



MuleSoft®

Module 2: Python

Opdrachten

Inhoudsopgave

Inhoudsopgave	2
LAB 2.1: MAAK JE EERSTE PROGRAMMA IN PYTHON	5
Start de MU programmeeromgeving	5
Voer Python code in	5
Voer de Python code uit	6
LAB 2.2: EEN PYTHON PROGRAMMA UITVOEREN VIA DE TERMINAL.....	7
Open een terminal window (opdrachtprompt)	7
Voer het Python programma uit	7
LAB 2.3: DE PYTHON OMGEVING LEREN KENNEN	9
Open een terminal window (opdrachtprompt)	9
Start de interactieve Python omgeving	9
Vraag hulp bij het gebruik van de interactieve omgeving	9
Vraag hulp bij een specifiek commando	11
Sluit de interactieve omgeving af	11
Bekijk de Python installaties	12
LAB 2.4: MAAK EEN START MET HET REKENPROGRAMMA	14
Start de MU programmeeromgeving	14
Voer de eerste regels code in	14
Test het programma	15
Verander het data type van de invoer naar numeriek	15
Maak een verzameling van mogelijk munten	16
Loop door de verzameling	16
Voeg de berekeningen toe	17
Voeg commentaar toe om de code te verduidelijken	18
Toon de berekening	19
Voer het programma uit	19
LAB 2.5: VERBETER HET REKENPROGRAMMA	21
Maak een nieuw bestand aan	21
Kopieer de code van gepast1.py	21
LAB 2.6: BREID HET REKENPROGRAMMA UIT	23
Maak een nieuw bestand aan	23

Kopieer de code van gepast2.py.....	23
Gebruik andere data types om met Euro's te kunnen werken	23
Leer code te verkorten	24
Voeg nieuwe functionaliteit toe: totaal aantal munten.....	24
Controleer je werk	24
Voer het programma uit	25

Module 2

In deze module:

- Ga je werken op je virtuele leer omgeving
- Maak je je eerste programma in Python
- Leer je een Python programma uit te voeren
- Ontdek je verschillende functies en mogelijkheden van Python

Lab 2.1: Maak je eerste programma in Python

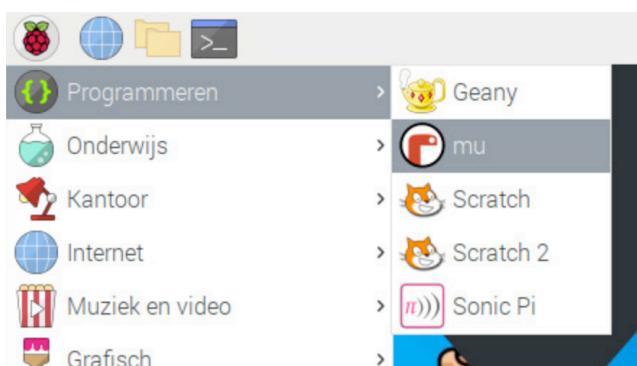
Dit lab laat zien hoe je op je eigen Virtuele Leer Omgeving kunt werken met Python.

Alle wijzigingen aan de omgeving blijven bewaard, en alle bestanden die je aanmaakt blijven bestaan.

Je kunt de omgeving blijven gebruiken tot 30 december 2021.

Start de MU programmeeromgeving

1. Klik op het Raspberry icoon op de menubalk.
2. Selecteer **Programmeren**.
3. Selecteer **mu**.

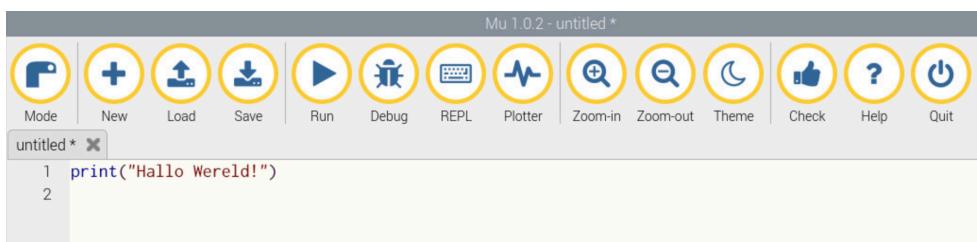


4. Wacht een ogenblik tot het beginscherm van Mu verdwijnt.

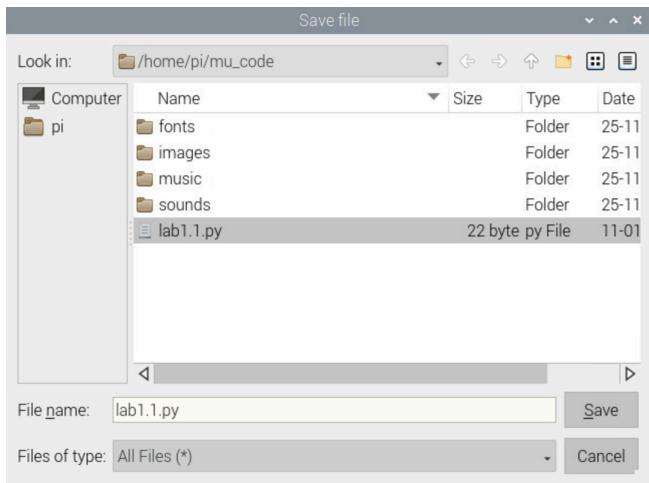
Voer Python code in

5. Verwijder de eerste regel: # Write your code here :-)
6. Voer de volgende regel code in:

```
print("Hallo Wereld!")
```



7. Klik in de menubalk op **Save**.
8. Nu verschijnt een dialoogvenster.
9. Geef bij "File name:" de volgende bestandsnaam op: **lab1.1.py**
10. Klik op **Save**.

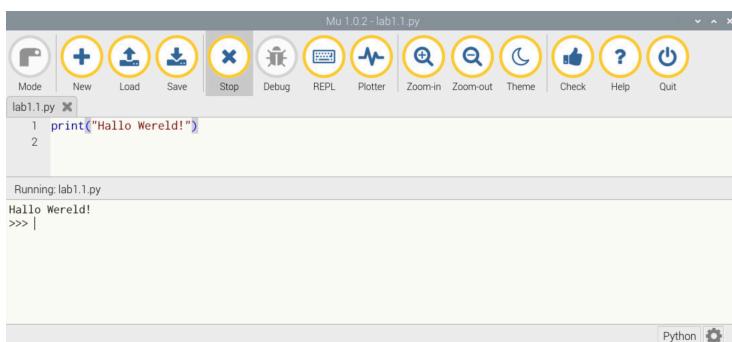


11. De code is nu opgeslagen.

Voer de Python code uit

12. In de menubalk, klik op **Run**.

13. De code wordt nu uitgevoerd; de tekst "Hallo Wereld!" verschijnt onder in beeld.



14. Klik op **Stop** in de menubalk.

15. Klik op **Quit** in de menubalk.

Lab 2.2: Een Python programma uitvoeren via de terminal

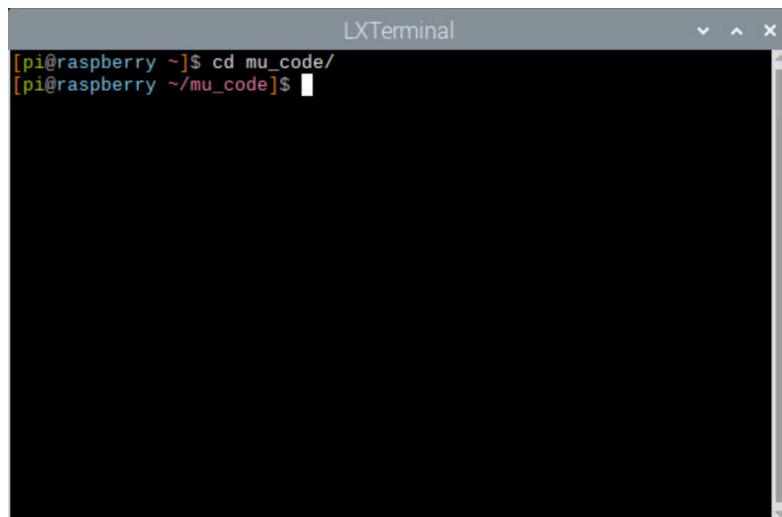
In dit lab leer je een andere manier om een Python programma uit te voeren, deze keer niet vanuit de programmeeromgeving (Mu), maar vanaf een zogenaamde opdrachtprompt.

Je gaat het programma uit lab 1.1 gebruiken om uit te voeren. Zorg er dus voor dat je dit lab hebt afgerond.

Open een terminal window (opdrachtprompt)

1. Klik op het Raspberry icoon op de menubalk.
2. Selecteer **Hulpmiddelen**.
3. Selecteer **LXTerminal**.
4. Wanneer de opdrachtprompt wordt getoond, type het volgende commando en druk op ENTER:

```
cd mu_code/
```



The screenshot shows a terminal window titled "LXTerminal". In the terminal, the user has typed the command `[pi@raspberry ~]$ cd mu_code/`. The cursor is visible at the end of the command line. The background of the terminal is black, and the text is white.

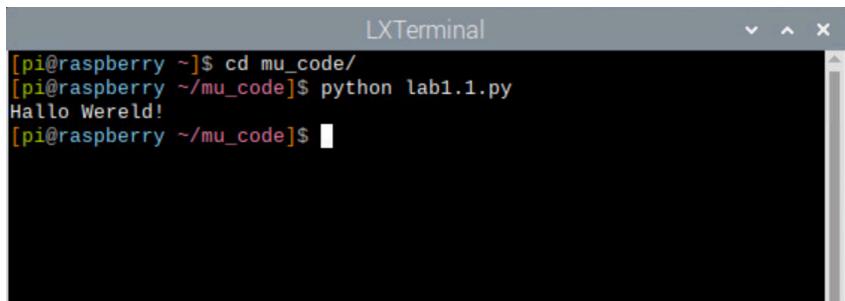
5. Je bevindt je nu in de directory `~/mu_code`, waar je in het vorige lab het Python programma hebt opgeslagen.

Voer het Python programma uit

6. Type het volgende commando in:

```
python lab1.1.py
```

7. Let op de uitvoer op het scherm; dit moet zijn **Hallo Wereld!**.



The screenshot shows an LXTerminal window with a dark background and light-colored text. At the top, it says "LXTerminal". The terminal session starts with "[pi@raspberry ~]\$ cd mu_code/", followed by "[pi@raspberry ~/mu_code]\$ python lab1.1.py", then "Hallo Wereld!", and finally "[pi@raspberry ~/mu_code]\$". There are small icons for closing, minimizing, and maximizing the window in the top right corner.

```
[pi@raspberry ~]$ cd mu_code/
[pi@raspberry ~/mu_code]$ python lab1.1.py
Hallo Wereld!
[pi@raspberry ~/mu_code]$
```

8. Type het volgende commando in:

```
exit
```

9. De opdrachtprompt wordt nu afgesloten.

Lab 2.3: De Python omgeving leren kennen

In dit lab duik je wat dieper in de installatie van Python. Je leert hoe je het versienummer van Python kunt opvragen, en hoe je kunt zien waar Python geïnstalleerd is op de Raspberry Pi. Tenslotte leer je de interactieve Python omgeving gebruiken.

Open een terminal window (opdrachtprompt)

1. Klik op het Raspberry icoon op de menubalk.
2. Selecteer **Hulpmiddelen**.
3. Selecteer **LXTerminal**.

Start de interactieve Python omgeving

4. Type het volgende commando in:

```
python --version
```

Observeer hoe het versienummer van de geïnstalleerde Python versie wordt getoond.

```
[pi@raspberry ~]$ python --version
Python 2.7.16
[pi@raspberry ~]$
```

5. Type nu het volgende commando:

```
python
```

Je komt nu in een interactieve Python omgeving, waar je bepaalde commando's kunt ingeven. Dit zijn Python commando's, en dus niet dezelfde commando's als in de opdrachtprompt.

```
[pi@raspberry ~]$ python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Vraag hulp bij het gebruik van de interactieve omgeving

6. Type het volgende commando:

```
help()
```

Nu verschijnt een interactieve hulpfunctie, waar commando's kunnen worden ingevoerd om meer informatie hierover te verkrijgen.

```
[pi@raspberry ~]$ python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> help()

Welcome to Python 2.7! This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/2.7/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics". Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help> :
```

7. Type het volgende commando:

```
print
```

Let op: gebruik hier GEEN ronde haken zoals in print().

Er verschijnt uitgebreide informatie over het Python commando print.

```
The "print" statement
*****  
  
print_stmt ::= "print" ([expression ("," expression)* [",",]]  
| ">>" expression [(",," expression)+ [",",]])  
  
"print" evaluates each expression in turn and writes the resulting  
object to standard output (see below). If an object is not a string,  
it is first converted to a string using the rules for string  
conversions. The (resulting or original) string is then written. A  
space is written before each object is (converted and) written, unless  
the output system believes it is positioned at the beginning of a  
line. This is the case (1) when no characters have yet been written  
to standard output, (2) when the last character written to standard  
output is a whitespace character except "' ''", or (3) when the last  
write operation on standard output was not a "print" statement. (In  
some cases it may be functional to write an empty string to standard  
output for this reason.)  
  
Note: Objects which act like file objects but which are not the  
built-in file objects often do not properly emulate this aspect of  
the file object's behavior, so it is best not to rely on this.  
  
A "'\n'" character is written at the end, unless the "print" statement  
ends with a comma. This is the only action if the statement contains  
just the keyword "print".  
  
Standard output is defined as the file object named "stdout" in the  
built-in module "sys". If no such object exists, or if it does not  
have a "write()" method, a "RuntimeError" exception is raised.  
  
"print" also has an extended form, defined by the second portion of  
the syntax described above. This form is sometimes referred to as  
""print" chevron." In this form, the first expression after the ">>"  
must evaluate to a "file-like" object, specifically an object that has  
a "write()" method as described above. With this extended form, the  
subsequent expressions are printed to this file object. If the first  
expression evaluates to "None", then "sys.stdout" is used as the file  
for output.  
:
```

8. Voer de letter “Q” in om deze informatie af te sluiten.
9. Type het volgende commando:

```
quit
```

Hiermee verlaat je de interactieve hulpfunctie.

Vraag hulp bij een specifiek commando

10. Type het volgende commando:

```
help('print')
```

Nu verschijnt wederom de help-tekst van het commando print.

11. Voer de letter “Q” in om deze informatie af te sluiten.
12. Vraag nu hulp voor een ander commando, door het volgende in te typen:

```
help('raise')
```

13. Bekijk kort de informatie, en voer de letter “Q” in om af te sluiten.

Gebruik Python commando's in de interactieve omgeving

14. Geef het volgende commando:

```
print("Dit is Python!")
```

```
>>> print("Dit is Python!")
Dit is Python!
>>> █
```

15. Voer nu de volgende commando's in:

```
mijnNaam="Pietje Puk"
print(mijnNaam)
```

```
>>> mijnNaam="Pietje Puk"
>>> print(mijnNaam)
Pietje Puk
>>>
```

Sluit de interactieve omgeving af

16. Voer het volgende commando in:

```
exit()
```

Hiermee wordt de interactieve omgeving afgesloten

Bekijk de Python installaties

17. Voer het volgende commando in de opdrachtprompt in:

```
which python
```

```
[pi@raspberry ~]$ which python  
/usr/bin/python
```

Dit geeft aan dat het commando “python” geïnstalleerd is als /usr/bin/python.

18. Type nu het commando “python”, niet gevuld door <ENTER>, maar gevuld door **twee maal de tab toets**.

```
python<tab><tab>
```

```
[pi@raspberry ~]$ python  
python      python2.7-config   python3          python3.7m      python3m  
python2     python2-config    python3.7       python3.7m-config python3m-config  
python2.7   python2-pbr      python3.7-config  python3-config  python-config  
[pi@raspberry ~]$ python■
```

19. Voer het volgende commando in:

```
python2 --version
```

20. Geef nu het volgende commando op:

```
python3 --version
```

```
[pi@raspberry ~]$ python2 --version  
Python 2.7.16  
[pi@raspberry ~]$ python3 --version  
Python 3.7.3  
[pi@raspberry ~]$ ■
```

Nu is te zien dat er meerdere versies van Python (versie 2.x en 3.x) geïnstalleerd zijn op de Raspberry Pi.

21. Voer het volgende commando in:

```
which python
```

22. Geef nu het volgende commando op:

```
file /usr/bin/python
```

```
[pi@raspberry ~]$ which python  
/usr/bin/python  
[pi@raspberry ~]$ file /usr/bin/python  
/usr/bin/python: symbolic link to python2  
[pi@raspberry ~]$ ■
```

23. Let op de uitvoer van het “file” commando.

De uitvoer van het “file” commando geeft aan dat het commando python eigenlijk een snelkoppeling is naar /usr/bin/python2.

24. Sluit de opdrachtprompt af door het volgende commando in te voeren:

```
exit
```

Lab 2.4: Maak een start met het rekenprogramma

In dit lab kom je in aanraking met typische programmeer beginselen door het schrijven van een eenvoudig rekenprogramma. De volgende labs bouwen voort op het werk van dit lab. Zorg er dus voor dit lab gereed te hebben voordat je verder gaat met de volgende labs.

Je gaat nu werken aan een programma om uit te rekenen hoeveel, en welke munten je nodig hebt om een bepaald bedrag in Euro's gepast te betalen in een winkel.

Voorbeeld: Je moet € 2,45 betalen voor een lekker broodje bij de Appie.

Dit bedrag (€ 2,45) kun je betalen met de volgende munten:

- 1x €2
- 2x €0,20
- 1x €0,05

In totaal heb je dus 4 munten nodig.

Jouw programma berekent het minimale aantal munten dat nodig is. Ga uit van alle bekende Euro munten: 2 Euro, 1 Euro, 50 cent, 20 cent, 10 cent, 5 cent, 2 cent en 1 cent.

De aanpak voor de berekening is als volgt: begin met de grootste munt, en kijk hoe vaak deze in het bedrag past. Ga hierna door met de volgende munten, op volgorde van grootte.

Start de MU programmeeromgeving

1. Klik op het Raspberry icoon op de menubalk.
2. Selecteer **Programmeren**.
3. Selecteer **mu**.
4. Wacht een ogenblik tot het beginscherm van Mu verdwijnt.

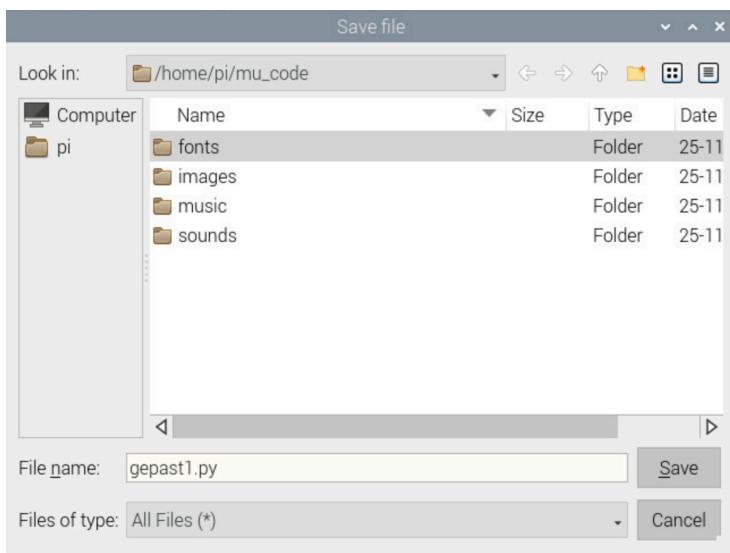
Voer de eerste regels code in

5. Verwijder de eerste regel: # Write your code here :-)
6. Voer de volgende regel code in:

```
input("Geef bedrag op in centen (tussen 0 en 500): ")
```

7. Klik in de menubalk op **Save**.

8. Nu verschijnt een dialoogvenster.
9. Geef bij "File name:" de volgende bestandsnaam op: **gepast1.py**



10. Klik op **Save**.

Test het programma

11. Klik in de menubalk op **Run**.



Je programma start nu en zal vragen om een bedrag in te voeren.

12. Voer een bedrag in, bijvoorbeeld **250**.
13. Klik in de menubalk op **Stop**.

Verander het data type van de invoer naar numeriek

14. Verander je eerste regel code door de invoer op te slaan in een variabele **bedrag**:

```
bedrag = input("Geef bedrag op in centen (tussen 0 en 500): ")
```

15. Zorg er nu voor dat het ingevoerde bedrag wordt opgeslagen als een getal. Verander hiervoor je eerste regel code:

```
bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
```

*De code die je zojuist hebt ingevoerd maakt gebruik van een data type, waarmee je in Python kunt aangeven of de waarde van een variabele moet worden gezien als tekst of als een getal. Om iets als getal te beschouwen, gebruik je het keyword **int** bij het definiëren van een variabele. Voorbeeld: leeftijd = int(46)*

16. Klik op **Save**.

Maak een verzameling van mogelijk munten

17. Voeg de volgende regels code toe, onder de eerste regel code:

```
for munt in (200, 100, 50, 20, 10, 5, 2, 1) :  
    print(munt)
```

De code hierboven is een zogenaamde For-loop, welke door een verzameling van waarden (in dit geval getallen) loopt. Verder is er een variabele "munt" gedefinieerd. Door het aanroepen van de variabele munt, bijvoorbeeld met behulp van het commando print, krijg je toegang tot de waarden in de verzameling.

Belangrijk: let op het inspringen van de tweede regel code! Gebruik hiervoor de TAB toets.

18. Klik op **Save**.

Loop door de verzameling

19. Klik op **Run**.

20. Voer een willekeurig getal in

Let op de uitvoer: alle munten uit te verzameling (de waarden tussen ronde haken) worden getoond. Dit komt door het gebruik van de For-loop.

The screenshot shows a Python development environment. At the top is a toolbar with icons for Mode, New, Load, Save, Stop, Debug, REPL, and Plotter. Below the toolbar, a code editor window displays a file named 'test.py' containing the following code:

```

1 input("Geef bedrag op in centen (tussen 0 en 500): ")
2 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
3     print(munt)
4

```

Below the code editor is a terminal window titled 'Running: test.py' showing the output of the program:

```

Geef bedrag op in centen (tussen 0 en 500): 300
200
100
50
20
10
5
2
1
>>> |

```

21. Klik op **Stop**.

Voeg de berekeningen toe

22. Verwijder de print regel (regel 3), en vervang deze door een nieuwe regel.

23. Declareer een variabele genaamd **aantal**, met als beginwaarde **0**:

```
aantal = 0
```

24. Voeg daaronder een zogenaamde **while loop** toe met de volgende regels code:

```

while (bedrag >= munt) :
    aantal = aantal + 1
    bedrag = bedrag - munt

```

Belangrijk: let op het inspringen van de tweede en derde regel code! Gebruik hiervoor de TAB toets.

25. Klik op **Save**.

26. Zorg ervoor dat je code hetzelfde is als hieronder:

```

1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
4     aantal = 0
5
6     while (bedrag >= munt) :
7         aantal = aantal + 1
8         bedrag = bedrag - munt

```

27. Indien nodig, klik op **Save**.

Voeg commentaar toe om de code te verduidelijken

28. Voeg de volgende regel commentaar toe boven de for-loop regel:

```
# Definieer een verzameling muntwaardes en loop hier doorheen.
```

```
1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 # Definieer een verzameling muntwaardes en loop hier doorheen.
4 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
5     aantal = 0
6
7     while (bedrag >= munt) :
8         aantal = aantal + 1
9         bedrag = bedrag - munt
```

29. Voeg nu de volgende regel commentaar toe boven de regel met de while-loop:

```
#Doe dit zo lang bedrag groter is dan gekozen munt
```

Belangrijk: zorg ervoor dat de het begin van de regel is ingesprongen, net zo ver als de regel met while.

```
1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 # Definieer een verzameling muntwaardes en loop hier doorheen.
4 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
5     aantal = 0
6
7     #Doe dit zo lang bedrag groter is dan gekozen munt
8     while (bedrag >= munt) :
9         aantal = aantal + 1
10        bedrag = bedrag - munt
```

30. Voeg nu meer commentaar toe, achter de code van regels 9 en 10:

```
aantal = aantal + 1      #Verhoog aantal gebruikte munten van een waarde
bedrag = bedrag - munt  #Verminder restbedrag met de waarde van de munt
```

31. Klik op **Save**.

```

1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 # Definieer een verzameling muntwaardes en loop hier doorheen.
4 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
5     aantal = 0
6
7     #Doe dit zo lang bedrag groter is dan gekozen munt
8     while (bedrag >= munt) :
9         aantal = aantal + 1      #Verhoog aantal gebruikte munten van een waarde
10        bedrag = bedrag - munt #Verminder restbedrag met de waarde van de munt
11

```

Toon de berekening

32. Voeg tenslotte de volgende regels code toe, onder de reeds aanwezig regels:

```

if (aantal > 0) :
    print(aantal, 'x', munt)

```

*Let op: de regel met **if** moet even ver inspringen als de regel met **while**. De regel met **print** moet verder inspringen dan de regel met **if**.*

33. Klik op **Save**.

34. Vergelijk je code met het voorbeeld hieronder, en pas aan als dat nodig is.

```

1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 # Definieer een verzameling muntwaardes en loop hier doorheen.
4 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
5     aantal = 0
6
7     #Doe dit zo lang bedrag groter is dan gekozen munt
8     while (bedrag >= munt) :
9         aantal = aantal + 1      #Verhoog aantal gebruikte munten van een waarde
10        bedrag = bedrag - munt #Verminder restbedrag met de waarde van de munt
11
12     if (aantal > 0) :
13         print(aantal, 'x', (munt))
14

```

Voer het programma uit

35. Klik op **Run**.

36. Voer een waarde in tussen 0 en 500, bijvoorbeeld **245**.

Geef bedrag op in centen (tussen 0 en 500): 245

37. Let op de uitvoer, welke aangeeft welke, en hoeveel munten je nodig hebt om tot een bedrag van 245 cent (€2,45) te komen.

```

1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 # Definieer een verzameling muntwaardes en loop hier doorheen.
4 for munt in (200, 100, 50, 20, 10, 5, 2, 1) :
5     aantal = 0
6
7     #Doe dit zo lang bedrag groter is dan gekozen munt
8     while (bedrag >= munt) :
9         aantal = aantal + 1      #Verhoog aantal gebruikte munten van een waarde
10        bedrag = bedrag - munt #Verminder restbedrag met de waarde van de munt
11
12    if (aantal > 0) :
13        print(aantal, 'x', (munt))
14

```

Running: gepast1.py

```

Geef bedrag op in centen (tussen 0 en 500): 245
1 x 200
2 x 20
1 x 5
>>>

```

38. Klik op **Stop**.

Sluit de Mu omgeving NIET af als je direct wilt doorgaan met de volgende labs.

Lab 2.5: Verbeter het rekenprogramma

In dit lab bouw je voort op de code van lab 2.1. Zorg er dus voor dat je lab 2.1 helemaal hebt afgerond en dat je code goed is. Als je code niet werkt, open dan bestand **gepast1.py** in de map **/home/pi/Documents/Module02/oplossingen**.

Je gaat de eerste versie van je programma verbeteren door bestaande code aan te passen en nieuwe code toe te voegen.

Zorg ervoor dat je de Mu programmeeromgeving al hebt geopend.

Maak een nieuw bestand aan

1. Klik op **New**.
2. Klik op **Save**.
3. Nu verschijnt een dialoogvenster.
4. Geef bij “**File name:**” de volgende bestandsnaam op: **gepast2.py**
5. Klik op **Save**.

Kopieer de code van **gepast1.py**

6. Controleer of bestand **gepast1.py** (uit lab 2.1) geopend is. Zo niet, open het dan door te klikken op **Load**.
7. Ga naar bestand **gepast1.py** door op het tabblad te klikken.



8. Beweeg je muis naar de code.
9. Geef **rechtermuisklik**, kies **Select All**.
10. Geef wederom **rechtermuisklik**, kies **Copy**.

A screenshot of a Python code editor window titled 'gepast2.py' and 'gepast1.py'. A context menu is open over the line of code 'bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))'. The menu options are: Undo, Redo, Cut, Copy, Paste, Delete, and Select All. The 'Paste' option is highlighted.

```
1 bedrag = int(input("Geef bedrag op in centen (tussen 0 en 500): "))
2
3 # D
4 for munt in beschikbareMunten:
5     aantal = 0
6
7     while bedrag >= munt:
8         aantal += 1
9         bedrag -= munt
10    if (aantal > 0):
11        print(aantal, 'x', (munt))
12
13
```

11. Ga naar het nieuwe bestand **gepast2.py** door op het tabblad te klikken.
12. Geef **rechtermuisklik**, kies **Paste**.
13. Klik **Save** in de menubalk.
14. Voeg de volgende regels code toe, direct onder regel 1:

```
beschikbareMunten = (200, 100, 50, 20, 10, 5, 2, 1)

print("Het bedrag van %i Eurocent kan met deze munten worden
betaald:" %bedrag)
```

15. Verander regel 6 in:

```
for munt in beschikbareMunten :
```

16. Klik op **Save**.
17. Klik op **Run**.
18. Geef een bedrag in centen op (tussen 0 en 500), bijvoorbeeld **245**.

A screenshot of a terminal window titled 'Running: gepast2.py'. It shows the output of the program when run with the input '245'. The output is:
Geef bedrag op in centen (tussen 0 en 500): 245
Het bedrag van 245 Eurocent kan met deze munten worden betaald:
1 x 200
2 x 20
1 x 5
>>>

19. Klik op **Stop**.

Sluit de Mu omgeving NIET af als je direct wilt doorgaan met de volgende labs.

Lab 2.6: Breid het rekenprogramma uit

In dit lab bouw je voort op de code van lab 2.2. Zorg er dus voor dat je lab 2.2 helemaal hebt afgerond en dat je code goed is. Als je code niet werkt, open dan bestand **gepast2.py** in de map **/home/pi/Documents/Module02/oplossingen**.

Je gaat het rekenprogramma verbeteren door te werken met bedragen in Euro's, in plaats van Euro-centen. Tenslotte ga je functionaliteiten toevoegen om te tonen hoeveel munten er in totaal nodig zijn om een bepaald bedrag gepast te betalen.

Zorg ervoor dat je de Mu programmeeromgeving al hebt geopend.

Maak een nieuw bestand aan

1. Klik op **New**.
2. Klik op **Save**.
3. Nu verschijnt een dialoogvenster.
4. Geef bij "File name:" de volgende bestandsnaam op: **gepast3.py**
5. Klik op **Save**.

Kopieer de code van **gepast2.py**

6. Controleer of bestand **gepast2.py** (uit lab 2.1) geopend is. Zo niet, open het dan door te klikken op **Load**.
7. Ga naar bestand **gepast2.py** door op het tabblad te klikken.
8. Beweeg je muis naar de code.
9. Geef **rechtermuisklik**, kies **Select All**.
10. Geef wederom **rechtermuisklik**, kies **Copy**.
11. Ga naar het nieuwe bestand **gepast3.py** door op het tabblad te klikken.
12. Geef **rechtermuisklik**, kies **Paste**.
13. Klik **Save** in de menubalk.

Gebruik andere data types om met Euro's te kunnen werken

14. Verander de code van regel 1 in het volgende:

```
bedrag = float(input("Geef bedrag tussen 0.00 en 5.00 Euro: "))
```

15. Verander de code van regel 2 in het volgende:

```
beschikbareMunten = (2.00, 1.00, 0.50, 0.20, 0.10, 0.05, 0.02, 0.01)
```

16. Voeg direct onder regel 2 de volgende (nieuwe) coderegel toe:

```
totaalAantal = 0
```

17. Controleer of de eerste 3 regels van je code overeenkomen met het onderstaande:

```
1 bedrag = float(input("Geef bedrag tussen 0.00 en 5.00 Euro: "))
2 beschikbareMunten = (2.00, 1.00, 0.50, 0.20, 0.10, 0.05, 0.02, 0.01)
3 totaalAantal = 0
4
```

18. Verander de laatste regel code (het print commando) in het volgende:

```
print("%i x %1.2f" % (aantal, munt))
```

Let er op dat het niveau van inspringen niet verandert!

19. Klik **Save** in de menubalk.

20. Nu worden bedragen weergegeven als Euro's, en niet meer in euro-centen.

Leer code te verkorten

21. Zoek de regel code **aantal = aantal + 1** en verander dit in het volgende:

```
aantal += 1
```

Voeg nieuwe functionaliteit toe: totaal aantal munten

22. Voeg een nieuwe regel toe in de while loop (dus onder de regel die begint met **while**):

```
totaalAantal += 1      #Verhoog totaal aantal gebruikte munten
```

Let wederom op het niveau van inspringen. Deze regel moet ingesprongen zijn ten opzichte van de regel die begint met while.

```
11 #Doe dit zo lang bedrag groter is dan gekozen munt
12 while (bedrag >= munt) :
13     aantal += 1          #Verhoog aantal gebruikte munten van een waarde
14     totaalAantal += 1    #Verhoog totaal aantal gebruikte munten
15     bedrag = bedrag - munt #Verminder restbedrag met de waarde van de munt
16
```

23. Voeg helemaal onderaan de volgende regels code toe:

```
print("-----")
print("Aantal munten: %i" %totaalAantal)
```

24. Klik **Save** in de menubalk.

Controleer je werk

25. Vergelijk je code met het onderstaande, en pas aan indien nodig:

```

1 bedrag = float(input("Geef bedrag tussen 0.00 en 5.00 Euro: "))
2 beschikbareMunten = (2.00, 1.00, 0.50, 0.20, 0.10, 0.05, 0.02, 0.01)
3 totaalAantal = 0
4
5 print("Het bedrag van %i Euro kan met deze munten worden betaald:" %bedrag)
6
7 # Definieer een verzameling muntwaardes en loop hier doorheen.
8 for munt in beschikbareMunten :
9     aantal = 0
10
11    #Doe dit zo lang bedrag groter is dan gekozen munt
12    while (bedrag >= munt) :
13        aantal += 1                      #Verhoog aantal gebruikte munten van een waarde
14        totaalAantal += 1                #Verhoog totaal aantal gebruikte munten
15        bedrag = bedrag - munt         #Verminder restbedrag met de waarde van de munt
16
17    if (aantal > 0) :
18        print("%i x %.2f" % (aantal, munt))
19
20 print("-----")
21 print("Aantal munten: %i" %totaalAantal)
22

```

Voer het programma uit

26. Klik in de menubalk op **Run**.
27. Geef een bedrag op, bijvoorbeeld **2.45**.

*Let op: gebruik een **punt** (.) en geen komma (,) als scheidingsteken.*

28. Kijk naar de uitvoer van het rekenprogramma.

```

Running: gepast3.py
Geef bedrag tussen 0.00 en 5.00 Euro: 2.45
Het bedrag van 2 Euro kan met deze munten worden betaald:
1 x 2.00
2 x 0.20
1 x 0.05
-----
Aantal munten: 4
>>>

```

*Het programma geeft aan dat het minimale aantal munten om gepast €2,45 te betalen **vier** is: 1x €2, 2x 20 cent en 1x 5 cent.*

29. Klik op **Stop**.

~~~ **Einde van de opdrachten** ~~~