IT Essentials

Hoofdstuk 2

Variabelen

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Inhoud

1. Inleiding
2. Variabelen en waarden
3. Variabele namen
    1. Conventies
    2. Oefenen met variabele namen
    3. Constanten
4. Debuggen met variabelen
5. Soft typing
6. Verkorte operatoren
7. Commentaar
8. Input

2.1 Inleiding

- zie opgave 1.1: bereken prijs van 60 boeken
  ➔ bereken prijs 73 of 555 of … boeken: Hoe?

- je hebt nood aan variabelen waarin de verschillende waardes achtereenvolgens opgeslagen kunnen worden

- code = je ontwerpt algoritme dat probleem op een **algemeen** toepasbare manier oplost

2.2 Variabelen en waarden

- variabele = plaats in geheugen met een naam en waar je een waarde in kan stoppen

```
number1 = 5
number2 = 6
total = number1 + number2
```

number1	5
number2	6
total	11

- = is ‘assignment’ operator

- Voorbeelden: goed of fout?

```
variable = 5
print(variable)
variable = "IT Essentials is leuk"
print(variable)
variable = int(5 / 4) + 7
print(variable)
```

= goed

```
b = 4
c = b
```

Code is goed, maar niet volgens de code conventies! Best betekenisvolle namen!

```
value = 8
value = value + 1
```

= goed

```
print(result)
result = 7
```

= fout!

2.3 Variabelennamen

2.3.1 Conventies

- enkel letters, cijfers en underscores (_)
- moet beginnen met letter of underscore
- mag geen gereserveerd woord zijn

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

- case sensitive
- conventies: betekenisvol / beginnend met letter / allemaal kleine letters / _ tussen woorden

- Voorbeeld (leesbaar?)

```
a = 3.14159265
b = 7.5
c = 8.25
d = a * b * b * c / 3
print(d)
```

Geen betekenisvolle namen!

Beter:

```
pi = 3.14159265
radius = 7.5
height = 8.25
volume_of_cone = pi * radius * radius * height / 3
print(volume_of_cone)
```

2.3.2 Oefenen met variabelennamen

- Voorbeeld (goed of fout?)

classification = 1	#1
Classification = 1	#2
cl@ssification = 1	#3
Class1f1cation = 1	#4
8classification = 1	#5
_classification = 1	#6
class = 1	#7
Class = 1	#8

2.3.3 Constanten

- waarde kan niet meer gewijzigd worden in meeste programmeertalen ⟺ in Python kan dit wel!

- conventie: hoofdletters met _ tussen woorden

- Voorbeeld

```
VAT_FACTOR = 1.21

total = 24.95
total_vat_inclusive = total * VAT_FACTOR
print(total_vat_inclusive)
```

2.4 Debugging met variabelen

- door het toevoegen van extra print()-statements

- voorbeeld

```
value1 = 5
value2 = 4
value3 = 5
print(value3 / (value1 % value2))
value1 = value1 + 1
print(value3 / (value1 % value2))
value1 = value1 + 1
print(value3 / (value1 % value2))
value1 = value1 + 1
print(value3 / (value1 % value2))
```

2.5 Soft typing

- soft typing: een variabele heeft geen vast type
- type van variabele opvragen: functie type()

```
value = 8
print(type(value))
value = "Maandag"
print(type(value))
```

➔ <class 'int'>

➔ <class 'str'>

- hard typing: data type van een variabele wordt bepaald bij declaratie van variabele en kan niet meer wijzigen (vb Java, C++)
- operatoren passen zich aan aan type van variabele:

```
value1 = 1
value2 = 4
value3 = "1"
value4 = "4"
print(value1 + value2)
print(value3 + value4)
```

➔ ?

➔ ?

2.6 Verkorte operatoren

Verkorte operator	Gebruik	Gelijk aan	Betekenis
+=	a += b	a = a + b	optelling
-=	a -= b	a = a - b	aftrekking
*=	a *= b	a = a * b	vermenigvuldiging
/=	a /= b	a = a / b	deling
//=	a //= b	a = a // b	gehele deling
%=	a %= b	a = a % b	modulo
**=	a **= b	a = a ** b	machtsverheffing

2.6 Verkorte operatoren

- voorbeeld

```
number_of_bananas = 100
number_of_bananas += 12
number_of_bananas -= 13
number_of_bananas *= 19
number_of_bananas /= number_of_bananas
print(number_of_bananas)
```

2.7 Commentaar

- = tekst die Python negeert tijdens uitvoering
- wordt gebruikt om code te verduidelijken
- #: 1 regel commentaar
- Voorbeeld

```
# calculate discriminant
a = 1
b = 2
c = 3
d = b * b - 4 * a * c
```

- 3 aanhalingstekens: meerdere regels commentaar, rekening houdend met indentatie
- Voorbeeld:

```
"""Write here your comment ... comment ...
comment... Multi-line comment serves as
documentation for others reading your code."""
euros = 10
sentence = "De totale kost is"
...
```

2.8 Input

- gebruiker geeft data (gegevens) in
- functie: input()
- voorbeeld:

```
text = input("Geef een tekst in: ")
print("Je hebt het volgende ingetypt:", text)
```

text = variabele van type string

- ander type input? ➔ casting

```
number = int(input("Geef een getal in: "))
number_square = number * number
print(number_square)
```