

# Python Programming

## Variabelen en basisgeeventypes

**Kristof Michiels**

# ! Disclaimer

- Dit is een alfa-versie van de cursus die zal gebruikt worden voor het vak Python Programming in semester 2
- Deze informatie is dus mogelijk nog onvolledig en kan nog onzorgvuldigheden of zelfs fouten bevatten

# Lesinhoud

# Lesinhoud

- Variabelen
  - Wat zijn ze?
  - Naamgeving
- Commentaar in je code
- Basisgeeventypes
  - Werken met tekst: strings en witruimte
  - Werken met getallen: integers en floats

# Variabelen

# Basisvoorbeeld variabelen

```
boodschap = "Welkom bij Python programming!"  
print(boodschap)  
  
boodschap = "Python leren programmeren behoort tot het mooiste wat er bestaat"  
print(boodschap)
```

- We maken een variabele aan en noemen ze "*boodschap*"
- Elke variabele is verbonden met een waarde. Dit is de informatie die met de variabele wordt geassocieerd
- Je kan de waarde van een variabele op elk moment wijzigen in je programma. Python zal steeds de huidige waarde van de variabele bijhouden en kunnen weergeven

# Basisvoorbeeld variabelen

```
boodschap = "Welkom bij Python programming!"  
print(boodschap)  
  
boodschap = "Python leren programmeren behoort tot het mooiste wat er bestaat"  
print(boodschap)
```

- Variabelen zijn labels waaraan je een waarde kan toekennen
- Een variabele "verwijst" dan naar een bepaalde waarde, veeleer dan dat het een doosje is waarin je een bepaalde waarde opslaat

# Naamgeving variabelen

- Gebruik korte, beschrijvende namen
- Vermijd het gebruik van hoofdletters
- Namen van variabelen kunnen enkel bestaan uit letters, getallen en *underscores* ("\_" )
- De naam mag starten met een *underscore*, maar niet met een getal
- Spaties zijn niet toegestaan. Maak eventueel gebruik van *underscores* om woorden van elkaar te scheiden
- Vermijd het gebruik van Python *keywords* als namen voor variabelen. Dit zijn gereserveerde woorden (zie volgende slide)



# Gereserveerde woorden in Python

```
help("keywords")
```

Here is a list of the Python keywords. Enter any keyword to get more help.

False	break	for	not
None	class	from	or
True	continue	global	pass
__peg_parser__	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield

# Meerdere toewijzen

```
voornaam, familienaam, beroep = "Kristof", "Michiels", "Docent"
```

- Je kan waarden aan meerdere variabelen toekennen in één enkele statement
- Dit maakt je programma's korter en beter leesbaar
- Je scheidt hiervoor de namen van de variabelen met komma's en doet hetzelfde met de waarden
- Python kent dan de respectievelijke waarde toe aan elke genoemde variabele

# Constanten

```
STUDENTEN_PER_GROEP = 4
```

- Een constante is zoals een variabele, maar met een waarde die niet verandert tijdens de looptijd van je programma
- Python heeft geen ingebouwde ondersteuning voor constanten maar developers gebruiken een naam bestaande uit enkel hoofdletters om aan te geven dat het hier over een niet te veranderen constante gaat

# Commentaar

# Commentaar

- Commentaar laat toe om zelfgeschreven notities toe te voegen aan je programma's
- In Python gebruiken we een hashtag ("#") aan het begin van een lijn commentaar om aan te geven dat we hiermee te maken hebben
- We geven hiermee aan de interpreter mee dat wat volgt niet hoeft geïnterpreteerd te worden

```
# Een typisch "Hallo wereld"-voorbeeld  
boodschap = "Welkom bij Python programming!"  
print(boodschap)
```

# Basisgevenstypes

# Werken met tekst: strings

- Een string bestaat uit een reeks tekst-karakters
- Alles binnen aanhalingstekens wordt beschouwd als een string
- Je mag gebruik maken van enkele of dubbele aanhalingstekens
- Deze flexibiliteit laat toe om aanhalings- en afkappingstekens te gebruiken in je strings
- Wees consistent in je keuzes

```
boodschap = "Dit is een string."  
boodschap = 'Dit is ook een string.'  
boodschap = 'Mijn vriend vroeg: "Programmeer jij ook in Python?" '  
boodschap = "Ik hou van het schrijven van Python programma's"
```

# Hoofdlettergebruik wijzigen in een string

```
naam = "Guido van Rossum"  
print(naam.title()) # Elk woord laten beginnen met een hoofdletter: Guido Van Rossum  
print(naam.upper()) # Alles in hoofdletters: GUIDO VAN ROSSUM  
print(naam.lower()) # Alles in kleine letters: guido van rossum
```

- We beschikken hiervoor over een aantal zgn. methods. Dit zijn acties die je op bepaalde data kan toepassen
- De dot-notatie (".") vertelt Python in het eerste vb. om de title-method uit te voeren op de variabele naam
- Elke method wordt gevolgd door haakjes, omdat methods vaak bijkomende informatie nodig hebben om hun werk te kunnen doen. Hier is dit evenwel niet het geval
- De lower()-method wordt vaak gebruikt om data op te slaan die door een gebruiker werd ingegeven



# Variabelen gebruiken in een string: f-strings

```
voornaam = "Guido"  
familienaam = "van Rossum"  
volledige_naam = f"{voornaam} {familienaam}"  
boodschap = f"Bedankt voor Python, {volledige_naam.title()}!"  
print(boodschap)
```

- Om variabelen te gebruiken binnen strings plaats je de letter f onmiddellijk voor het eerste aanhalingsteken
- Je omringt de variabelen die je in de string gebruikt met accolades
- Python vervangt elke variabele door de waarde
- De f in f-strings staat voor "format"

# Witruimte toevoegen aan strings

```
print("\tHelderheid")
print("Pythonisch programmeren:\nHelder\nGeloofwaardig\nEfficiënt")
print("Pythonisch programmeren:\n\tHelder\n\tGeloofwaardig\n\tEfficiënt")
```

- Met witruimte doelen we op niet-afdrukbare tekens als spaties, tabs en einde-lijnsymbolen
- Je gebruikt ze om je output op een beter leesbare manier te organiseren
- Een tab-insprong toevoegen aan je tekst doe je met "\t"
- Een nieuwe regel voeg je toe met "\n"
- Combinaties zijn mogelijk: "\n\t" zorgt ervoor dat Python op een nieuwe regel begint, met een tabinsprong

# Witruimte elimineren binnen strings

```
boodschap = " Ik hou van Python "  
print(boodschap.rstrip())  
print(boodschap.lstrip())  
print(boodschap.strip())
```

- Python maakt het eenvoudig om (eventueel) aanwezige witruimte te verwijderen
- Er kan een onderscheid worden gemaakt tussen witruimte links, rechts of aan beide zijden van een string
- Handig als je twee strings met elkaar wil vergelijken

# Werken met gehele getallen (integers)

```
getal1 = -2  
getal2 = 3  
print(getal1 + getal2) # 1  
print(getal1 ** getal2) # -8  
print((getal1 + getal2) * getal1) # -2
```

- Integers kunnen in Python opgeteld (+), afgetrokken (-), vermenigvuldigd (\*) en gedeeld (/) worden
- Python gebruikt twee vermenigvuldigingssymbolen (\*\*) om exponenten weer te geven
- Elke expressie kan meerdere bewerkingen bevatten. De volgorde van bewerkingen wordt gerespecteerd
- Je mag gebruik maken van haakjes om deze volgorde aan te passen

# Werken met decimale getallen (floats)

```
getal1 = 0.4  
getal2 = 0.2  
print(getal1 + getal2) # 0.60000000000000001  
print(getal1 * 2) # 0.8
```

- Python noemt elk kommagetal een float
- Deze term wordt in de meeste programmeertalen gebruikt en verwijst naar het feit dat een decimaalteken op elke positie in een getal kan voorkomen
- Houd er rekening mee dat je soms een willekeurig aantal decimalen in je antwoord kunt krijgen

# Gehele en decimale getallen

```
print(9 / 3) # 3.0
print(13 + 2.0) # 15.0
print(4 * 5.0) # 20.0
print(2.0 ** 3) # 8.0
totaal_aantal_aardbewoners = 7_902_193_151
print(totaal_aantal_aardbewoners) # 7902193151
```

- Wanneer je twee getallen deelt is het resultaat altijd een float
- Python gebruikt standaard een float als resultaat voor elke bewerking die een float bevat, zelfs als de uitvoer een geheel getal is
- Bij grote getallen kun je cijfers groeperen met underscores om ze leesbaarder te maken

# Python Programming - les 1 - [kristof.michiels01@ap.be](mailto:kristof.michiels01@ap.be)