WORKING WITH EXCEL SPREADSHEETS

Roy Van den Broeck & Seppe Mariën

INLEIDING

- Openpyxl module
  - Laat lezen en bewerken van spreadsheets toe
- .xlsx bestanden
  - Werkt op zowel Excel als Openoffice etc…

INSTALLING THE OPENPYXL MODULE

- Windows
  - Pip tool automastich op windows
  - Cmd: pip install openpyxl (desnoods path toevoegen)
- Ubuntu of Debian Linux
  - Eerst installeren van pip tool
    - Terminal: sudo apt-get install python3-pip
  - Dan openpyxl installeren
    - Terminal: sudo pip3 install openpyxl
- Testen installatie in python

```
>>>import openpyxl
```

OPENING EXCEL DOCS W/ OPENPYXL

- Openpyxl.load_workbook(‘example.xlsx’)
  - Neemt de filename
  - Returned <u>workbook</u> data type

- Workbook object
  - Stelt een Excel file voor
  - Zoals file object een geopend text bestand voorsteld

OPENING EXCEL DOCS W/ OPENPYXL

- Excel bestand moet in de current directory zijn:
  - Import os
  - Os.getcwd()
    - Os.chdir()

GETTING SHEETS FROM THE WORKBOOK

- Opvragen van sheets in workbook
  - Wb.get_sheet_names()
- Opvragen van active sheet
  - Wb.get_active_sheet()
- Sheet opslagen in variabelen
  - Sheet = wb.get_sheet_by_name('Sheet3')
- titel van de sheet
  - Sheet.title

GETTING SHEETS FROM THE WORKBOOK

```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> wb.get_sheet_names()
['Sheet1', 'Sheet2', 'Sheet3']
>>> sheet = wb.get_sheet_by_name('Sheet3')
>>> sheet
<Worksheet "Sheet3">
>>> type(sheet) <class 'openpyxl.worksheet.worksheet.Worksheet'>
>>> sheet.title
'Sheet3'
>>> anotherSheet = wb.active
>>> anotherSheet
<Worksheet "Sheet1">
```

CELL OBJECT

- Cell heeft value maar ook een locatie

– Value	bv. Apples	
– Row	bv. 1	
– Column	bv. B	MAAR ook bv. 2
– Coordinate	bv. B1	

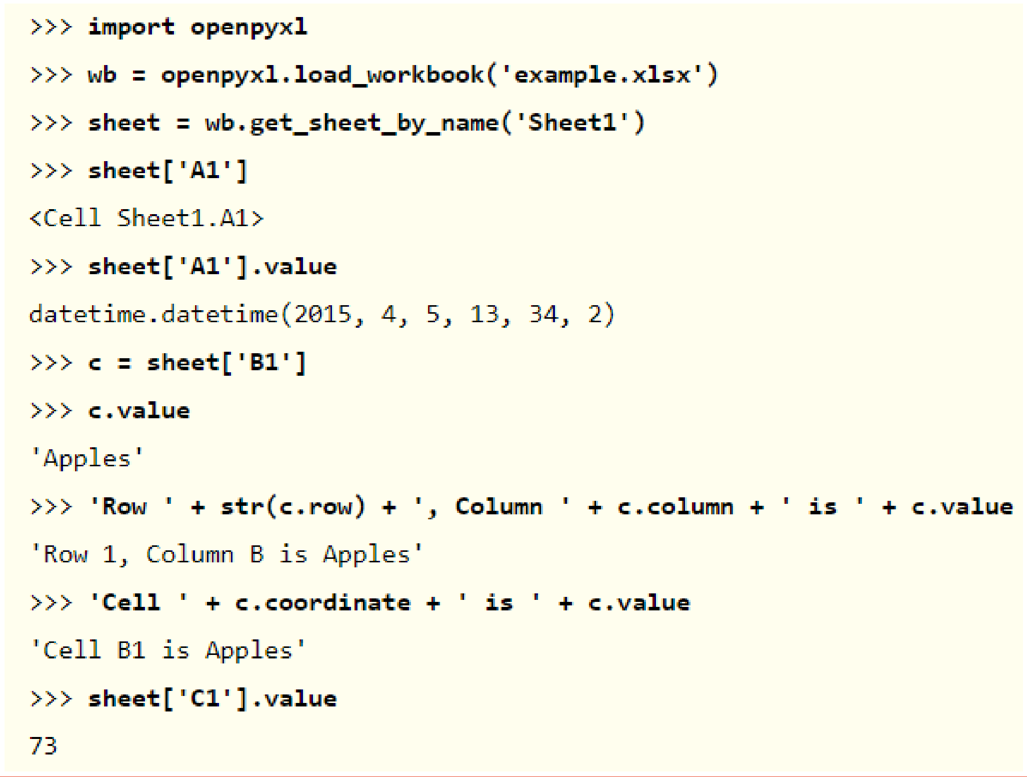
  – (row=1, column = 2) B1

- Datums worden automatisch geïnterpreteerd en hebben een <u>datatime</u> value ipv een string.

GETTING CELLS FROM THE SHEETS

- sheet = wb.get_sheet_by_name('Sheet1’)
  - Vergelijkbaar met array

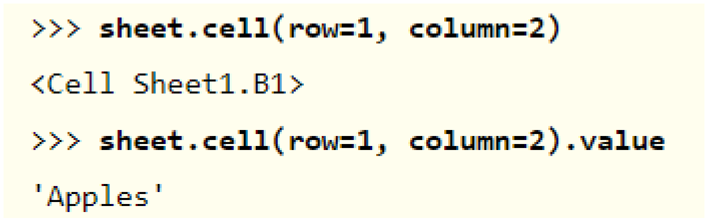
- Waarde van een cel
  - Sheet[‘A1’].value
  - C = sheet[‘A1’]
    - C.value



```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> sheet = wb.get_sheet_by_name('Sheet1')
>>> sheet['A1']
<Cell Sheet1.A1>
>>> sheet['A1'].value
datetime.datetime(2015, 4, 5, 13, 34, 2)
>>> c = sheet['B1']
>>> c.value
'Apples'
>>> 'Row ' + str(c.row) + ', Column ' + c.column + ' is ' + c.value
'Row 1, Column B is Apples'
>>> 'Cell ' + c.coordinate + ' is ' + c.value
'Cell B1 is Apples'
>>> sheet['C1'].value
73
```

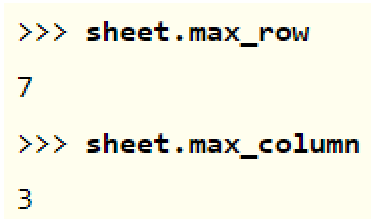
GETTING CELLS FROM THE SHEETS

- Cijfers ipv letters
  - Als kolommen 2 letters krijgen
  - Sheet.cell(row=1, column=2).value



```
>>> sheet.cell(row=1, column=2)
<Cell Sheet1.B1>
>>> sheet.cell(row=1, column=2).value
'Apples'
```

- Max/Min
  - row
    - Sheet.get_highest_row
    - Sheet.max_row
    - Sheet.min_row
  - colomn
    - Sheet.get_highest_column
    - Sheet.max_column
    - Sheet.min_ column
    - Geeft een int terug

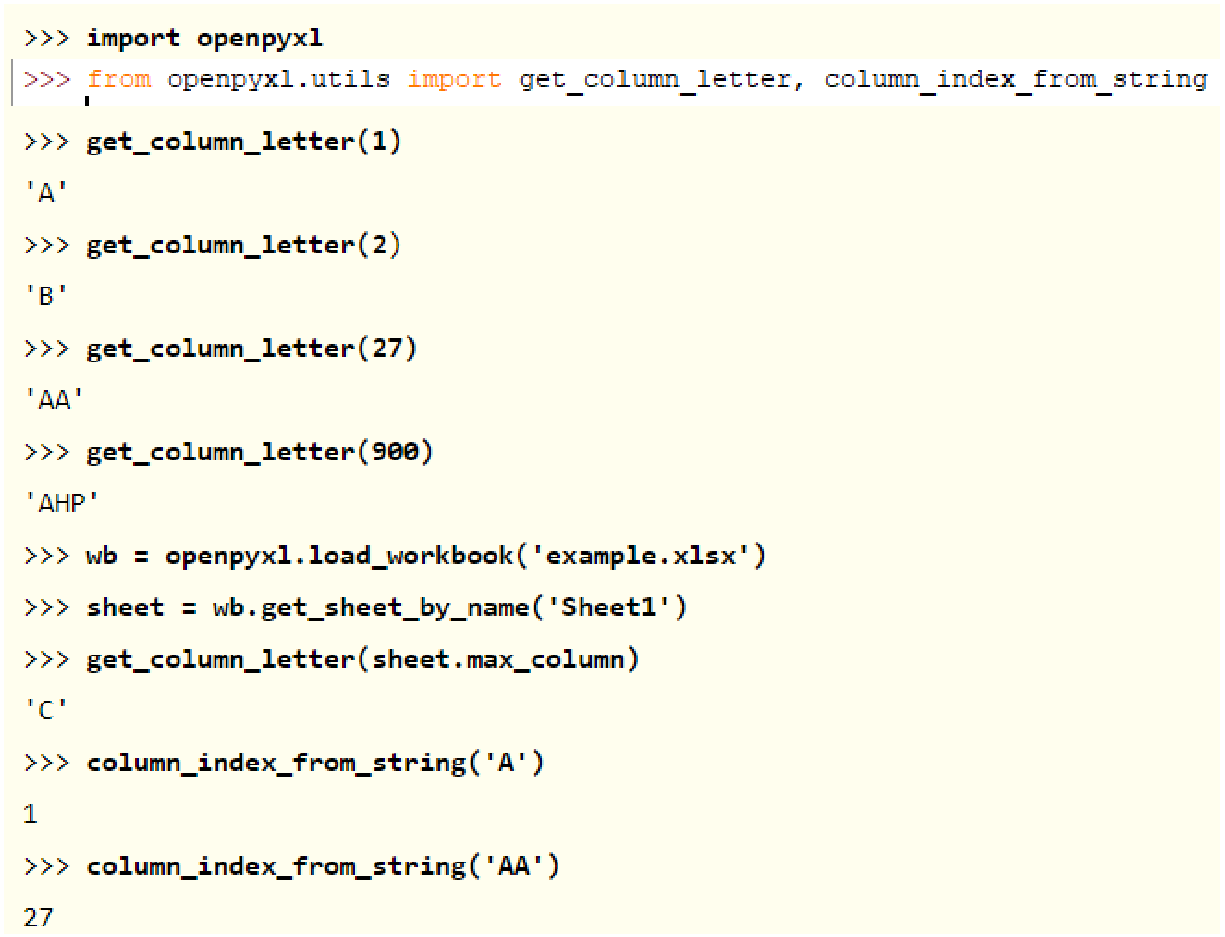


```
>>> sheet.max_row
7
>>> sheet.max_column
3
```

CONVERTING COLUMN LETTERS/NUMBERS

- Functie Importeren
  - from openpyxl.utils
    - Import column_index_from_string()
    - Import get_column_letter()
  - Na importeren
    - Get_column_letter()
      - » Letters naar cijfers
    - Column_index_string()
      - » Cijfers naar letters

CONVERTING VOORBEELD



```
>>> import openpyxl
>>> from openpyxl.utils import get_column_letter, column_index_from_string
>>> get_column_letter(1)
'A'
>>> get_column_letter(2)
'B'
>>> get_column_letter(27)
'AA'
>>> get_column_letter(900)
'AHP'
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> sheet = wb.get_sheet_by_name('Sheet1')
>>> get_column_letter(sheet.max_column)
'C'
>>> column_index_from_string('A')
1
>>> column_index_from_string('AA')
27
```

GETTING ROWS/COLUMNS FROM SHEETS

- Waarde van verschillende cellen
  - C = Sheet[‘A1’:’C3’]
    - 2D array
    - C[0][0].value
      - » Waarde in cel A1
- Alle waarde printen
  - Nested loop

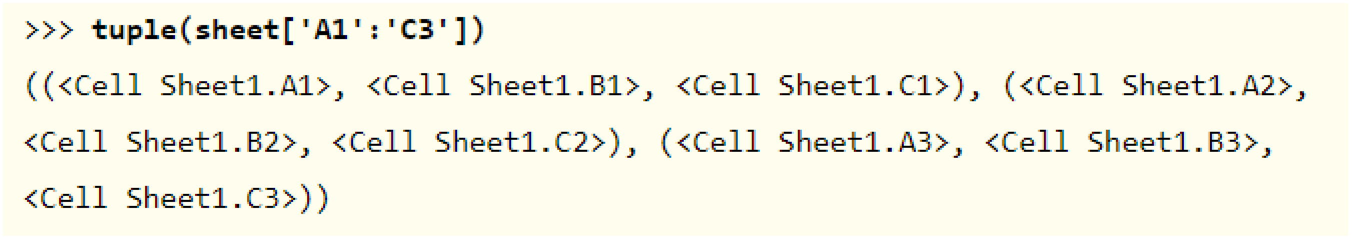
```
>>> for rowOfCellObjects in sheet['A1':'C3']:
        for cellObj in rowOfCellObjects:
            print(cellObj.coordinate, cellObj.value)
         print('--- END OF ROW ---')
```

```
A1 2015-04-05 13:34:02
B1 Apples
C1 73
--- END OF ROW ---
A2 2015-04-05 03:41:23
B2 Cherries
C2 85
--- END OF ROW ---
A3 2015-04-06 12:46:51
B3 Pears
C3 14
--- END OF ROW ---
```

TUPLE

- Tuple()
  - Displayed cell objects in een tuple

- Definitie: “een eindige rij van objecten”

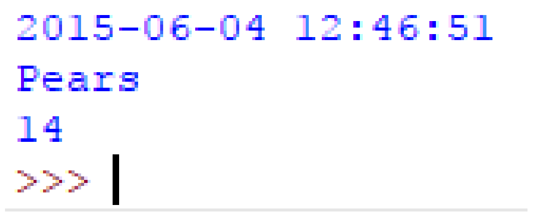


```
>>> tuple(sheet['A1':'C3'])
((<Cell Sheet1.A1>, <Cell Sheet1.B1>, <Cell Sheet1.C1>), (<Cell Sheet1.A2>,
<Cell Sheet1.B2>, <Cell Sheet1.C2>), (<Cell Sheet1.A3>, <Cell Sheet1.B3>,
<Cell Sheet1.C3>))
```

ITER_ROWS()

- Over verschillende rijen gaan
  - for row in sheet.iter_rows(min_row = 3, max_row = 3):
    - for cell in row:
      - » print(cell.value)

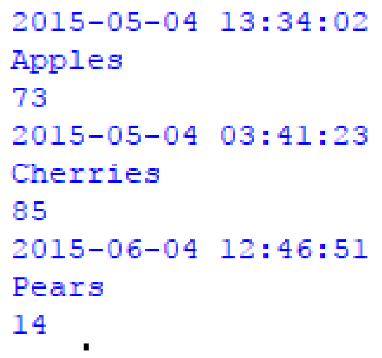
        Print cellen van rij 3



```
2015-06-04 12:46:51
Pears
14
>>> |
```

  - for row in sheet.iter_rows(min_row = 1, max_row = 3):
    - for cell in row:
      - » print(cell.value)

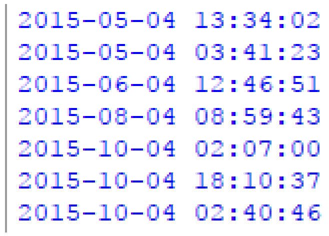
        Print cellen van rij 1 t.e.m. rij 3



```
2015-05-04 13:34:02
Apples
73
2015-05-04 03:41:23
Cherries
85
2015-06-04 12:46:51
Pears
14
```

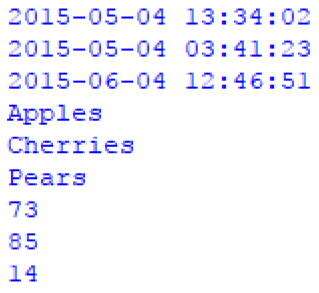
ITER_COLS()

- Sheet.iter_cols(min_col = 1, max_col = 1)
  - for cell in cellObj:
    - print(cell.value)
      - » Print kolom A



```
2015-05-04 13:34:02
2015-05-04 03:41:23
2015-06-04 12:46:51
2015-08-04 08:59:43
2015-10-04 02:07:00
2015-10-04 18:10:37
2015-10-04 02:40:46
```

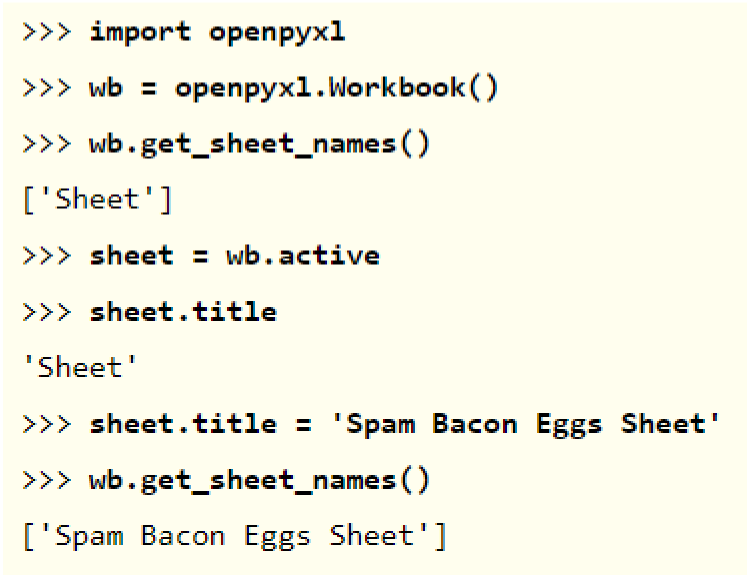
- sheet.iter_cols(min_row = 1, max_row = 3)
  - Zelfde als iter_rows
    - Maar print data per kolom ipv per rij



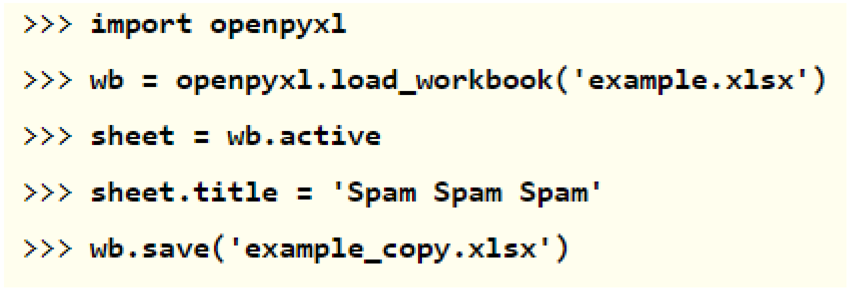
```
2015-05-04 13:34:02
2015-05-04 03:41:23
2015-06-04 12:46:51
Apples
Cherries
Pears
73
85
14
```

CREATING AND SAVING EXCEL DOCS

- Openpyxl.workbook()
  - Maakt blanco workbook
- Save()
  - Niet vergeten saven!
  - Saved niet automatisch
  - Bij andere filenaam wordt kopie gemaakt



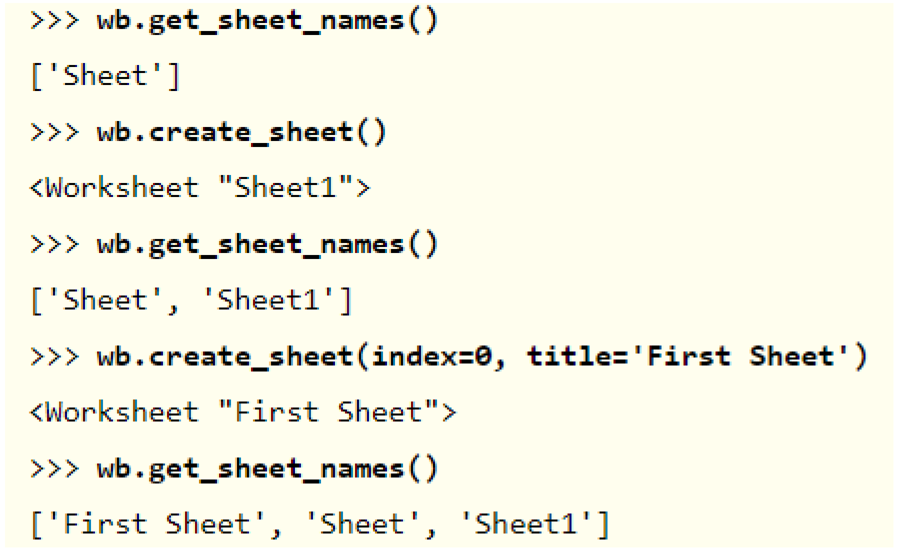
```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> wb.get_sheet_names()
['Sheet']
>>> sheet = wb.active
>>> sheet.title
'Sheet'
>>> sheet.title = 'Spam Bacon Eggs Sheet'
>>> wb.get_sheet_names()
['Spam Bacon Eggs Sheet']
```



```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> sheet = wb.active
>>> sheet.title = 'Spam Spam Spam'
>>> wb.save('example_copy.xlsx')
```

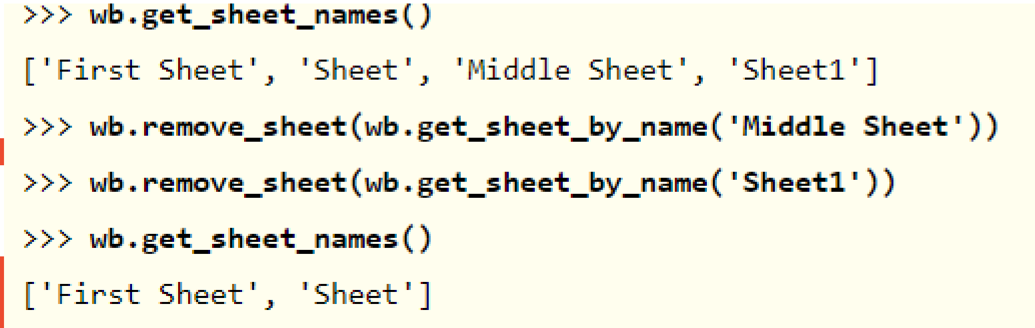
CREATING AND REMOVING SHEETS

- Create_sheet()
  - Aanmaken sheet
  - Nieuwe sheet: SheetX
  - Index en titel kunnen meegegeven worden



```
>>> wb.get_sheet_names()
['Sheet']
>>> wb.create_sheet()
<Worksheet "Sheet1">
>>> wb.get_sheet_names()
['Sheet', 'Sheet1']
>>> wb.create_sheet(index=0, title='First Sheet')
<Worksheet "First Sheet">
>>> wb.get_sheet_names()
['First Sheet', 'Sheet', 'Sheet1']
```

- Remove_sheet()
  - Verwijderen sheet
  - Als enkel naam gekend is:
  - Get_sheet_by_name() value in remove_sheet()



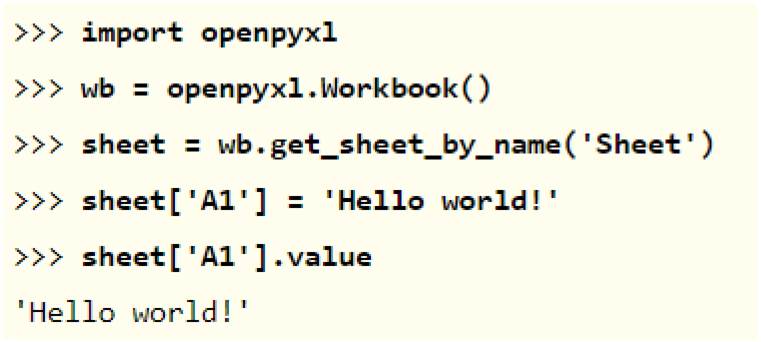
```
>>> wb.get_sheet_names()
['First Sheet', 'Sheet', 'Middle Sheet', 'Sheet1']
>>> wb.remove_sheet(wb.get_sheet_by_name('Middle Sheet'))
>>> wb.remove_sheet(wb.get_sheet_by_name('Sheet1'))
>>> wb.get_sheet_names()
['First Sheet', 'Sheet']
```

DON’T FORGET TO SAVE

- Indien aanpassingen gedaan
  - Wb.save(‘filenaam.xlsx’)

WRITING VALUES TO CELLS

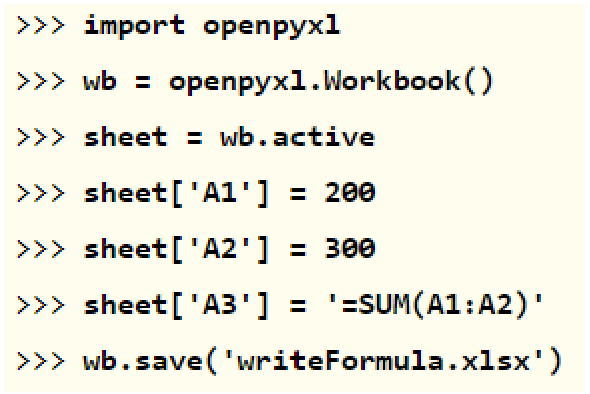
- Sheet[‘A1’] = ‘Hello world!’
  - Vrij simple geen uitleg nodig



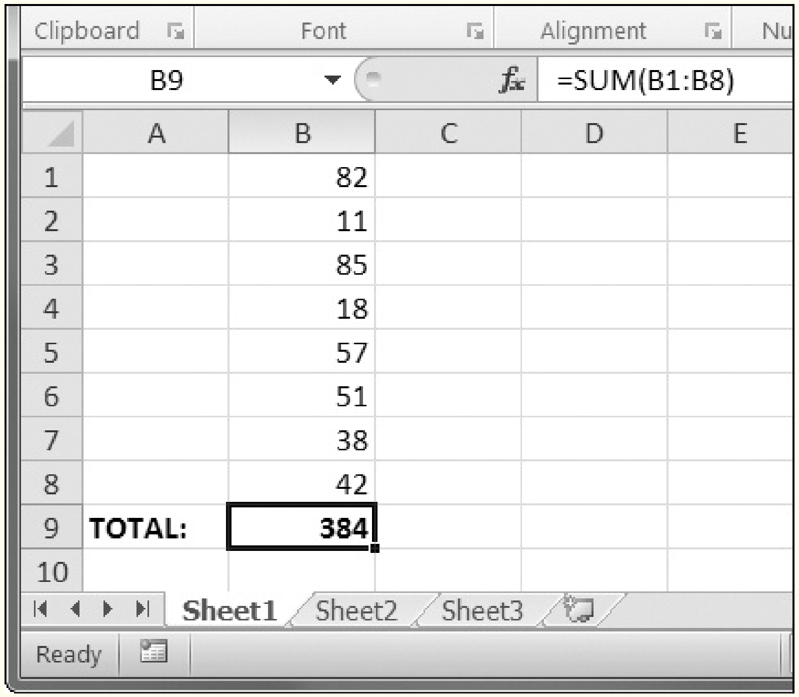
```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.get_sheet_by_name('Sheet')
>>> sheet['A1'] = 'Hello world!'
>>> sheet['A1'].value
'Hello world!'
```

FORMULAS

- Sheet['B9'] = '=SUM(B1:B8)'
  - Schrijven zoals in vorige dia al werd uitgelegd



```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
>>> sheet['A1'] = 200
>>> sheet['A2'] = 300
>>> sheet['A3'] = '=SUM(A1:A2)'
>>> wb.save('writeFormula.xlsx')
```



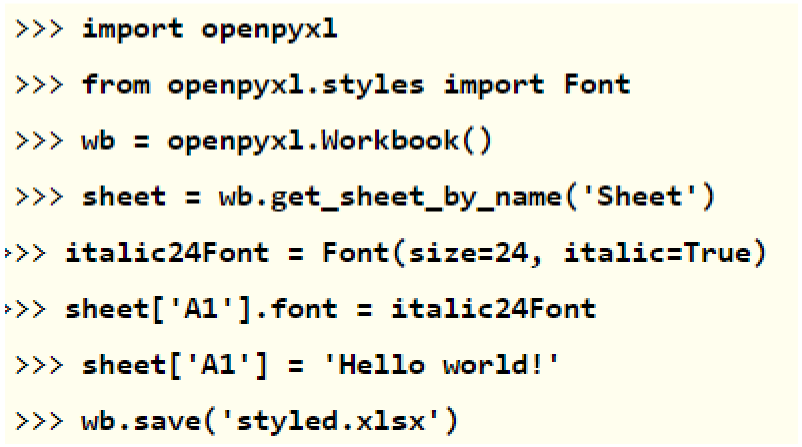
	A	B	C	D	E
1		82			
2		11			
3		85			
4		18			
5		57			
6		51			
7		38			
8		42			
9	TOTAL:	384			
10					

OPGELET

- Als de waarde van [B9] word opgevraagd:
  - Sheet['B1'].value
  - Dan krijgt men =SUM(B1:B8) en <u>NIET</u> de som
- Oplossing
  - Wb2 = load_workbook('Example.xlsx', data_only=True)

SETTING THE FONT SYLE OF CELLS

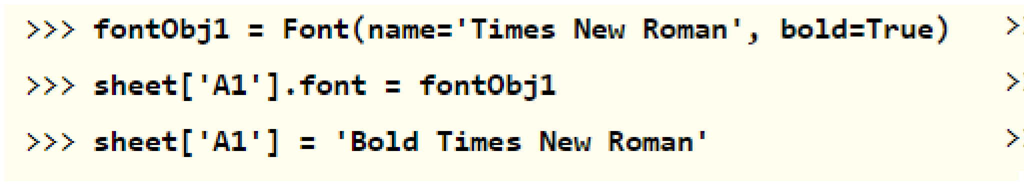
- From openpyxl.styles import Font
  - Dit laat toe om Font() te gebruiken
  - Ipv openpyxl.styles.Font()



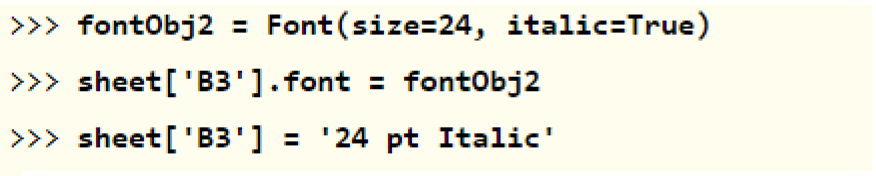
```
>>> import openpyxl
>>> from openpyxl.styles import Font
>>> wb = openpyxl.Workbook()
>>> sheet = wb.get_sheet_by_name('Sheet')
>>> italic24Font = Font(size=24, italic=True)
>>> sheet['A1'].font = italic24Font
>>> sheet['A1'] = 'Hello world!'
>>> wb.save('styled.xlsx')
```

FONT OBJECTS

- Font() heeft parameters die meegeven kunnen worden
  - Naam van het font (string)
  - Grootte van het font (integer)
  - Dikte van het font (Boolean)
  - Schuin geschreven font (Boolean)



```
>>> fontObj1 = Font(name='Times New Roman', bold=True)
>>> sheet['A1'].font = fontObj1
>>> sheet['A1'] = 'Bold Times New Roman'
```

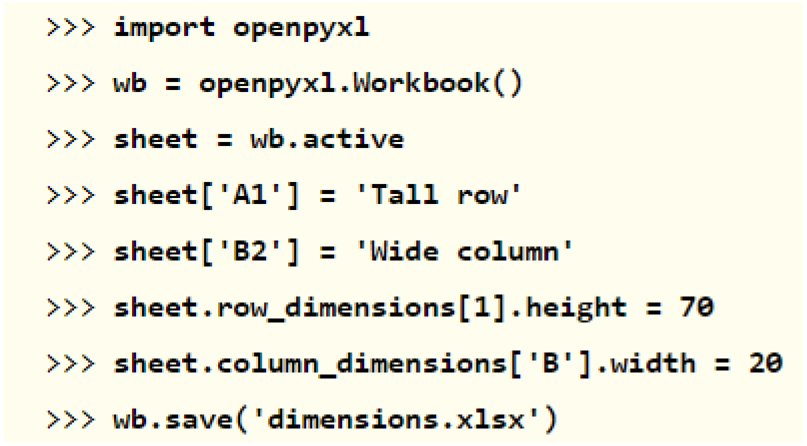


```
>>> fontObj2 = Font(size=24, italic=True)
>>> sheet['B3'].font = fontObj2
>>> sheet['B3'] = '24 pt Italic'
```

THOMAS MORE

SETTING ROW HEIGHT AND COLUMN WIDTH

- Row_dimensions
  - Height
  - Width
- Column_dimensions
  - Height
  - Widht



```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
>>> sheet['A1'] = 'Tall row'
>>> sheet['B2'] = 'Wide column'
>>> sheet.row_dimensions[1].height = 70
>>> sheet.column_dimensions['B'].width = 20
>>> wb.save('dimensions.xlsx')
```

MERGING CELLS

- Merge_cells()
  - Voegt meerdere cellen samen in 1 grote cel
- Waarde steken in deze cellen?
  - Waarde steken in cel links-boven van de merge
- Unmerge_cells()
  - Scheid de samengevoegde cellen

MERGING CELLS

Merge

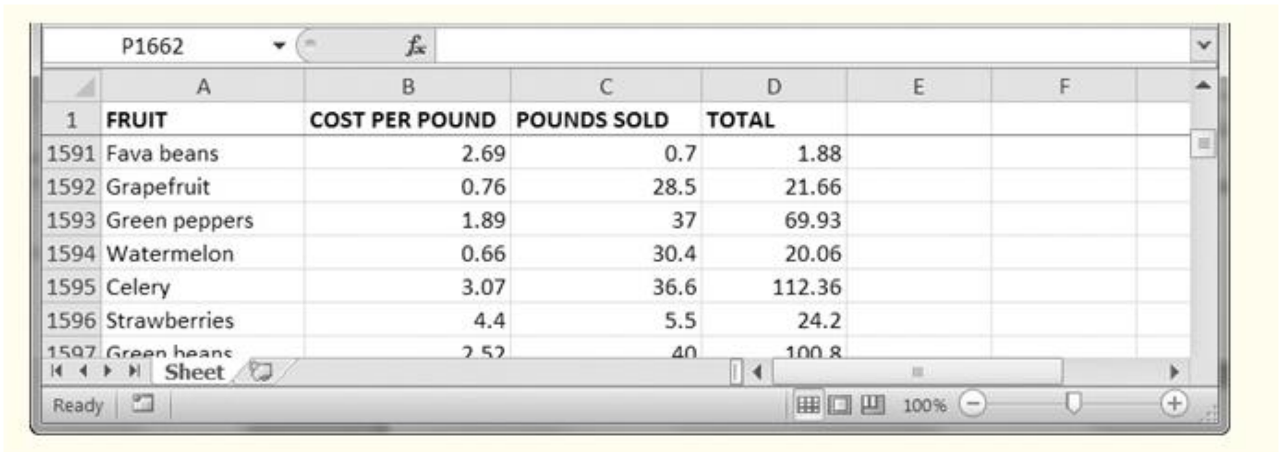
```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
>>> sheet.merge_cells('A1:D3')
>>> sheet['A1'] = 'Twelve cells merged together.'
>>> sheet.merge_cells('C5:D5')
>>> sheet['C5'] = 'Two merged cells.'
>>> wb.save('merged.xlsx')
```

Unmerge

```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('merged.xlsx')
>>> sheet = wb.active
>>> sheet.unmerge_cells('A1:D3')
>>> sheet.unmerge_cells('C5:D5')
>>> wb.save('merged.xlsx')
```

FREEZE PANE

- Sheet.freeze_panes = ‘B2’
- Freezen rijen boven de geselecteerde
  - Rij 1 is frozen
- Freezen kolomen links van de geselecteerde
  - Kolom A is frozen



P1662

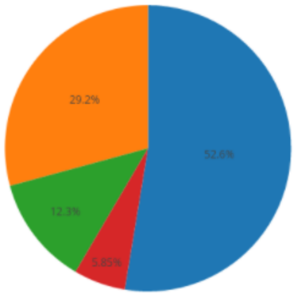
	A	B	C	D	E	F
1	FRUIT	COST PER POUND	POUNDS SOLD	TOTAL		
1591	Fava beans	2.69	0.7	1.88		
1592	Grapefruit	0.76	28.5	21.66		
1593	Green peppers	1.89	37	69.93		
1594	Watermelon	0.66	30.4	20.06		
1595	Celery	3.07	36.6	112.36		
1596	Strawberries	4.4	5.5	24.2		
1597	Green beans	2.52	40	100.8		

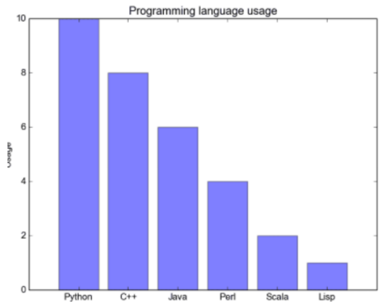
FREEZING PANES

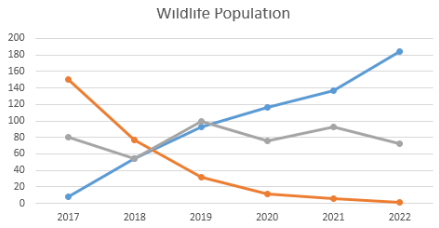
`freeze_panes` setting	Rows and columns frozen
`sheet.freeze_panes = 'A2'`	Row 1
`sheet.freeze_panes = 'B1'`	Column A
`sheet.freeze_panes = 'C1'`	Columns A and B
`sheet.freeze_panes = 'C2'`	Row 1 and columns A and B
`sheet.freeze_panes = 'A1'` or `sheet.freeze_panes = None`	No frozen panes

CHARTS

- 3 Soorten charts:
  - Pie chart
  - Bar Chart
  - Line Chart







- Om een chart te maken 5 stappen nodig:
  - Een reference object
  - Een series object
  - Een chart object
  - Series oject appenden aan chart object
  - Chart object toevoegen aan het worksheet object
    - (Specifieren links boven welke cell voor de chart is)

CHARTS

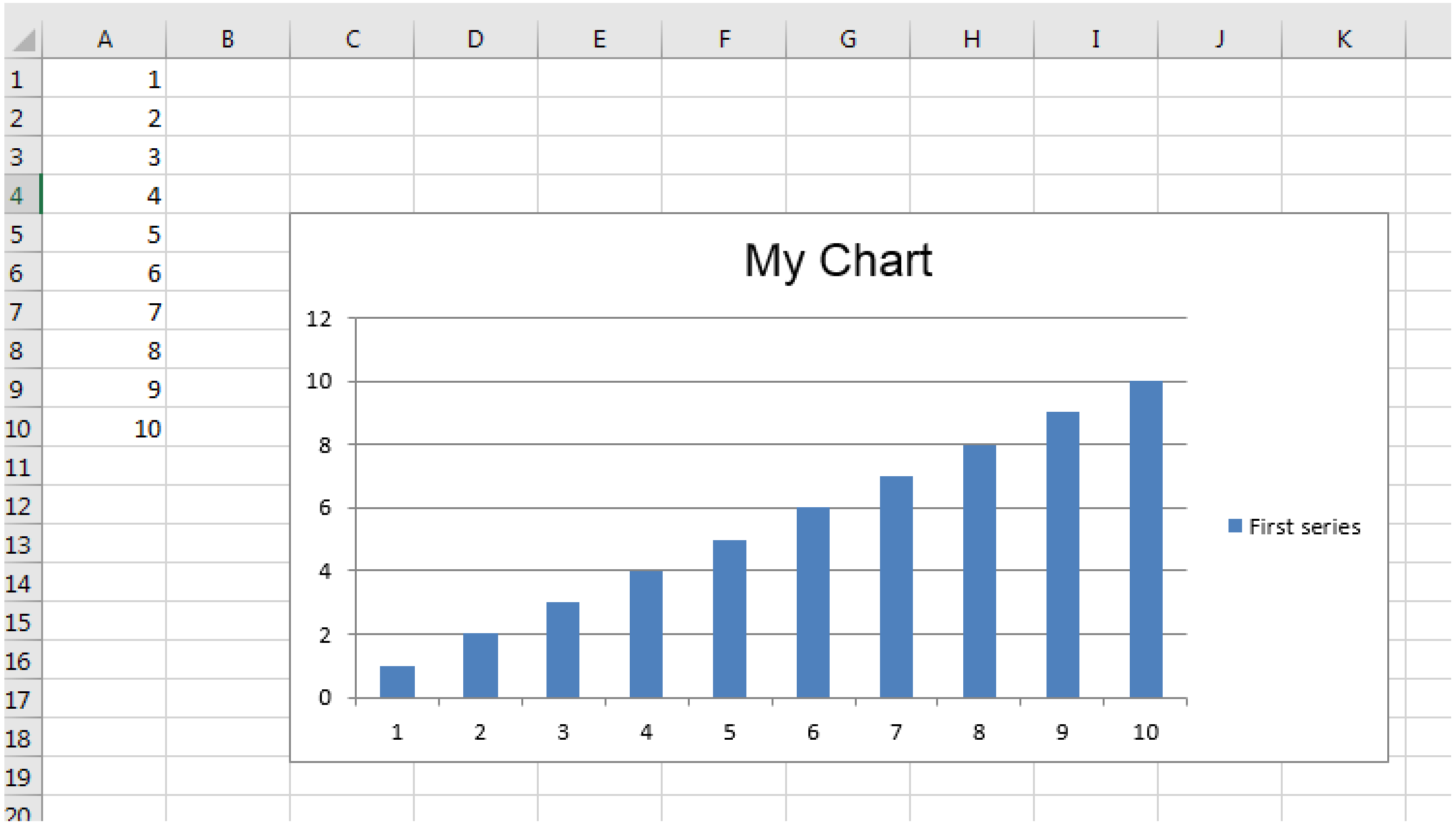
```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> sheet = wb.active
>>> for i in range(1, 11):         # create some data in column A
        sheet['A' + str(i)] = i

>>> refObj = openpyxl.chart.Reference(sheet, min_col=1, min_row=1, max_col=1, max_row=10)

>>> seriesObj = openpyxl.chart.Series(refObj, title='First series')

>>> chartObj = openpyxl.chart.BarChart()
>>> chartObj.title = 'My Chart'
>>> chartObj.append(seriesObj)
>>> sheet.add_chart(chartObj, 'C5')
>>> wb.save('sampleChart.xlsx')
```

CHARTS



	A
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

OPGELET

- Versie 2.3.3 van OpenPyXL
- Load_workbook() laad geen grafieken
- As je een workbook laad en direct saved word de grafiek verwijderd.

QUIZ (1/2)

Q 1. What does the openpyxl.load_workbook() function return?
:

Q 2. What does the get_sheet_names() workbook method return?
:

Q 3. How would you retrieve the Worksheet object for a sheet named 'Sheet1'?
:

Q 4. How would you retrieve the Worksheet object for the workbook's active sheet?
:

Q 5. How would you retrieve the value in the cell C5?
:

Q 6. How would you set the value in the cell C5 to "Hello"?
:

Q 7. How would you retrieve the cell's row and column as integers?
:

Q 8. What do the max_column and max_row sheet methods return, and what is the data
: type of these return values?

Q 9. If you needed to get the integer index for column 'M', what function would you need
: to call?

QUIZ (2/2)

Q: 10. If you needed to get the string name for column 14, what function would you need to call?

Q: 11. How can you retrieve a tuple of all the Cell objects from A1 to F1?

Q: 12. How would you save the workbook to the filename *example.xlsx*?

Q: 13. How do you set a formula in a cell?

Q: 15. How would you set the height of row 5 to 100?

Q: 16. How would you hide column C?

Q: 17. What is a freeze pane?

Q: 18. What five functions and methods do you have to call to create a bar chart?