National University of Singapore
School of Computing
CS1010S: Programming Methodology
Semester I, 2024/2025

Mission 6
3-station IPPT

Release date: 4th November 2024
Due: 17th November 2024, 23:59

Required Files

- `mission06-template.py`
- `ippt.py`
- `pushup.csv`
- `situp.csv`
- `run.csv`
- `ippt_takers_data.csv`

Background

Every year, able-bodied Singaporean males who are in or have completed compulsory military training (a.k.a. National Service) have to take the Individual Physical Proficiency Test (IPPT). The IPPT is a physical fitness test meant to assess the fitness of our soldiers to ensure that our soldiers are physically fit in times of uncertainty.

In 2014, the Ministry of Defence announced a new test format and scoring system for IPPT, which was implemented in 2015. The new IPPT consists of the following 3 stations:

- **Push-up**, which measures upper body strength
- **Sit-up**, which measures abdominal strength
- **2.4km Run**, which measures cardiovascular fitness and lower body strength

Soldiers participating in the 3-station IPPT will first attempt as many push-ups and sit-ups that they can do in one minute each. Depending on their age, they are given points based on the number of repetitions they have completed. Next, they have to complete a 2.4km run, and would be given points depending on how fast they complete the run.

The final IPPT score is the sum of all the scores obtained at the 3 stations, which determines if they have passed or failed the IPPT. To reward and motivate soldiers to maintain a high level of fitness, a monetary reward is given for soldiers who have demonstrated a high level of physical fitness. The reward amount depends on the final score obtained, and can be broken down into Pass, Pass with Incentive, Silver, and Gold. This is known as the IPPT Award.

In this mission, you will learn to read data from a file, and use the data to determine a soldier's IPPT score and award. You will also learn to use the table data structure to store and access the data.

This mission consists of **7** tasks.

IPPT Data

The IPPT scoring criteria is provided to you in 3 different files:

- Push-ups: `pushup.csv`
- Sit-ups: `situp.csv`
- 2.4km Run: `run.csv`

Push-up / Sit-up

Push-ups and sit-ups in the IPPT are scored in terms of the number of repetitions that a soldier can do in one minute. `pushup.csv` and `situp.csv` lists the scores that a soldier of a certain age will obtain if they managed a particular number of repetitions.

The first row describes the number of repetitions, whereas the first column lists the age of the soldier.

From the file `situp.csv`, we observe that an 18-year-old who did 10 sit-ups would get 0 points (green box), whereas a 40-year-old who did 10 sit-ups would get 3 points (yellow box).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	AGE/REP	1	2	3	4	5	6	7	8	9	10	11	12
2	18	0	0	0	0	0	0	0	0	0	0	0	0
3	19	0	0	0	0	0	0	0	0	0	0	0	0
4	20	0	0	0	0	0	0	0	0	0	0	0	0
5	21	0	0	0	0	0	0	0	0	0	0	0	0
6	22	0	0	0	0	0	0	0	0	0	0	0	0
7	23	0	0	0	0	0	0	0	0	0	0	0	0
8	24	0	0	0	0	0	0	0	0	0	0	0	0
9	25	0	0	0	0	0	0	0	0	0	0	0	0
10	26	0	0	0	0	0	0	0	0	0	0	0	0
11	27	0	0	0	0	0	0	0	0	0	0	0	0
12	28	0	0	0	0	0	0	0	0	0	0	0	1
13	29	0	0	0	0	0	0	0	0	0	0	0	1
14	30	0	0	0	0	0	0	0	0	0	0	0	1
15	31	0	0	0	0	0	0	0	0	0	0	1	2
16	32	0	0	0	0	0	0	0	0	0	0	1	2
17	33	0	0	0	0	0	0	0	0	0	0	1	2
18	34	0	0	0	0	0	0	0	0	0	1	2	3
19	35	0	0	0	0	0	0	0	0	0	1	2	3
20	36	0	0	0	0	0	0	0	0	0	1	2	3
21	37	0	0	0	0	0	0	0	0	1	2	3	4
22	38	0	0	0	0	0	0	0	0	1	2	3	4
23	39	0	0	0	0	0	0	0	0	1	2	3	4
24	40	0	0	0	0	0	0	0	1	2	3	4	5
25	41	0	0	0	0	0	0	0	1	2	3	4	5
26	42	0	0	0	0	0	0	0	1	2	3	4	5
27	43	0	0	0	0	0	0	1	2	3	4	5	6

2.4km Run

The 2.4km run in the IPPT is scored based on the timing a soldier takes to complete a 2.4km run. To simplify things, we will represent the time of the run in seconds. `run.csv` lists the score that a soldier of a certain age will obtain if they complete the run under a particular timing.

The first row describes the time in seconds, while the first column lists the age of the soldier.

A different score is given whenever there is a timing difference of 10 seconds. From the run.csv file, we observe that an 18-year-old who ran 08:30 (510 seconds) or faster would get 50 points and one who ran between 08:31 (511 seconds) and 08:40 (520 seconds) inclusive would get 49 points.

The data file shows the scores for the timing between 510 seconds to 1110 seconds. Naturally, a timing less than 510 seconds would get 50 points, and a timing exceeding 1110 seconds would get 0 points.

Administrivia

- The following functions have been provided to you:
  - `make_ippt_table(pushup_table, situp_table, run_table)`
  - `get_situp_table(ippt_table)`
  - `get_pushup_table(ippt_table)`
  - `get_run_table(ippt_table)`
- The required packages have been imported for you in the template file. You are **not allowed** to import additional packages.

Table Data Structure

To help you manipulate and access the data in a tabular fashion, we have provided an implementation of a table data structure which would help you access the scores in the IPPT table.

The functions that are provided for the `table` data structure:

- `create_table(data, row_keys, col_keys)`
- `access_cell(table, row_key, col_key)`

`create_table` takes in 3 parameters. The first parameter `data` is a tuple of tuples, which contains the table of data. The second parameter `row_keys` is the tuple of keys associated to each row, and the third parameter `col_keys` is the tuple of keys associated to each column.

As an example, consider the following table:

S/N	Name	Gender	Age	Course
1	Wai Hon	M	23	Business Analytics
2	Yang Shun	M	25	Computer Science
3	Xiangxin	F	18	Computer Science
4	Soedar	M	25	Computer Science

The following code sample illustrates how to represent the above data in Python using our own table data structure:

```
user_data = (("Wai Hon", "M", 23, "Business Analytics"),
             ("Yang Shun", "M", 25, "Computer Science"),
             ("Xiangxin", "F", 18, "Computer Science"),
             ("Soedar", "M", 25, "Computer Science"))

user_table = create_table(user_data,
                          (1, 2, 3, 4),
                          ("Name", "Gender", "Age", "Course"))
```

`access_cell`, is a general accessor which would retrieve a particular cell from a `table` given a `row_key` and a `column_key`. Compare the returned value of the execution of the `access_cell` functions below with the table above.

```
access_cell(user_table, 1, "Course")    # Business Analytics
access_cell(user_table, 2, "Age")       # 25
access_cell(user_table, 3, "Name")      # Xiangxin
access_cell(user_table, 4, "Gender")    # M
```

Finally, you may find the higher order functions `map` and `filter` to be useful in solving some of these tasks.

Task 1: Read Data (4 marks)

Implement `read_data`, a function that would read the input data file, and return a table of scores for a particular station.

Sample Execution:

```
situp_table = read_data("situp.csv")
pushup_table = read_data("pushup.csv")
run_table = read_data("run.csv")

# Sit-up score of a 24-year-old who did 10 sit-ups.
access_cell(situp_table, 24, 10)    # 0

# Push-up score of a 18-year-old who did 30 push-ups.
access_cell(pushup_table, 18, 30)   # 16

# Run score of a 30-year-old who ran 12 minutes (720 seconds)
access_cell(run_table, 30, 720)     # 36

# Since our run.csv file does not have data for 725 seconds, we should
# get None if we try to access that cell.
access_cell(run_table, 30, 725)     # None
```

Task 2: Custom Accessors (6 marks)

The default accessor function, `access_cell`, is a general accessor which retrieves a particular cell given a `row_key` and a `column_key`. As such, it is not aware of the type of data

the table contained, and would not return the correct score if the `column_key` does not exists in our data.

We would like to create a custom accessor for each station. When the `column_key` does not exist, the cell with the closest `column_key` value will be returned. For example, `pushup_score(pushup_table, 18, 61)` returns 25 which is the score in cell (18, 60) because 61 doesn't exist and the closest column is 60.

Both push-up and sit-up tables have column keys ranging from 1 to 60. The run table has column keys ranging from 510 to 1110. Moreover, the score data are in an interval of 10 seconds, we are not able to access the run score for a timing that falls within the 10 seconds interval. For example, a 30-year-old male who has a run timing of 735 seconds will be considered under the band of 740 seconds and will get 35 points (timing is rounded up to the nearest band). Using a custom accessor would allow us to retrieve the correct score.

Create new accessors for each of the stations. You may assume that the inputs are guaranteed to be non-negative integers and that all given `row_key` will be in the valid range.

Sit-up and Push-up

```
situp_score(situp_table, 24, 0)          # 0
pushup_score(pushup_table, 18, 61)       # 25
pushup_score(pushup_table, 18, 70)       # 25
```

2.4km Run

```
run_score(run_table, 30, 720)            # 36
run_score(run_table, 30, 725)            # 35
run_score(run_table, 30, 735)            # 35
```

Task 3: Calculate IPPT Award (2 marks)

An IPPT Award is awarded based on the total score that a soldier has obtained for all the 3 stations. We will use the following table to determine the IPPT Award.

Award Name	Award String	Total Score
Fail	`"F"`	< 51
Pass	`"P"`	≥ 51
Pass with Incentive	`"P$"`	≥ 61
Silver	`"S"`	≥ 75
Gold	`"G"`	≥ 85

Define a function, `ippt_award`, which will return the Award String for a given IPPT Score

```
ippt_award(50)           # F
ippt_award(51)           # P
ippt_award(61)           # P$
ippt_award(75)           # S
ippt_award(85)           # G
```

Task 4: Calculate IPPT Results (2 marks)

We can now calculate the IPPT score and the qualifying award of a soldier's test attempt. The result consists of both the total points and the award, and is represented as a tuple of `(Total IPPT Score, IPPT Award String)`.

Implement the function `ippt_results(ippt_table, age, pushup, situp, run)`, which will calculate the total IPPT score and award given the age of the soldier, number of sit-ups, number of push-ups, and 2.4km run timing. Remember what has been provided in the **Administrivia**.

```
ippt_results(ippt_table, 25, 30, 25, 820)          # (53, 'P')
ippt_results(ippt_table, 28, 56, 60, 530)          # (99, 'G')
ippt_results(ippt_table, 38, 18, 16, 950)          # (36, 'F')
ippt_results(ippt_table, 25, 34, 35, 817)          # (61, 'P$')
ippt_results(ippt_table, 60, 70, 65, 450)          # (100, 'G')
```

IPPT Takers Data

The provided dataset **ippt_takers_data.csv** contains the following columns:

Name	Age	Push-Ups	Sit-Ups	2.4-Km-Run

Task 5: Parsing IPPT Takers Results (6 marks)

You are given a dataset which has all the data of the IPPT takers. However, `read_csv` takes in fields as strings. Your task is to implement a function **`parse_results(csvfilename)`** that parses the data into the correct data types, and use previously defined functions to compute some fields. **Note:** You should not rewrite any logic that you had already defined prior to this, doing so will result in a deduction of marks.

- **Name** (`str`): A name for each IPPT taker.
- **Age** (`int`): Age of IPPT taker.
- **Push-Ups** (`int`): Number of Push-Ups done.
- **Sit-Ups** (`int`): Number of Sit-Ups done.
- **2.4-Km-Run** (`int`): Number of seconds taken to complete 2.4 Km Run done.
- **Total-Score** (`int`): Total IPPT Score.
- **Award** (`str`): IPPT Award String.

The function **`parse_results(csvfilename)`**, takes in one argument, **`csvfilename`**: a *string* of the filename of the CSV file. Your function should return a *list-of-tuples* with the fields represented in their correct data types as specified in the list above. You should not omit the header row and the data should be parsed in same order it is represented in the data file.

Sample execution:

```
ippt_takers_data = parse_results("ippt_takers_data.csv")
ippt_takers_data[0]
# => ('Name', 'Age', 'Push-Ups', 'Sit-Ups', '2.4-Km-Run',
# 'Total-Score', 'Award')
ippt_takers_data[1]
# => ('Sean Hendricks', 38, 25, 74, 1212, 42, 'F')
ippt_takers_data[2]
# => ('Phillip Brown DDS', 59, 15, 15, 852, 61, 'P$')
ippt_takers_data[3]
# => ('Ryan Gray MD', 24, 45, 78, 1074, 46, 'F')
```

Task 6: Number of IPPT Awards (4 marks)

Implement the function **`num_awards`** that finds the distribution of the IPPT Awards for the given dataset and age. The function takes in two arguments, **`ippt_takers_data`**: a *list-of-tuples* of the IPPT takers data, and **`age`**: an *int* of the age of the IPPT takers. The function should return a *dictionary* of the distribution of the IPPT Awards for the given age. Note: The order of the keys in the dictionary does not matter as long as the keys follow the IPPT Awards in Task 3.

Sample Execution:

```
num_awards(ippt_takers_data, 25)
# => {'F': 56, 'P': 20, 'G': 18, 'P$': 14, 'S': 10}
num_awards(ippt_takers_data, 28)
# => {'F': 54, 'S': 13, 'G': 16, 'P': 8, 'P$': 12}
```

Task 7: Top-K IPPT Scores (4 marks)

Implement the function **`top_k_scores`** that takes three inputs **`ippt_takers_data`**: a *list-of-tuples* of the IPPT takers data, **`age`**: an *int* of the age of the IPPT takers, and **`k`**: an *int* of the number of top IPPT takers to return. The function should return a *list-of-tuples* of the top-**`k`** IPPT takers with the highest IPPT scores, sorted first in descending order of their IPPT scores and then alphabetically. You should return the IPPT taker's **`Name`** and **`Total-Score`** in a tuple.

`Top-k` here means that you should take IPPT takers according to their IPPT score. If there are less than **`k`** IPPT takers, return all the IPPT takers. If the remaining IPPT takers, from the (k + 1)-th and so on, have the same score as the k-th taker, you should also include those IPPT takers as well.

Note: The order of the result is absolute, meaning any difference in the order of the tuples will result in an incorrect judgement.

Sample Execution:

```
top_k_scores(ippt_takers_data, 5, 25)
# => [('Joseph Burns', 100), ('Mike Williams', 98), ('Rachel Serrano', 98),
# ('Margaret Jennings', 97), ('Alexandra Day', 95), ('Stephen Boyer', 95)]
top_k_scores(ippt_takers_data, 1, 28)
# => [('Eric Villegas', 100), ('Melissa Evans', 100)]
top_k_scores(ippt_takers_data, 2, 28)
# => [('Eric Villegas', 100), ('Melissa Evans', 100)]
```