


[Man Chain illustration](#) by [Frits Ahlefeldt-Laurvig](#) | CC BY-ND

Week 3: Exceptions | Doctests

Programming (TICT-V1PROG-15)
HBO-ICT propedeuse periode 1 2015-2016



Feed up: Wat gaan we deze les leren?

- Zelfstudie
 - Operators
 - Docstrings
- Exceptions
 - Foutafhandeling
 - Try / Except / Finally
- Testing & Debugging
 - Software ontwikkelproces
 - Testing
 - Debugging



Herhaling: Wat is de uitvoer?

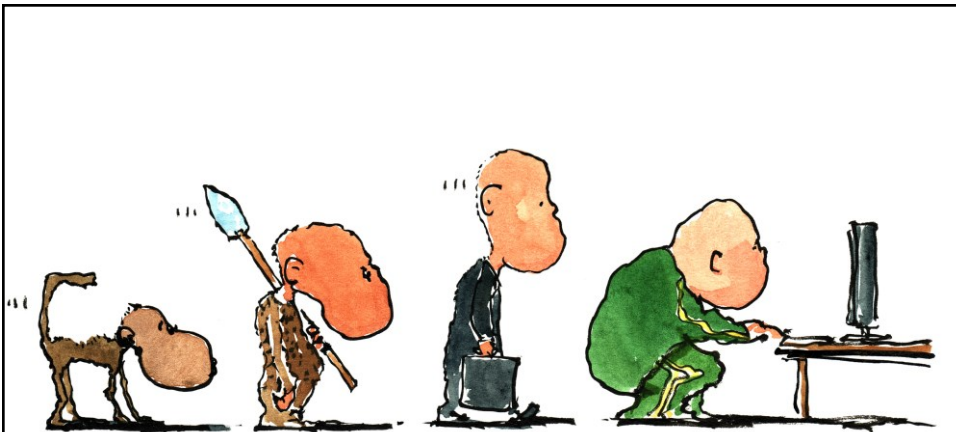
- `a="helloworld";`
`print(a[-1])`
`print(a[3])`
`print(a[3:5])`
- `l=[3,4,4,5];`
`print(len(l)<sum(l) and l.count(4)==2)`



Herhaling: Wat is de uitvoer?

- `print(49//8==6 and 49%8<9)`
- `for i in range(1,100,25): print(i)`
- `i=3; i*=6; print(i)`





Man Chain illustration by Frits Ahlefeldt-Laurvig | CC BY-ND

Exceptions

Programming (TICT-V1PROG-15)
HBO-ICT propedeuse periode 1 2015-2016

**HOGESCHOOL
UTRECHT**

Exceptions: Foutafhandeling

```
bestand='bestaat_niet.txt'
f=open(bestand)
print("Einde...")

S_a
C:\Python34\python.exe C:/Users/Administrator/PycharmProjects/les5/les5_a.py
Traceback (most recent call last):
  File "C:/Users/Administrator/PycharmProjects/les5/les5_a.py", line 2, in <module>
    f=open(bestand)
FileNotFoundError: [Errno 2] No such file or directory: 'bestaat_niet.txt'

Process finished with exit code 1
```

- Oei: Het programma stopt, de ontwikkelaar heeft geen controle meer.

Exceptions: Oplossing 1

```
import os

bestand = 'bestaat_niet.txt'

if not os.path.exists(bestand):
    print('Het bestand', bestand, 'bestaat niet.')
else:
    f = open(bestand)
```

- Probeer fouten in je code af te vangen



Exceptions: Oplossing 2

```
import os

bestand = 'bestaat_niet.txt'

try:
    f = open(bestand)
except:
    print(bestand, 'kan niet geopend worden.')
```

- EAFP-principe:
Easier to Ask Forgiveness than Permission
- Hiermee vang je nog meer fouten af. Welke bijvoorbeeld?



Exceptions: Oplossing 3

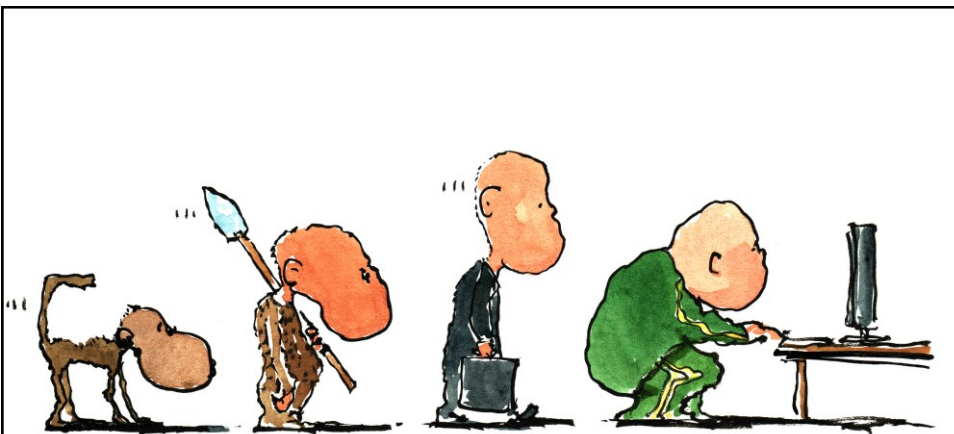
```
import sys
bestand = 'bestaat_niet.txt'
f = None
try:
    f=open(bestand,'r')
    for line in f:
        print(line,end='')

except EnvironmentError as error_info:
    print(bestand,'kan niet geopend worden:',error_info)

except: # alle overige fouten
    print("Oeps, een andere fout:",sys.exc_info())

finally:
    try:
        f.close()
    except:
        print('Bestand kan niet worden gesloten')

print('\nProgramma gaat verder.')
```



Man Chain illustration by Frits Ahlefeldt-Laurvig | CC BY-ND

Testing

Programming (TICT-V1PROG-15)
HBO-ICT propedeuse periode 1 2015-2016

HU HOGESCHOOL
UTRECHT

UTRECHT

Testing: The Software Development Process

- Bij het schrijven van software worden verschillende fases doorlopen:

1. Wat	(Analyse)
2. Hoe	(Design)
3. Doen	(Implementatie: Maken en Debuggen)
4. Test	(Testen)
5. Gebruiken	(Operatie/Deployment)

11



Testing & Debuggen:

- Testen:
Controleer of de resultaten overeenkomen met de verwachting
- Debuggen:
Zoek de oorzaken voor onverwacht "gedrag" en repareer deze.

12



Testing

- Er bestaan verschillende soorten testen. Ontwikkelaars maken vaak "unit tests" waarmee ze kleine delen van de code testen.
- Dit is vooral belangrijk als je later iets wilt veranderen aan je code.
- Met de module doctest of unittest is dit mogelijk.
- Doctest is het meest eenvoudig in gebruik en voor het maken van de tests.



Testing: doctest, een tutorial

1. Schrijf een component, bv een functie.
2. Test de functie via een "Interactive Python sessie".
3. Kopieer de tests en de resultaten naar je functie in de docstring.
4. Best Practice: Maak je module met functies en zorg ervoor dat de tests worden uitgevoerd als je de module "start".



Testing: doctest, step-by-step

1. Schrijf een component, bv een functie.

```
def verdubbel(a):
    """ Verdubbel de input en geef deze waarde terug.
    """
    return a+a
```



Testing: doctest, step-by-step

2. Test de functie via een "Interactive Python sessie".

```
Python Console
>>> from calculation import *
>>> verdubbel(1)
2
>>> verdubbel(10)
20
>>> verdubbel(0)
0
>>> verdubbel(-10)
-20
>>> verdubbel(9999999999)
19999999998
>>> verdubbel(9999999999,44)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: verdubbel() takes 1 positional argument but 2 were given
>>> verdubbel(9999999999.44)
19999999998.88
>>>
```



Testing: doctest, een tutorial

3. Kopieer de tests en de testresultaten naar je functie in de docstring.

```

1 def verdubbel(a):
2     """ Verdubbel de input en geef deze waarde terug.
3
4     >>> verdubbel(1)
5         2
6     >>> verdubbel(10)
7         20
8     >>> verdubbel(0)
9         0
10    >>> verdubbel(-10)
11        -20
12    >>> verdubbel(9999999999)
13        19999999998
14    >>> verdubbel(9999999999,44)
15        Traceback (most recent call last):
16          File "<input>", line 1, in <module>
17            TypeError: verdubbel() takes 1 positional argument but 2 were given
18    >>> verdubbel(9999999999.44)
19        19999999998.88
20    """
21    return a+a

```

Testing: doctest, een tutorial

4. Voeg de onderstaande twee regels toe en kijk of het werkt...

```

16    TypeError: verdubbel() takes 1 positional argument but 2 were given
17    >>> verdubbel(9999999999.44)
18        19999999998.88
19    """
20    return a+a
21
22    import doctest
23    doctest.testmod()
24

```

Run Calculation

C:\Python34\python.exe C:/Users/Administrator/PycharmProject/...
Process finished with exit code 0

Testing: doctest, een tutorial

5. Best Practice:

- Maak een module met jouw functies.
- Voeg testen toe aan de module via een interactieve sessie.
- Voeg de onderstaande regels toe onderaan je module:

```
if __name__ == "__main__":  
    import doctest  
    doctest.testmod()
```

- Importeer je module vanuit een ander script.
- Test je module door deze "te runnen"...



Meer over Exceptions:

- Perkovic: Chapter 4.4 & 7.3 (Exceptions)



Opdracht

Maak thuis de opdrachten die horen bij:

- Week 3: Expressies en Documentatie | College 1

21

 HOGESCHOOL
UTRECHT