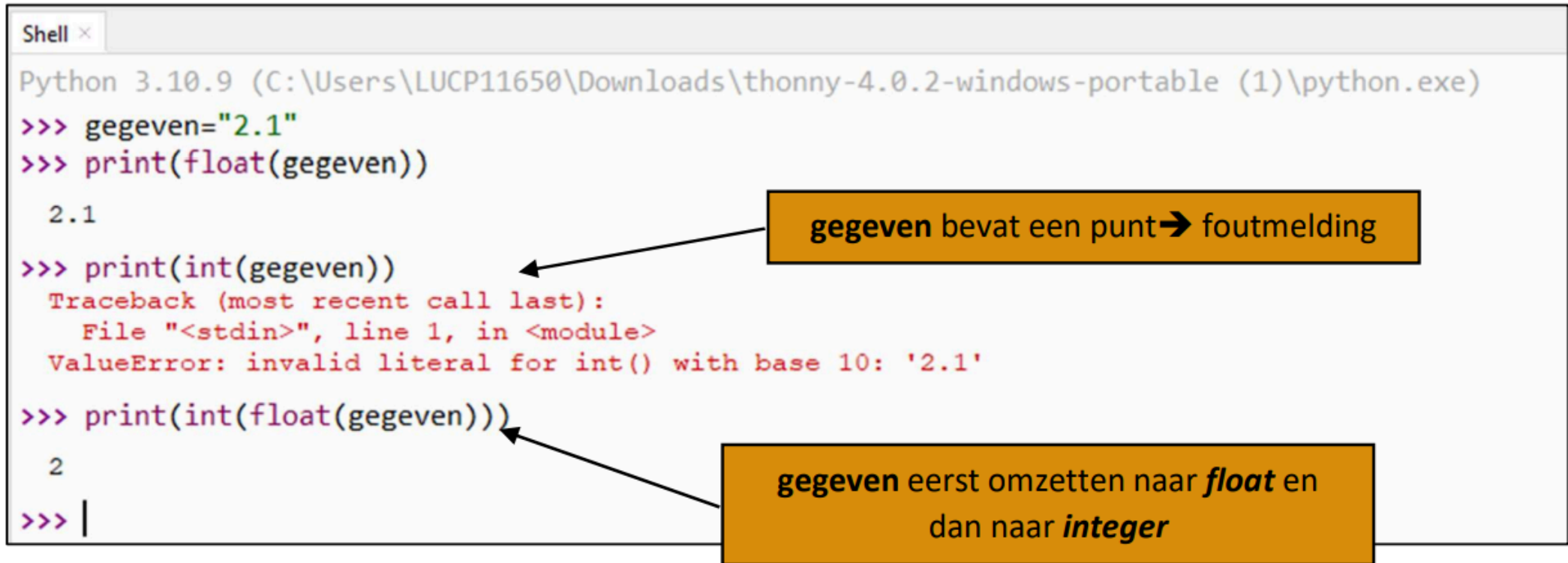
```
decimaal_getal = float(gegeven)
```

Converteren naar een string:

```
tekst = str(gegeven)
```

foutjes?

Als Python niet in staat is om het gegeven te converteren, dan krijg je een foutmelding.



The screenshot shows a Python 3.10.9 shell window. The code executed is as follows:

```
Python 3.10.9 (C:\Users\LUCP11650\Downloads\thonny-4.0.2-windows-portable (1)\python.exe)
>>> gegeven="2.1"
>>> print(float(gegeven))
2.1
>>> print(int(gegeven))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '2.1'
>>> print(int(float(gegeven)))
2
>>> |
```

Two orange callout boxes with arrows pointing to the code:

- The first box points to `print(int(gegeven))` and contains the text: **gegeven** bevat een punt → foutmelding
- The second box points to `print(int(float(gegeven)))` and contains the text: **gegeven** eerst omzetten naar *float* en dan naar *integer*

Opdracht: pas je kennis toe, werken als een pro!

Open je script "bussen"

Pas je geschreven programma in Thonny aan.

Aantal personen : vragen om in te geven

Capaciteit van één bus: 52

Je programma moet het aantal volle bussen dat nodig is en het aantal personen dat overblijft (die niet meer in een volledige bus passen) berekenen.

Gebruik input() om de nodige gegevens op te vragen en int() om die gegevens naar een bruikbaar geheel getal om te zetten.

Gebruik # voor commentaarregels.

Pas je script nog eens aan:

laat de gebruiker ook de capaciteit van een bus in te geven.

mogelijke oplossing "bussen" met input() en int()

```
1 # Voorbeeldcode "bussen":  
2 aantal_personen = int(input("aantal personen: "))  
3 buscapaciteit = int(input("aantal personen per bus: "))  
4 volle_bussen = aantal_personen // buscapaciteit  
5 overblijvend = aantal_personen % buscapaciteit  
6 print("Aantal volle bussen:", volle_bussen)  
7 print("Personen zonder plaats in een bus:", overblijvend)
```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
aantal personen: 347
```

```
aantal personen per bus: 47
```

```
Aantal volle bussen: 7
```

```
Personen zonder plaats in een bus: 18
```

```
>>> |
```

deel 6

oefenen in Thonny:

1. operatoren
2. "oppervlakte cirkel" deel 1 en deel 2
3. "snoepautomaat"
4. "korte scripts"

oefening 1 "Operatoren"

1. Bereken 5 tot de macht 3
2. Bereken de vierkantswortel van 64 (gebruik $(1/2)$ -macht)
3. Wat is de uitkomst van $17 // 4$?
4. Wat is de rest van $17 \% 4$?
5. Maak een zin met een naam via concatenatie (+)
6. Bereken een vierkantswortel van 625 (gebruik $(1/2)$ -macht)

Oefening 2: "oppervlakte cirkel" deel 1

De oppervlakte van een cirkel met straal r is gelijk aan $r^2 \times \pi$. Daarbij is $\pi=3.141592653589793$.

Opdracht

Onderstaand codefragment kent de straal van een cirkel toe aan de variabele `straal`. De waarde van π wordt toegekend aan de variabele `pi`.

Gevraagd :

Schrijf een programma dat de oppervlakte van de cirkel berekent en die aan de variabele `oppervlakte` toekent.

een stukje code om te starten:

```
# straal van cirkel
straal = 2.73
# waarde van  $\pi$ 
pi = 3.141592653589793
```

Oefening 2: "oppervlakte cirkel" deel 2

Gevraagd deel 2 :

Shrijf een programma dat:

- aan de gebruiker een straal van een cirkel vraagt
- de oppervlakte van de cirkel berekent
- het resultaat aan de variabele oppervlakte toekent en afdrukt

gebruik

`input()`

`float()`

`print()`

"oefening 3: "snoepautomaat"

Een snoepje uit de automaat kost 25 cent.

Schrijf een programma (script) dat:

1. De gebruiker vraagt om een bedrag in euro in te voeren.
2. Het aantal snoepjes berekent dat de gebruiker kan kopen en hoeveel eurocent hij/zij terugkrijgt.
3. De resultaten aan de gebruiker weergeeft.

Tips:

- maak eerst een algoritme (schema, volgorde...)
- vertaal daarna je algoritme naar Python
- schrijf je script in Thonny
- test je volledige programma.

oefening 4 "korte scripts schrijven"

1. Schrijf een programma dat naar een x-waarde vraagt, de functie $f(x) = x^2 - x - 2$ berekend en de bijbehorende functiewaarde afdruckt.
2. Schrijf een programma dat een ingegeven waarde in graden Celsius omzet (berekent) naar graden Fahrenheit en het resultaat afdruckt. De omzettingsformule is $^{\circ}\text{F} = (^{\circ}\text{C} \times 9/5) + 32$ waarbij $^{\circ}\text{C}$ het aantal graden Celsius en $^{\circ}\text{F}$ het aantal graden Fahrenheit voorstelt.
3. Schrijf een programma dat naar je geboortejaar vraagt, je leeftijd berekent en afdruckt. Hier wordt er geen rekening mee gehouden of je dit jaar al jarig bent geweest.
4. Schrijf een programma dat naar je Voornaam vraagt en "Hallo, (je Voornaam), je bent geweldig!" weergeeft.
5. Schrijf een programma dat naar een aantal gelopen kilometers van de 3 parcours en naar de tijd per gelopen parcours vraagt, de gemiddelde snelheid (km/uur) van alle gelopen kilometers berekent en weergeeft.

deel 7

test jezelf:

- beantwoord de volgende vragen zonder hulp

juiste antwoorden zal je op een apart blad vinden

In welk werkvenster van Thonny voer je de code in?

Typ h

A

Editor

Assistant

B

C

Shell

CLI

D

Hoe plaats je tekst (string) in de code?

A

tussen ronde haakjes

tussen enkele
aanhalingstekens

B

C

achter #

tussen dubbele
aanhalingstekens

D

Wat is de benaming voor een bug waarbij de tekens niet in de juiste volgorde staan of ontbreken?

A

syntax bug

code bug

B

C

syntax error

code error

D

Welke van deze variabelenamen is correct volgens de Python-syntax?

A

1ste_leerling

eerste-leerling

B

C

leerling 1

leerling_1

D

Zal deze concatenatie werken?

```
1 a=1  
2 b="ba1"  
3 print(a+b)
```

A

JA

NEE

B

C

Ik weet het niet

Welk teken moet je plaatsen voor een commentaar in je script?

A

&

@

B

C

()

#

D

Welke van deze casting functies is geen
getaltype functie

A

str()

float()

B

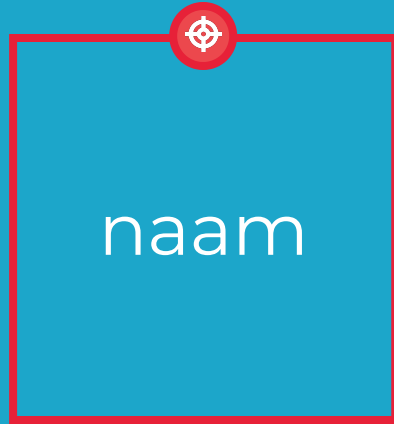
C

int()

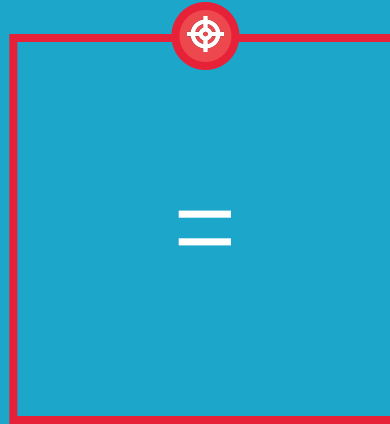
geen idee

D

sleepvraag



naam



=



"Kuifje"



waarde



variabele



toekennen

klas1 = 24
is een voorbeeld van...

A

concatinatie

expressie

B

C

assignment statement

sequentie

D

datatype van alle decimale getallen

A

float

string

B

C

integer

bool

D

Dit _ (streepje) betekent...

A

underscore, dit moet in Python gebruikt worden ipv ALLE spaties

koppelteken, voor alle samengestelde woorden

B

C

underscore, dit moet in Python gebruikt worden wanneer de spaties niet toegelaten zijn

streepje om iets te onderstrepen

D

```
a = "3"  
b = 2  
c = a * b  
print(c)
```

OUTPUT?

A

6

33

B

C

foutmelding

222

D

een comando dat het resultaat, van een programma in Python, zichtbaar maakt op het scherm

A

`input()`

output

B

C

`str()`

`print()`

D

datatype van alle gehele getallen

A

float

integer

B

C

string

bool

D

```
a = 7  
a = 3 + a  
b = 6  
num = a * b  
print(num)
```

OUTPUT?

A

60

syntax error

B

C

42

18

D

om de gegevens te kunnen ontvangen of inlezen gebruiken we de functie...

A

`int()`

`print()`

B

C

`str()`

`input()`

D

$a / 5$
is een voorbeeld van...

A

expressie

assignment statement

B

C

concatination

input

D

datatype van alle tekens en teksten

A

string

float

B

C

integer

bool

D

```
a = 5  
B = 7  
c = a + b  
print(c)
```

OUTPUT?

A

12

9

B

C

foutmelding

7

D

```
num_1 = 2
num_2 = 5
num_3 = 8
num_1 = num_2 + num_3 - num_1
num_2 = num_1 + num_3
print(num_1)
OUTPUT?
```

A

19

10

B

C

5

12

D

In de Shell zie je "Syntax error" staan, dit betekent:

A

foutmelding, computer is
vastgelopen

foutmelding, mijn script bevat
onbekende fouten

B

C

foutmelding, mijn script is juist, app
doet raar

foutmelding, mijn script bevat
schrijffouten

D



4BW

wordt vervolgd...