SIT220/731 2024.T3: Task 1P

Python Introduction

Last updated: November 4, 2024

Contents

1 Introduction

This task is related to Module 1; see the *Learning Resources* on the unit site.

This task is due on **Week 3 (Friday)**. However, ideally, you should complete it by the end of **Week 1**. Hence, start tackling it as early as possible. If we find your first solution incomplete or otherwise incorrect, you will still be able to amend it based on the generous feedback we will give you. In case of any problems/questions, do hot hesitate to attend our on-campus/online classes or use the Discussion Board on the unit site.

Submitting after the aforementioned due date will incur a late penalty. This task is part of the **hurdle requirements** in this unit. Not submitting the correct version on time results in failing the unit.

All submissions will be checked for plagiarism. You are expected to work independently on your task solutions. Never share/show parts of solutions with/to anyone.

2 Task

Create a single Jupyter/IPython notebook (see the *Artefacts* section below for all the requirements – read the whole task specification first!), where you perform what follows.

Do not use **`numpy`** nor **`pandas`**. This is an exercise on base Python.

1. Input three Python lists of identical lengths (at least five items each) giving basic data on your friends/family, for example:

```
names   = ["Barbara", "Anne", "Greg", "Maggie", "Nynke"]
heights = [173,       161,    180,    158,      177 ]  # in centimetres
weights = [61,        79,     83,     58,       57  ]  # in kilograms
```

   It is assumed that `names[i]` gives the name of the `i`-th person whose height in centimetres is `heights[i]` and weight in kilograms is `weights[i]`.

   You should enter different data, not exactly the values above.

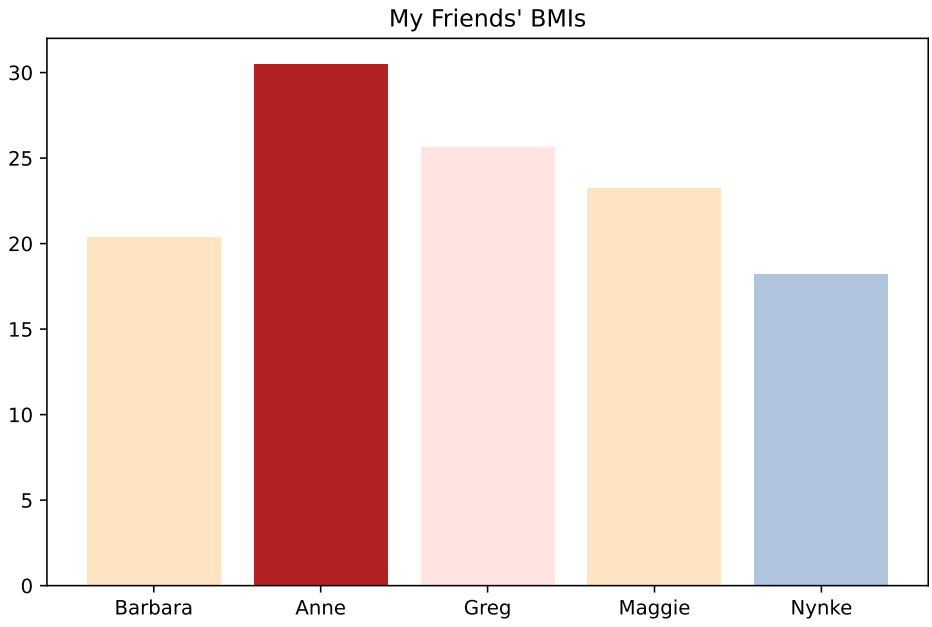
2. Using a *for* loop, create (programmatically) a Python list `bmis` such that `bmis[i]` gives the body mass index of the `i`-th person.

3. Based on the BMI categories as defined by the WHO (underweight if below 18.5, normal range below 25.0, overweight below 30.0, obese otherwise), use a *for* loop to print a series of strings like `"{name} has BMI of {bmi} which is {bmi_category}."`:

```
## Barbara      has BMI of 20.38 which is normal.
## Anne         has BMI of 30.48 which is obese.
## Greg         has BMI of 25.62 which is overweight.
## Maggie       has BMI of 23.23 which is normal.
## Nynke        has BMI of 18.19 which is underweight.
```

   Make sure the formatting is neat and tidy.

4. Draw a bar plot like below, where the bar colours depend on the BMI categories.



Hint: you can find the relevant code example in *Module 1* on our unit site.

5. The Wikipedia article on BMI correctly identifies this index as a very simple measure. In your own words, discuss what are the benefits and limitations of BMI from both the medical and societal perspective, including its possible misuses (write at least three text paragraphs).

Enter `names`, `heights`, and `weights` only once at the beginning of the notebook. Note that your code must work correctly if someone decides to modify these lists: for example, add another person to the database.

Do not hardcode the values of the `bmis` – they must be computed programmatically.

Minimise the printing of unnecessary objects.

3 Additional Tasks for Postgraduate (SIT731) Students (*)

Postgraduate students are **additionally required** to solve/address/discuss what follows. Integrate these new requirements into the above subtasks cleverly (do not create a separate section of the report).

1. For each person, calculate *also* the *New BMI (exponent of 2.5)* measure as defined in https://en.wikipedia.org/wiki/Body_mass_index.
2. *Modify* the textual reporting scheme (subtask 3 above) so that the output strings are like `"{name} has BMI of {bmi} which is {bmi_category}. The new BMI index is {new_bmi}."`.
3. In your own words, discuss the possible advantages and limitations of this measure.

4 Artefacts

The solution to the task must be included in a single Jupyter/IPython notebook (an .ipynb file) running against a Python 3 kernel. The use of G**gle Colab is discouraged. Nothing beats a locally-installed version where you have full control over the environment. Do not become dependent on third-party middlemen/distributors. Choose freedom instead.

Make sure that your notebook has a **readable structure**; in particular, that it is divided into sections. Use rich Markdown formatting (text in dedicated Markdown chunks – not just Python comments).

Do not include the questions/tasks from the task specification. Your notebook should read nicely and smoothly – like a report from data analysis that you designed yourself. Make the flow read natural (e.g., *First, let us load the data on… Then, let us determine… etc.*). Imagine it is a piece of work that you would like to show to your manager or clients — you certainly want to make a good impression. Check your spelling and grammar. Also, use formal language.

At the start of the notebook, you need to provide: the **title** of the report (e.g., *Task 42: How Much I Love This Unit*), your **name**, **student number**, **email address**, and whether you are an **undergraduate (SIT220) or postgraduate (SIT731)** student.

Then, add 1–2 introductory paragraphs (an introduction/abstract – what the task is about).

Before each nontrivial code chunk, briefly **explain** what its purpose is. After each code chunk, **summarise and discuss the obtained results** (in a few sentences).

Conclude the report with 1–2 paragraphs (summary/discussion/possible extensions of the analysis etc.).

Limitations of the OnTrack ipynb-to-pdf renderer:

Ensure that your report as seen in OnTrack is aesthetic (see *Download submision PDF* after uploading the .ipynb file). The OnTrack ipynb-to-pdf renderer is imperfect. We work with what we have. Here are the most common Markdown-related errors.

- Do not include any externally loaded images (via the `![label](href)` Markdown command), for they lead to upload errors.
- Do not input HTML code in Markdown.
- Make sure you leave one blank line before and after each paragraph and bullet list. Do not use backslashes at the end of the line.
- Currently, also *LaTeX formulae* and Markdown tables are not recognised. However, they do not lead to any errors.

Checklist:

1. Header, introduction, conclusion (Markdown chunks).
2. Text divided into sections, all major code chunks commented and discussed in your own words (Markdown chunks).
3. Every subtask addressed/solved. In particular, all reference results that are part of the task specification have been reproduced (plots, computed aggregates, etc.).
4. The report is readable and neat. In particular:
    - all code lines are visible in their entirety (they are not too long),
    - code chunks use consecutive numbering (select *Kernel - Restart and Run All* from the Jupyter menu),
    - rich Markdown formatting is used (`# Section Title`, `* bullet list`, `1. enumerated list`, `| table |`, `*italic*`, etc.),
    - the printing of unnecessary/intermediate objects is minimised (focus on reporting the results specifically requested in the task specification).

Submissions which do not *fully* (100%) conform to the task specification *on* the cut-off date will be marked as FAIL.

Good luck!

5 Intended Learning Outcomes

ULO	Is Related?
ULO1 (Data Processing/Wrangling)	YES
ULO2 (Data Discovery/Extraction)	
ULO3 (Requirement Analysis/Data Sources)	
ULO4 (Exploratory Data Analysis)	
ULO5 (Data Privacy and Ethics)	