Assignment 1

COMP 208 Fall 2021

posted:	Sept 20, 2021 23:59
due:	Sept 28, 2021 23:59

Important Instructions

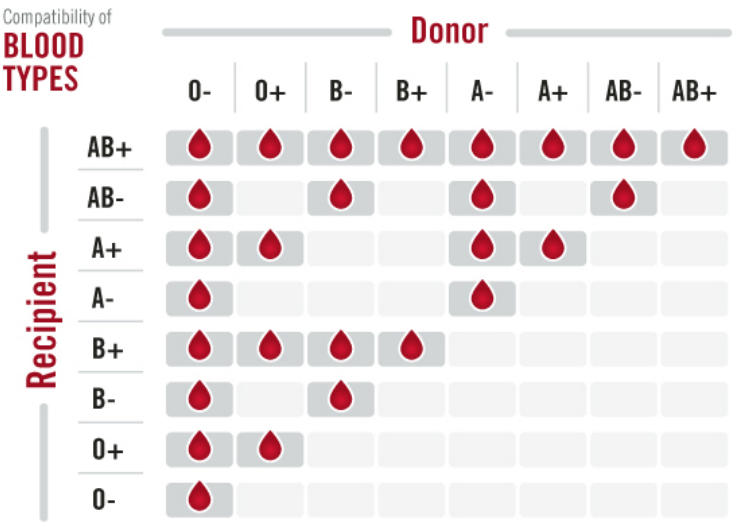
- Download the Python files you will need for this assignment; they are available on myCourses along with this PDF. For each question in this assignment, write your code in the appropriate file. Do not change any code that may already be present in these Python files; it may lead to errors on codePost and you may lose points.
- Submit your files on codePost (https://codepost.io/), and not on myCourses, email or Ed Discussion.
- Write your name and student ID at the top of each program.
- **You must do this assignment individually.** Do not use any pieces of code you did not write yourself.
- **Multiple submissions** are allowed on codePost until due date (plus 48 hours, see late submission policy below).
- **Late submission** will be accepted up to 2 days (48 hours) after the due date and will be penalized by 10% per day. Note that submitting 30 minutes late is the same as submitting 23 hours late. Late penalty will apply irrespective of the reason, be it wrong file submitted, laptop issues or any such reasons.
- **Grading.** For each question, your program will be automatically tested on a variety of test cases which will be run after due date. The test cases you will see on codePost before due date do not have any points; they are only there to help you avoid simple errors.
  - For each question, **up to 20% of the mark may be deducted** by the TA for poor code style: (i) not naming variables descriptively when appropriate; (ii) no comments in the program; (iii) very long lines or too many blanks lines; (iv) unnecessarily complicated or duplicated code. Refer to the style guide on myCourses for further details.
  - Make sure that your code runs. i.e. there are no Syntax Errors. In general, code with errors will receive a very low mark.

Question 1 — Blood type compatibility [25 points]

Required knowledge: if-statement, string indexing, string slicing

Filename: `blood_types.py`

Human blood types are characterized by the antigen group (either A, B, AB, or O), and by the rhesus, either positive (+) or negative (-). Blood transfusions between a donor and a recipient are only possible if the donor blood type is compatible with the recipient blood type. The compatibility table is shown below.



Write a function `is_compatible(donor, recipient)` which returns `True` if `donor` blood type is compatible with `recipient` blood type.

A very long approach, which should be avoided, is to simply follow the compatibility table and write conditions for each pair of donor and recipient values. A better approach would be to consider compatibility of antigen and rhesus separately as follows:

- Donor antigen and recipient antigen are incompatible only if they are different, with the exceptions that (1) recipient antigen AB is compatible with all donor antigens and (2) donor antigen O is compatible with all recipient antigens.
- Donor rhesus and recipient rhesus are incompatible if donor rhesus is positive (+) and recipient rhesus is negative (-).
- Finally, the blood types are compatible only if both antigen and rhesus are compatible.

You can assume that arguments to the function will always be strings and values will always be valid blood types.

Hint: You can use string indexing, slicing and len() function to separate blood type into antigen and rhesus. e.g. the string `"AB+"` into two strings `"AB"` and `"+"`.

Examples

```
print(is_compatible("O+", "AB-"))  # False
print(is_compatible("O+", "A+"))  # True
print(is_compatible("A-", "AB-"))  # True
```

Question 2 — Computing π [25 points]

Required knowledge: for loop, number arithmetic

Filename: `leibniz.py`

The Leibniz formula for π is an infinite series that can be used to compute the value of π. It is given by:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - ...$$

We can compute an approximate value of π by adding together a finite number of terms from the above series, and then multiplying the sum by 4. The more terms we add, the better the approximation will be.

Your task is to write a function `compute_pi` that takes the number of steps **n** as an argument, computes and returns an approximate value of π by adding together **n** terms of the series. For example, when **n** is 4, the approximate value will be:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7}\right) = 4(0.7238095238095239) = 2.8952380952380956$$

And when **n** is 100, the approximation improves to be 3.1315929035585537.

Examples

```
print(compute_pi(1))  # 4.0
print(compute_pi(4))  # 2.8952380952380956
print(compute_pi(100))  # 3.1315929035585537
```

Note on precision: It is okay if value of π computed by your program does not exactly match with the examples. Your output need to be correct only up to 8 digits after the decimal point.