Data Science and Visualization, S2026

Atomic types, operators, and containers

Jonas Smedegaard [1]

[1]Roskilde University, Department of People and Technology

2026-02-10

Exercise 1: Using markdown

In the empty markdown cell below:

1. Make a level 1 heading with yesterday's calender date
2. Make the level 2 heading 'In the morning'
3. Make a numbered list of the things you did. Put at least one word in bold
4. Make the level 2 heading 'In the afternoon'
5. Make a bullet point list of the things you did. Insert at least one hyperlink to a web page you visited
6. Make the level 1 heading 'My favorite equation'
7. Write a [linear equation][Linear_equation] with a single unknown

If you need help, go to https://www.markdownguide.org/basic-syntax/ for an overview of basic markdown syntax and https://jupyterbook.org/v1/content/math.html for an introduction to writing mathematical expressions in Jupyter Notebook

2026-02-09

In the morning

- woke up
- lit a **fire**
- checked mail
- had breakfast

- coded some Rust ## In the afternoon
- coded some more Rust
- checked the Moodle page for tomorrow's lecture # My favorite equation

$$x = \frac{1}{2}x + 5$$

Exercise 2: Atomic types

2.1 In the code cell below, use the functions `type()` and `print()` to examine and print the types of the variables `u`, `v`, `x`, `y`, and `z`

```
# Assign five objects of different atomic types
# to five different variables u, v, x, y, and z
u = None
v = True
x = 10.2
y = 45
z = "100.01"

# Print the types of u, v, x, y, and z

#print("u is of type ", type(u))
#print("v is of type ", type(v))
#...
vartypes = {"u": u, "v": v, "x": x, "y": y, "z": z}
for name, value in vartypes.items():
    print(f"{name} is of type ", type(value))
```

```
u is of type  <class 'NoneType'>
v is of type  <class 'bool'>
x is of type  <class 'float'>
y is of type  <class 'int'>
z is of type  <class 'str'>
```

2.2 In the code cell below:

1. Cast the variable `x` defined in the code cell above to an integer and assign the resulting object to a new variable `a`. Then print the value and type of `a`

2. Cast the variable `y` defined in the code cell above to a string and assign the resulting object to a new variable `b`. Then print the value and type of `b`
3. Cast the variable `y` defined in the code cell above to a float and assign the resulting object to a new variable `c`. Then print the value and type of `c`

```
# Recast x, y, and z to three new variables a, b, and c.
# Print the types and values of a, b, and c
a = int(x)
print("variable a has value ", a, " and type ", type(a))

b = str(y)
print("variable b has value ", b, " and type ", type(b))

c = float(y)
print("variable c has value ", c, " and type ", type(c))
```

```
variable a has value  10  and type  <class 'int'>
variable b has value  45  and type  <class 'str'>
variable c has value  45.0  and type  <class 'float'>
```

Exercise 3: Operators

3.1 In the code cell below, use arithmetic operators to compute the following expressions. Then print the results:

1. $1489 - 2078 + 1359$
2. $195 * 236/17$
3. 7.4^5
4. $(348 - 63)^4/72$

```
# Compute the expressions and print the results
print("1: ", 1489 - 2078 + 1359)
print("2: ", 195 * 236 / 17)
print("3: ", 7.4**5)
print("4: ", (348 - 63)**4 / 72)
```

```
1:  770
2:  2707.0588235294117
3:  22190.066240000004
4:  91631953.125
```

3.2 In the code cell below, use augmentation operators to do the following:

1. Divide the variable `x` by 10 and overwrite it with the resulting value. Then print the value of `x`
2. Add the string " Visualization" to the variable `y` and overwrite it with resulting value. Then print the value of `y`
3. Repeat the string variable `z` three times and overwrite it with the resulting value. Then print the value of `z`

```
# Assign three objects to three different variables x, y, and z.
x = 35
y = "Data Science and"
z = "Python"

# Change and overwrite x, y, and z. Print the values of x, y, and z
x /= 10
print("1: ", x)
y += " Visualization"
print("2: ", y)
z *= 3
print("3: ", z)
```

```
1:  3.5
2:  Data Science and Visualization
3:  PythonPythonPython
```

3.3 In the code cell below, use Boolean operators and the function `len()` to check if the following statements are true:

1. $120 = 9 * 19 - 51$
2. $6 * 12 + 44 > 5^3$
3. $(-3)^2 + 5 * -3 + 6 \geq 0$
4. There are 28 **or** 32 letters including spaces in the phrase 'Data Science and Visualization'
5. There are 17 letters including spaces in the phrase 'Boolean operators' **and** $20 < 3.5 * 4 + 3^2$

```
# Check if the statements are true
print("1: ", 120 == 9 * 19 - 51)
print("2: ", 6 * 12 + 44 > 5**3, " (", 6 * 12 + 44, " < ", 5**3, ")")
print("3: ", (-3)**2 + 5 * -3 + 6 >= 0)
phrase_len = len("Data Science and Visualization")
print("4: ", (28 | 32) == phrase_len, " (length is ", phrase_len, ")")
print("5: ", len("Boolean operators") and (20 < 3.5 * 4 + 3**2))
```

```
1:  True
2:  False  ( 116  <  125 )
3:  True
4:  False  (length is  30 )
5:  True
```

Exercise 4: Accumulated savings

The the future value of an investment can be computed using the formula:

$$A = P(1 + r)^t$$

where A is the final amount, P is the initial amount, r is the annual interest rate, and t is the number of years. Do the following:

1. The initial amount is 10,000, the annual interest rate is 2.5%, and you save up for 6 years. Assign these values to three new variables `P1`, `r1`, and `t1`. Use the formula to compute the final amount. Assign the result to a new variable `A1` and print the value.
2. The initial amount is 15,000, the annual interest rate is 4.25%, and you save up for 3 years. Assign these values to three new variables `P2`, `r2`, and `t2`. Use the formula to compute the final amount. Assign the result to a new variable `A2` and print the value.
3. The initial amount is 2,500, the annual interest rate is 1.75%, and you save up for 12 years. Assign these values to three new variables `P3`, `r3`, and `t3`. Use the formula to compute the final amount. Assign the result to a new variable `A3` and print the value.

Use the `round()` function to return the results with 2 decimal places. You will use the variables created in this excercise in the next exercise

```
# Compute and print accumulated savings under scenario 1
P1 = 10000
r1 = 2.5/100
t1 = 6
A1 = round(P1*(1+r1)**t1, 2)
print("A1 = ", A1)
# Compute and print accumulated savings under scenario 2
P2 = 15000
r2 = 4.25/100
t2 = 3
```

```
A2 = round(P2*(1+r2)**t2, 2)
print("A2 = ", A2)
# Compute and print accumulated savings under scenario 3
P3 = 2500
r3 = 1.75/100
t3 = 12
A3 = round(P3*(1+r3)**t3, 2)
print("A3 = ", A3)
```

```
A1 =  11596.93
A2 =  16994.93
A3 =  3078.6
```

Exercise 5: Containers

5.1 Use the code cell below to do the following:

1. Create a list with the names of the seven days of the week. Assign the list to the variable `days`
2. Use slice notation and the indexing operator on the list `days` to create two new lists `weekdays`, i.e., 'Monday' through 'Friday', and `weekend`, i.e. 'Saturday' and 'Sunday'. Print the new lists
3. Use a list method to do add an extra 'Sunday' to the end of the list `weekend`. Then print the list
4. Use a list method to remove the first occurence of the 'Sunday' from the list `weekend`. Now print the list again
5. Use a list method to reverse the order of the list `weekdays` so the first element is now 'Friday'. Then print the list again

```
# Create a list the names of the days of the week
days = [
    "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
    "Saturday", "Sunday"
]
# Create two list with weekdays and weekend
weekdays = days[0:5]
print("1: week days: ", weekdays)
weekend = days[-2:7]
print("2: weekend: ", weekend)
# Add "Sunday" to the end of the list weekend
weekend.append("Sunday")
```

```
print("3: extended weekend: ", weekend)
# Remove the first occurence of "Sunday" from the list weekend
weekend.pop(1)
print("4: reduced extended weekend: ", weekend)
# Reverse of the order of the list weekdays
weekdays.reverse()
print("5: week days reversed: ", weekdays)
```

```
1: week days:  ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
2: weekend:  ['Saturday', 'Sunday']
3: extended weekend:  ['Saturday', 'Sunday', 'Sunday']
4: reduced extended weekend:  ['Saturday', 'Sunday']
5: week days reversed:  ['Friday', 'Thursday', 'Wednesday', 'Tuesday', 'Monday']
```

5.2 Use the code cell below to do the following:

1. Create a dictionary that maps the keys 'Initial amount', 'Interest rate', 'Years', and 'Final amount' to the values stored in the variables `P1`, `r1`, `t1`, and `A1` that you created under the first scenario in exercise 4. Assign the dictionary to the new variable `S1`.
2. Create corresponding dictionaries for the second and third scenarios in exercise 4 and assign them to the new variables `S2` and `S3`
3. Now create a nested dicitionary that maps the keys 'Scenario 1', 'Scenario 2', and 'Scenario 3' to the dictionaries stored in the variables `S1`, `S2`, and `S3`. Assign the nested dictionary to the variable `accumulated_savings`
4. Use the indexing operator on the nested dictionary `accumulated_savings` to access the final amount under scenario 2 and print it

```
# Create a dictionary for the first scenario in exercise 4
S1 = {
    "Initial amount": P1,
    "Interest rate": r1,
    "Years": t1,
    "Final amount": A1,
}
# Create corresponding dictionaries
# for the second and third scenarios in exercise 4
S2 = {
    "Initial amount": P2,
    "Interest rate": r2,
    "Years": t2,
    "Final amount": A2,
```

```
}
S3 = {
    "Initial amount": P3,
    "Interest rate": r3,
    "Years": t3,
    "Final amount": A3,
}
# Create a nested dictionary
# of accumulated savings under the three scenarios
accumulated_savings = {
    "Scenario 1": S1,
    "Scenario 2": S2,
    "Scenario 3": S3,
}
# Access the final amount under scenario 2 and print it
print("4: final amount under scenario 2: ",
      accumulated_savings["Scenario 2"]["Final amount"])
```

```
4: final amount under scenario 2:  16994.93
```

Exercise 6: String methods

Use the code cell below to do following:

1. Go to the FUTUREDEMICS page on Roskilde University's research portal. Copy the URL and assign it to a new variable `URL`
2. Use a string method to count the number forslashes '/' in the URL
3. Use the functions `len()` and `set()` to count how many different unique characters the URL contains
4. Use a string method and the indexing operator to split `URL` into two new strings: One containing the base URL up to and incldluing '.dk' and one containing the part of the URL after the second to last forslash, i.e., the title of the research project. Assign the new strings to the variables `base_URL` and `project_title`
5. Use a string method to replace all hyphens '-' in `project_title` with whitespaces
6. Use a string method to strip the string " futuredemics" from `project_title`
7. Use a string method to put `project_title` in titlecase. Then print it

```
# Assign the FUTUREDEMICS URL to a new variable URL
URL = "https://forskning.ruc.dk/"\
+ "da/projects/"\
+ "nordic-pandemic-preparedness-modelling-network-futuredemics/"
# Count the number of forslashes in URL
#print("2: slashes: ", URL.count("/"))
# Count the number of unique characters in URL
#print("3: unique chars: ", len(set(URL)))
# Create two new string variables from URL
URL_parts = URL.partition(".dk")
base_URL = "".join(URL_parts[0:2])
project_title = URL_parts[2].split("/")[-2]
#print(f"4: base URL: {base_URL}\n   project title: {project_title}")
# Replace hyphens with whitespace in project_title
project_title = project_title.replace("-", " ")
#print("5: project title, spaced: ", project_title)
# Strip the word "futuredemics" from the project tilte
project_title = project_title.removesuffix(" futuredemics")
#print("6: project title, shortened: ", project_title)
# Title case project_title
project_title = project_title.title()
# Print the project title
print(project_title)
```

```
Nordic Pandemic Preparedness Modelling Network
```

Exercise 7: Exporting Jupyter Notebooks to PDF

Several possibilities for exporting Jupyter Notebooks to PDF exists. Two of them are:

1. **Print to PDF via HTML**:
    - Select File > Save and Export Notebook As > HTML in Jupyter Notebook. An HTML file is saved in you Downloads folder.
    - Open the HTML, press `Ctrl + P`, select Print to PDF, and choose a file path
2. **PDF via LaTeX**:
    1. Install necessary dependencies:
    - Go to https://miktex.org/download and download and install MiKTeX

- Go to https://pandoc.org/installing.html and download and install Pandoc
- Run `pip install nbconvert` in a code cell in Jupyter Notebook
- Select Kernel > Restart Kernel in Jupyter Notebook

2. Export to PDF: Select File > Save and Export Notebook As > PDF in Jupyter Notebook. An PDF file is saved in you Downloads folder. Open it, click save, and choose a file path

The advantage of the first approach is that it has no dependencies, i.e., you do not have to install other applications to make it work. The drawback of the first approach is that you have limitied control over the output. For example, lines in code cell that are too long to fit the page are cut off rather than shown across several lines. To make the second approach work, you have to install the dependencies listed. Once you have done that, it is fast and easy to convert to PDF via LaTeX. If following the steps listed here does not enable you to produce a PDF via LaTeX (if you use Mac or Linux you may run into trouble), I recommend that you use your favorite search engine or go talk to a large language model to find a solution that works on your system.

Try exporting your notebook to PDF using both methods. You will need to export a Jupyter Notebook to PDF when you will eventually submit your exam mini-project on May 1.