Data Science and Visualization, S2026

Programming structures

2026-02-17

Exercise 1: Conditional statements

1.1 In the empty code cell below, use conditional statements to write a program that takes in an object of an arbitrary type and outputs the following:

- If the object is a string with more than 30 characters, print the statement 'This is a long string'
- If the object is a string with less than 30 characters, print the statement 'This is a short string'
- If the object is an integer or a float that is greater than 100, print the statement 'This is a large number'
- If the object is an integer or a float that is less than 100, print the statement 'This is a small number'
- If the object is not a string, an integer, or a float, print the the statement 'This is not a string or a number'

When you have written the program, test that the program works as intended on different types of objects

1.2 In the empty code cell below, use conditional statements to write a program that takes in an object of an arbitrary type and outputs the following:

- If the object is an integer or float that is positive, is less than 100, is a multiple of 3, and is a multiple of 5, square it and print the result
- If the object is an integer or float that is positive, is less than 100, is a multiple of 3, but is not a multiple of 5, raise it to the third power and print the result
- If the object is an integer or float that is positive, is less than 100, and is not a multiple of 3, raise it to the fourth power and print the result
- If the object is an integer or float that is positive and greater than 100, add 100 to it and print the result
- If the object is an integer or float that is equal to zero or negative, subtract 100 from it and print the result

- If the object is not an integer or float, print the statement 'This is not a number'

When you have written the program, test that the program works as intended on different types of objects. You will need to use the modulo operator to write the program. You will need work with many levels of logical conditions, so keep in mind that we use indentation by four whitespaces to delineate levels in the hierarchy

Exercise 2: Loops

2.1 The code cell below creates a list of 4 strings representing 4 Danish words. The list is assigned to the variable `incl_danish_letters`. Do the following:

1. Create an empty list and assign it to the variable `excl_danish_letters`
2. Use a `for` loop to iterate over the elements of `incl_danish_letters`, replace the letter 'ø' with the letters 'oe', and then append the resulting string to the list `excl_danish_letters`.
3. When `for` loop is completed, print the value of `excl_danish_letters`

You will need to use a string method to write the `for` loop

```
# Create a list of 4 Danish words
# and assign it to the variable incl_danish_letters
incl_danish_letters = ["rød", "grød", "med", "fløde"]

# Create an empty list
# and assign it to the variable excl_danish_letters

# Use a for loop to replace "ø" with "oe"

# Print the value of excl_danish_letters
```

2.2 The code below creates a list of of different objects, some of which are words representing words with and without Danish letters. The list is assigned to the variable `l`. Do the following:

1. Assign the integer value 0 to two new variables `with_danish_letters` and `without_danish_letters`

2. Use a `for` loop to iterate over the elements of `l`. For each element in `l` that contains either 'æ', 'ø', or 'å', increment `with_danish_letters` by 1. For each element in `l` that does not, increment `without_danish_letters` by 1
3. When the `for` loop is completed, print the values of `with_danish_letters` and `without_danish_letters`

You will need to use conditional statements and the keyword `continue` to write the `for` loop

```
# Create a new list and assign it to the variable l
l = [
    "Odense", "Hillerød", None, "København", "Roskilde", True,
    "Brøndby", "Grenå", 100,  "Esbjerg", "Næstved"
]

# Assign the integer value 0 to two new variables

# Use a for loop to count the number of words in l
# with and without Danish letters

# Print the values of with_danish_letters and without_danish_letters
```

2.3 The code cell below creates a list of courses offered by INM. Do the following:

1. Create a new set and assign it to the variable `combinations`
2. Suppose that you can choose any ***3*** courses from the list you want next semester. Use a nested `for` loop, the set `combinations`, and the `frozenset()` function to figure out how many different options you have, i.e., how many unique combinations are there of the courses in the list?
3. When the loop is completed print the length of `combinations`

```
# Create a list of INM courses
courses = [
    'Algebra', 'Bioprocesses', 'Botany', 'Calculus', 'Cell Biology',
    'Chemistry 1', 'Chemistry 2', 'Electrodynamics', 'Empirical Data',
    'Essential Computing', 'Experimental Methods', 'Linear Algebra',
    'Logic', 'Mechanics', 'Molecular Biology', 'Scientific Computing',
    'Theory of Science', 'Zoology'
]

```

```
# Create a new set and assign it to the variable combinations

# Use a nested for loop to find the number of combinations

# Print the number of combinations
```

Exercise 3: GDP per capita and population growth

The code cell below creates a list of Danish population counts in number of people and GDP in Danish Kroner from 1980 to 2024. The numbers are taken from https://www.statbank.dk/statbank5a/default.asp?w=1536

```
# Create a list of Danish population counts from 1980 to 2024
population_count = [
    5122065, 5123989, 5119155, 5116464, 5112130,
    5111108, 5116273, 5124794, 5129254, 5129778,
    5135409, 5146469, 5162126, 5180614, 5196641,
    5215718, 5251027, 5275121, 5294860, 5313577,
    5330020, 5349212, 5368354, 5383507, 5397640,
    5411405, 5427459, 5447084, 5475791, 5511451,
    5534738, 5560628, 5580516, 5602628, 5627235,
    5659715, 5707251, 5748769, 5781190, 5806081,
    5822763, 5840045, 5873420, 5932654, 5961249
]

# Create a list of Danish GDP from 1980 to 2024
GDP = [
    1061766e6, 1044727e5, 1079590e6, 1113152e6,
    1162762e6, 1215402e6, 1291354e6, 1303233e6,
    1304824e6, 1318827e6, 1346741e6, 1363521e6,
    1402850e6, 1400236e6, 1469056e6, 1514527e6,
    1566212e6, 1616484e6, 1651897e6, 1699194e6,
    1767252e6, 1783295e6, 1800910e6, 1815208e6,
    1872596e6, 1933235e6, 2000184e6, 2014425e6,
    2025536e6, 1927603e6, 1981158e6, 1988226e6,
    1994714e6, 2033469e6, 2076855e6, 2128263e6,
    2195314e6, 2263289e6, 2295506e6, 2334244e6,
    2326592e6, 2501738e6, 2561083e6, 2470753e6,
    2535701e6
```

```
]
```

3.1 GDP per capita can be computed using the formula:

$$PCAP = GDP/POP$$

where PCAP denotes GDP per capita and POP denotes population count. In the code below:

1. Create an empty list and assign it to the variable `GDP_per_capita`
2. Use a `for` loop, a built-in sequence function, the two lists created in code cell above, and the formula to compute Danish GDP per capita for the years 1980-2024. Round the values to 2 decimal places and append them to the list `GDP_per_capita`
3. Use a built in sequence function to create a list containing the years from 1980 to 2024. Assign the list to the variable `year`
4. Create a new dictionary with the keys 'year', 'population count', 'GDP', and 'GDP per capita'. Use the lists with the corresponding names as the values. Assign the dictionary to the variable `d`

```
# Create an empty list and assign it to the variable GDP_per_capita

# Use a for loop
# to compute Danish GDP per capita for the years 1980-2024

# Create a list containing the years from 1980 to 2024
# and assign to the variable year

# Create a dictionary
# with year, population_count, GDP, and GDP_per_capita
# and assign it to the variable d
```

3.2 In the code cell below:

1. Create an empty list and assign it to the variable `population_growth`
2. Assign the none type object to two new variables `pop_current_year` and `pop_last_year`
3. Use a `for` loop and the list `population_count` to compute Danish population growth measured in mio. people for each year between 1981-2024. Round the values to 4 decimal places and append them to the list `population_growth`

4. Add the key 'population growth' to the dictionary `d` and use the list with corresponding name as the value.
5. Use the dictionary and a another `for` loop to answer the following question: In which year between 1981-2024 was Danish population growth the highest? Here you can make good use of the function max() combined with the slicing operator `[start:stop]`

```
# Create an empty list
# and assign it to the variable population_growth

# Assign the integer value 0 to two new variables

# Use a for loop
# to compute population growth for the years 1981-2024

# Add population_growth to the dictionary d

# Use a for loop
# to find the year with the highest population growth
```

3.3 Given a growth rate r and the value of GDP per capita in the $PCAP_t$, we can compute the value of GDP per capita $PCAP_{t+1}$ next year using the formula:

$$PCAP_{t+1} = PCAP_t(1 + r)$$

Assume (wrongly, for sure) that the growth rate from the year 2024 and forever onwards will be 2%. Assign the integer value 2024 to a new variable `year` and the value of GDP per capita that you computed in exercise 3.1 to another new variable `GDP_per_capita`. Then use the formula and a `while` loop to find the first year when Danish GDP capita will exceeed 1 mio. DKK

```
#  Assign the integer value 2024 to a new variable year

# Assign the value of GDP per capita in 2024 to a new variable GDP

# Use a while loop
# to find the first year where GPD per capita exceed 1 mio.

# Print the value of the year
```

Exercise 4: Comprehensions

The code cell below creates three lists of names, means of transportation, and distance in kilometers

```
names = [
    "Søren", "Thomas", "Magnus", "Iben", "Mia",
    "Casper", "Frank", "Anders", "Signe", "Anna"
]
means = [
    "Bus", "Train", "Bus", "Train", "Bus",
    "Car", "Car", "Train", "Car", "Bus"
]
distances = [
    137.90091212041625, 278.4024225886178, 51.61539428811396,
    35.67295085315202, 167.76993783032339, 106.2643570032314,
    306.66050065687415, 121.50183254326156, 94.33908028762917,
    141.6248089005064]
```

4.1 In the code cell below:

1. Use a list comprehension and the list `means` to overwrite this list with a new list of means of transportation that uses the first letter of each string as an abbreviation. As a string is an iterable, you can use the indexing operator on it like any other iterable. Then print the value of `means`
2. Use another list comprehension and the list `distances` to overwrite this list with a new list of distances rounded to 2 decimal places. Then print the value of `distances`

```
# Overwrite the list means with abbreviations

# Print the value of means

# Overwrite the list distance with numbers
# rounded to 2 decimal places

# Print the value of means
```

4.2 In the code cell below:

1. Use a dictionary comprehension and a built-in sequence function, and the lists `names` and `means` to create a dictionary that uses the names as keys and the means of transportation as the values. Assign the dictionary to the variable `d`. Then print the value of `d`
2. Assume that travelling by bus allows you to travel with an average speed of 50 kilometers per hour, while travelling by train or car allows you to travel with an average speed of 80 kilometers per hour. Use a dictionary comprehension, a built-in sequence function, and the lists `names`, `means` , and `distances` to create a new dictionary that maps names to travelling time in hours, rounded to 2 decimal places. Assign the dictionary to the variable `e`. Then print the value of `e`

```
# Create a dictionary that maps names to means of transportation
# and assign it to the variable d

# Print the value of d

# Create a dictionary that maps names to travelling time
# and assign it to the variable d

# Print the value of e
```

Exercise 5: Functions

5.1 The code cell creates a dictionary that maps the keys 1, ..., 5 to 5 lists of integers. The dictionary is assigned to the variable `d`. Do the following: 1. Write a `def` function that takes in a list and outputs the minimum value, the maximum value, and the median. To find the median in a list with an odd number elements, you can sort the list using a list method, find the middle index position by adding 1 to the numer of elements in the list and dividing by 2, and then retrieve the value at that index position. Recall that division by always produces a float, and the indexing operator expects an integer. 2. Use a `for` loop and your function to find the minimum, maximum, and median of the 5 lists in `d`. Print the values in each iteration of the `for` loop

```
# Create a dicitionary with 5 lists of integers
# and assign it to the variable d
d = {
    1: [
        673, 1254, 744, 408, 988, 14, 271, 520, 141, 371,
```

```
        83, 107, 341, 161, 550
    ],
    2: [
        205, 344, 564, 474, -195, 315, 978, 462, 707, 199,
        527, 150, 578, 939, 362, 91, 573
    ],
    3: [666, 622, 830, 318, 519, 206, 677],
    4: [
        1029, 301, 415, 340, 603, 144, 239, 48, 411, 324, 624,
        395, 179, 774, 54, 557, 424, 1029, 319, 1056, 561
    ],
    5: [1310, 462, 86, 438, 708]
}

# Define a function
# that finds minimum, maximum, and median of a list

# Loop over the keys and value in d
# an print the minimum, maximum, and median
```

5.2 The code cell below creates a list of URLs from Roskilde University's research portal. Do the following:

1. Use the list `URLs`, the built-in function `list()`, the built-in function filter(), and `lambda` functions to create two new lists of URLs relating to organizations and URLs relating to pulications. Assign the lists to the variables `organization_URLs` and `publication_URLs`
2. Use the lists `organization_URLs` and `publication_URLs`, the built-in function `list()`, the built-in function map(), and a `lambda` function to create two new lists of title-formatted names of organizations and publication. Assign the lists to the variables `organizations` and `publications`. To do this, you need the same built-in string methods that you used in exercise 6 in the exercise set from lecture 1
3. Print the values of `organizations` and `publications`

```
# Create a list of URLs
URLs = [
    "https://forskning.ruc.dk/da/organisations/arbejdsliv/",
    "https://forskning.ruc.dk/da/publications/"\
+ "a-stackelberg-nash-game-framework-for-overcoming-remanufacturing-/",
    "https://forskning.ruc.dk/da/organisations/"\
```

```
+ "centre-for-statecraft-and-international-order/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "economic-policy-institutions-and-change/",
    "https://forskning.ruc.dk/da/publications/"\
+ "handbook-of-postcolonial-and-decolonial-linguistics/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "magt-identitet-og-kritik/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "center-for-interdisciplinary-plastic-research/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "centre-for-mathematical-modeling-human-health-and-disease/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "environmental-humanities/",
    "https://forskning.ruc.dk/da/publications/"\
+ "childbirth-in-dispute-an-introduction-to-the-contested-space/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "molecular-and-medical-biology/",
    "https://forskning.ruc.dk/da/organisations/"\
+ "center-for-everyday-life-of-families-in-the-welfare-state/",
    "https://forskning.ruc.dk/da/publications/"\
+ "a-systematic-review-of-conflict-within-collaborative-governance/"
]

# Create two new lists of URLs for organizations and publications

# Create two lists of organization and publication titles

# Print the values of the lists of organizations and publications
```

Exercise 6: Accumulated savings, revisited

The code cell below creates four lists of initial amounts, annual interest rates, number of years, and currencies

```
initial_amount = [
    18940, 1873, 1234, 10000000, 1236, 98428, 1267345096,
    180, 4820, 128732, 9382620, 8736529, 1827390, 37261524,
    4321, 89403, 287, 182930, 87659, 76
]
```

```
rate = [
    .0541, .0326, .0605, .0153, .0122, .057, .0148, .047,
    .0411, .0737, .0185, .0075, .0414, .0189, .0609, .0711,
    .0255, .0644, .0567, 0.0638
]
years = [
    23, 11, 4, 19, 25, 1, 20, 17, 24, 6,
    16, 6, 10, 16, 9, 22, 13, 4, 7, 3]
currency = [
    "Euro", "Dollar", "Pound", "Yen", "Franc", "Rupee", "Rupee",
    "Franc", "Pound", "Yen", "Dollar", "Euro", "Yen", "Franc",
    "Rupee", "Pound", "Dollar", "Euro", "Rupee", "Dollar"
]
```

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

The exchange rates for Danish Kroner as issued by Danmarks Nationalbank on January 23, 2026 are:

- Euro: 746.86
- US dollars: 636.06
- Pound sterling: 860.34
- Swiss francs: 805.07
- Japanese yen: 4.0216
- Indian rupee: 6.92

You can cam convert the value of an amount in a foreign currencry to Danish Kroner using the formula:

$$DKK = CUR * e/100$$

where DKK is the value in Danish Kroner, CUR is the value in the foreign currency, and e is the exchange rate

Use the code cell below to do the following:

1. Write a `def` function that takes in an initial amount, an interest rate, a number of years, and a currency, and outputs the final amount in Danish Kroner
2. Use a list comprehension, a built-in sequence function, and the list created in the code cell above to create a new list of final amounts rounded to 2 decimal places. Assign the list to the variable `final_amount_DKK`
3. Create a nested dictionary of accumulated savings. The outer dictionary should map the keys 'Scenario 1', ..., 'Scenario 20' to 20 inner dictionaries. Each inner dictionary should map the keys 'Initial amount', 'Rate', 'Years', 'Currency', and 'Final amount' to values at identical index positions from the list defined in the code cell above. Assign the dictionary to the variable `accumulated savings`. There are several ways to do this, but try to use as few lines of code as possible. You can make good use of a combination of the built-in sequence functions `enumerate()` and `zip()`
4. Output (do not print) the value of `accumulated_savings`

```
# Write a function to compute final amount

# Create a list of final amounts in Danish Kroner

# Create a dictionary of accumulated savings

# Output the value of accumulated savings
```