CS1010S — Programming Methodology
National University of Singapore

Re-Practical Examination

27 Apr 2018

Time allowed: 2 hours

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **three** questions. The time allowed for solving this test is **2 hours**.
3. The maximum score of this test is **30 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. The total score for this test is capped at **16 marks** for those attempting the exam for the second time.
5. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
6. While you are also provided with the template `repractical-template.py` to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness using IDLE and not depend only on the provided test cases. Do ensure that you submit your answers correctly by running the test cases at least once.
7. In case there are problems with Coursemology.org and we are not able to upload the answers to Coursemology.org, you will be required to name your file `<mat no>.py` where `<mat no>` is your matriculation number and leave the file on the Desktop. If your file is not named correctly, we will choose any Python file found at random.
8. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology and correctly left in the desktop folder at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Practical Exam, or the parts that are not uploaded correctly.
9. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.

GOOD LUCK!

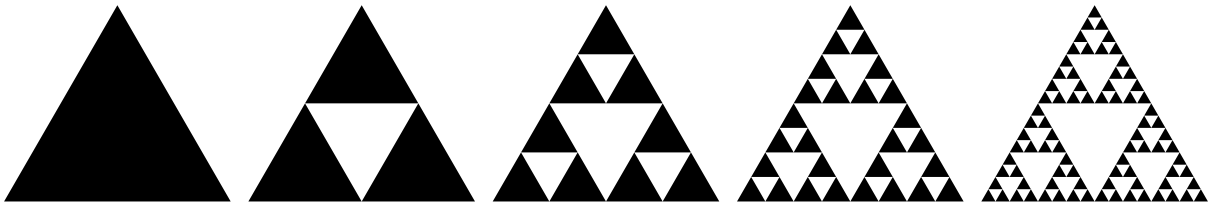
Question 1 : Sierpinski Triangle [10 marks]

The Sierpinski triangle (also with the original orthography Sierpiński), also called the Sierpinski gasket or the Sierpinski Sieve, is a fractal and attractive fixed set with the overall shape of an equilateral triangle, subdivided recursively into smaller equilateral triangles. Originally constructed as a curve, this is one of the basic examples of self-similar sets, i.e., it is a mathematically generated pattern that can be reproducible at any magnification or reduction. It is named after the Polish mathematician Wacław Sierpiński, but appeared as a decorative pattern many centuries prior to the work of Sierpiński. *Source: Wikipedia*

To construct a Sierpinski triangle, we start with an equilateral and repeatedly remove triangular subsets:

1. Start with an equilateral triangle
2. Subdivide it into four smaller congruent equilateral triangles and remove the central triangle.
3. Repeat step 2 with each of the remaining smaller triangles forever.

The following diagram shows the Sierpinski triangles of level n, i.e. with the removal applied n times.



0 1 2 3 4

A. Implement the function `num_triangles` that takes in a non-negative integer n, and returns the number of triangles found in a Sierpinski triangle of level n. Note we count both black and white triangles as well as any larger triangles made up of smaller ones. [4 marks]

Sample execution:

```
>>> num_triangles(0)
1

>>> num_triangles(1)
5

>>> num_triangles(2)
17

>>> num_triangles(3)
53

>>> num_triangles(4)
161
```

B. Implement the function `area` that takes in a non-negative integer n, and returns the ratio of the total area covered by the black triangles, to the largest equilateral triangle of a level n Sierpinski triangle. For example, a level 1 Sierpinski triangle, the black area makes up 0.75 of the total triangle area. [3 marks]

Sample execution:

```
>>> area(0)
1

>>> area(1)
0.75

>>> area(2)
0.5625

>>> area(3)
0.421875

>>> area(4)
0.31640625
```

C. We can also draw a Sierpinski triangle with numbers like a Pascal's triangle.

```
          1
        1 1
      1 2 1
    1 3 3 1
  1 4 6 4 1
1 5 10 10 5 1
```

Pascal's Triangle

```
          1
        1 1
      1 0 1
    1 1 1 1
  1 0 0 0 1
1 1 0 0 1 1
```

Sierpinski Triangle

Note that the 1's in the Sierpinski triangle simply denotes the odd numbers in the Pascal's triangle.

Implement the function `row` that takes in a non-negative integer n, and returns the nth row of the Sierpinski triangle as a list of digits 1 and 0. [3 marks]

Sample execution:

```
>>> row(0)
[1]

>>> row(1)
[1, 1]

>>> row(2)
[1, 0, 1]
```

```
>>> row(3)
[1, 1, 1, 1]

>>> for i in range(16):
        print(" "*(15-i), *row(i))
               1
              1 1
             1 0 1
            1 1 1 1
           1 0 0 0 1
          1 1 0 0 1 1
         1 0 1 0 1 0 1
        1 1 1 1 1 1 1 1
       1 0 0 0 0 0 0 0 1
      1 1 0 0 0 0 0 0 1 1
     1 0 1 0 0 0 0 0 1 0 1
    1 1 1 1 0 0 0 0 1 1 1 1
   1 0 0 0 1 0 0 0 1 0 0 0 1
  1 1 0 0 1 1 0 0 1 1 0 0 1 1
 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Question 2 : [10 marks]

Important note: You are provided with a data file `wind.csv` for this question for testing, but your code should work correctly for *any* data file with the same format and it *will* be tested with other data files.

You have found some part-time work at the Meteorological Service of Singapore and are tasked to analyse the historical wind data that was collected at various places around Singapore.

The data file given contains the daily average as well as highest temperature. There are some station with some dates missing the wind information. Such entries would have already been removed so while you can expect valid data in the csv file, you should not assume that the values are contiguous or complete.

A. Implement the function `monthly_max` which takes in the filename, a location (string) and a year (int). It returns for each month, the average daily maximum temperature as a dictionary with the keys being months, and the values is the average of the maximum daily temperatures recorded rounded to 2 decimal places. [5 marks]

Sample execution:

```
>>> monthly_max("wind.csv", "Changi", 2013)
{'Jan': 32.51, 'Feb': 29.63, 'Mar': 31.86, 'Apr': 33.29,
 'May': 30.48, 'Jun': 30.54, 'Jul': 30.91, 'Aug': 33.98,
 'Sep': 33.22, 'Oct': 32.56, 'Nov': 28.88, 'Dec': 30.79}

>>> monthly_max("wind.csv", "Paya Lebar", 2015)
{'Jan': 38.35, 'Feb': 39.59, 'Mar': 36.0, 'Apr': 33.06,
 'May': 32.39, 'Jun': 31.9, 'Jul': 37.14, 'Aug': 33.64,
 'Sep': 30.51, 'Oct': 27.75, 'Nov': 29.11, 'Dec': 36.84}

>>> monthly_max("wind.csv", "Marina Barrage", 2018)
{'Jan': 31.09, 'Feb': 33.93, 'Mar': 31.85}
```

B. Implement the function `windest_location` which takes in the filename, and a year (int). It returns for each month in the year, the location with the highest average daily mean as a dictionary with the keys being months, and the values a tuple pair of the location and the value rounded to 2 decimal places. [5 marks]

Sample execution:

```
>>> windest_location("wind.csv", 2015)
{'Jan': ('Paya Lebar', 18.0), 'Feb': ('Paya Lebar', 18.84),
 'Mar': ('Paya Lebar', 13.91), 'Apr': ('Paya Lebar', 10.44),
 'May': ('East Coast Parkway', 10.87),
 'Jun': ('East Coast Parkway', 13.03),
 'Jul': ('East Coast Parkway', 15.75),
 'Aug': ('East Coast Parkway', 14.66),
 'Sep': ('East Coast Parkway', 13.01),
 'Oct': ('East Coast Parkway', 10.06),
 'Nov': ('Paya Lebar', 8.36), 'Dec': ('Paya Lebar', 11.73)}
```

```
>>> windest_location("wind.csv", 2017)
{'Jan': ('Marina Barrage', 16.34), 'Feb': ('Marina Barrage', 18.16),
 'Mar': ('Marina Barrage', 13.65), 'Apr': ('Marina Barrage', 12.46),
 'May': ('East Coast Parkway', 11.67),
 'Jun': ('East Coast Parkway', 11.67),
 'Jul': ('East Coast Parkway', 14.9),
 'Aug': ('East Coast Parkway', 13.97),
 'Sep': ('East Coast Parkway', 11.93),
 'Oct': ('East Coast Parkway', 10.75),
 'Nov': ('East Coast Parkway', 10.78),
 'Dec': ('East Coast Parkway', 11.02)}

>>> windest_location("wind.csv", 2018)
{'Jan': ('Paya Lebar', 12.88), 'Feb': ('Paya Lebar', 20.0),
 'Mar': ('Paya Lebar', 13.64)}
```

Question 3 : Pacific Rim [10 marks]

Warning: Please read the entire question carefully and plan well before starting to write your code.

— BACKGROUND (Okay to skip) —

In 2013, huge alien sea monsters called Kaiju emerge from an interdimensional portal called the Breach at the bottom of the Pacific Ocean. Over the course of three years, the Kaiju wreck havoc upon coastal cities along the Ring of Fire, such as San Francisco, Cabo San Lucas, Sydney, Manila, and Hong Kong. Humanity responds by constructing massive robotic machines called Jaegers to combat the Kaiju threat. Each Jaeger is piloted by two or more people, who are connected by a neural bridge in a process called "drifting" to share the mental stress of piloting the machine.

In 2020, the Jaeger Gipsy Danger, piloted by brothers Raleigh and Yancy Beckett, defends Anchorage from the Kaiju named Knifehead. They both engage in combat, with Gipsy gaining the upper hand and defeating Knifehead by blasting it with its plasma cannon. Unknown to them, Knifehead is still alive. Knifehead then re-emerges, ambushes them, critically damages Gipsy Danger by severing its left arm, and throws Yancy out of the cockpit, killing him. Raleigh manages to pilot Gipsy alone, activating the remaining plasma cannon and finally killing Knifehead. Traumatized by Yancy's death, Raleigh quits the Jaeger program.

In 2025, the world governments decide to end funding for the continuous construction of Jaegers, in favor of building massive coastal walls, as the Kaiju are growing more powerful, their attacks more frequent, and Jaegers are being destroyed faster than they are being built. The remaining four Jaegers are relocated to Hong Kong under the command of Marshal Stacker (Eric.) Pentecost, who plans to end the Kaiju War by destroying the Breach using a nuclear weapon. However, the wall is revealed to be a failure, as a Kaiju named Mutavore easily breaks through the wall at Sydney before being killed by the Australian Jaeger, Striker Eureka.

Pentecost tracks down Raleigh at a wall-construction site in Alaska and persuades him to rejoin the program. Traveling to the Hong Kong base, the Shatterdome, Raleigh is introduced to Mako Mori, director of the Jaeger restoration program and Pentecost's adoptive daughter. Four Jaegers remain in operation âĂŞ the refurbished Gipsy Danger, the Russian Cherno Alpha, the Chinese Crimson Typhoon, and Striker Eureka, piloted by father and son Herc and Chuck Hansen, and intended to deliver the nuclear weapon to seal the Breach.

To find a new co-pilot, Raleigh participates in tryouts, assuming Mako is "drift-compatible", despite Pentecost's protests. During their first drifting test, Raleigh is distracted by the memory of Yancy's death. Mako, in turn, is lost in the memory of the Kaiju attack on Tokyo that orphaned her, and nearly fires Gipsy's plasma cannon. Mako is promptly relieved of piloting duties, and Raleigh confronts Pentecost, the former Jaeger pilot who fought and killed the Kaiju that attacked Tokyo and adopted Mako after the battle, telling him that he is being overprotective of her.

Pentecost consults Kaiju experts Newton Geiszler and Hermann Gottlieb. Her-

mann claims that the Breach will stabilize and the Kaiju will increase in number, but it will allow the assault to succeed, while Newton suggests drifting with a Kaiju's brain to learn more about them. Newton goes ahead with his plan despite Hermann's protests and discovers that the Kaiju are actually bioweapons grown by alien colonists, that live in the dimension on the other side of the breach called the Anteverse, who plan on wiping out humanity. With Pentecost's permission, Newton searches for black market dealer Hannibal Chau, who sells body parts of dead Kaiju, to obtain a fresh Kaiju brain to drift with. However, he soon figures out that since drifting is a two-way link, the Kaiju hive mind gained access to his knowledge just as he did theirs. Soon after, two new Kaijus, Leatherback and Otachi, emerge simultaneously to find Newton.

All Jaegers except Gipsy Danger are dispatched to intervene. Otachi destroys Crimson Typhoon, while Leatherback destroys Cherno Alpha and paralyzes Striker Eureka with an EMP blast. Otachi then heads out to find Newton, destroying much of Hong Kong in its wake. Out of options, Pentecost reluctantly allows Mako and Raleigh to pilot Gipsy Danger. Gipsy is then deployed to fight the two kaiju, first taking on Leatherback, who stayed near the deactivated Striker. Gipsy kills Leatherback by blowing it apart with its plasma casters, then follows Otachi into the middle of Hong Kong where they narrowly manage to save Newton from being eaten. Otachi takes flight and tries to carry Gipsy into the atmosphere, but Gipsy manages to kill Otachi midair with its sword, and plummets back to earth unharmed.

Examining Otachi's body, Newton and Hannibal find out that it is pregnant. The infant Kaiju bursts out and chases Newton, but seemingly dies shortly after birth: when Hannibal approaches to investigate, it springs back to life and eats him whole before choking on its umbilical cord and dying for good. Now with an accessible Kaiju brain, Newton and Hermann drift with the infant's brain, discovering that the Breach can only open in the presence of a Kaiju's DNA.

As Herc was injured during the previous fight, Pentecost, who is dying from radiation sickness resulting from piloting a Jaeger built with a nuclear core before the implementation of radiation shielding, pilots Striker Eureka with Chuck. Along with Gipsy Danger, they approach the Breach. Three Kaijus emerge from the Breach to defend it: Raiju, Scunner, and Slattern, who is the largest one ever encountered. Raiju manages to cripple Gipsy but is quickly killed, while Striker is immobilized by Slattern, who calls to Scunner for aid. Pentecost and Chuck decide to detonate the bomb, as they are easily overwhelmed, depending on Gipsy's nuclear reactor to seal the Breach. The explosion kills Scunner, however Slattern survives, so Gipsy grapples it, burns it to death using its reactor and, considering Newton and Hermann's discovery, rides its corpse into the Breach. As the Jaeger reaches the other side, Raleigh successfully overloads the reactor and ejects Mako and himself using escape pods. The reactor explodes, killing a few nearby alien creatures and destroying the Breach, while Raleigh and Mako's escape pods surface in the Pacific Ocean. Herc, now the Marshal, orders the war clock to stop, indicating mankind's victory, and Raleigh and Mako embrace as rescue helicopters arrive.

In a mid-credits scene, Hannibal, revealed to have survived being swallowed, cuts

his way out of the Kaiju infant and asks where his lost shoe is.

– Source: Wikipedia

— END OF BACKGROUND —

For this question, we will be modelling Jaegers, which are giant fighting robots, and their pilots.

A *Pilot* has a name and some compatibility threshold (which is an integer). It takes more than one pilot to operate a *Jaeger*, and the pilots operating have to undergo training together to achieve *drift compatibility*.

Every time a pair of pilots train together, their mutual compatibility goes up by 1 unit. In order to achieve drift compatibility, the mutual compatibility of all pilots must be greater or equal to the pilot with the highest compatibility threshold.

For example, if pilots Raleigh has a compatibility level of 3 and Mako has a compatibility level of 5, the two of them must have trained at least 5 times together before being able to drift together. It those not matter if any of them have trained with other pilots before.

The class `Pilot` is initialized with a name (a string) and a compatibility level (an int) and supports the following methods:

- `train(partner)` takes in another `Pilot` and trains with him/her. If the other pilot is the pilot him/herself, then the string `'<pilot name> cannot train with self'` is returned.

  Otherwise, the mutual compatibility between these two pilots increases by one and the string `'<pilot name> trains with <partner name>'` is returned.

- `show_partners()` takes no inputs, and returns a tuple of pairs, with each pair containing the name of a partner pilot and the mutual compatibility level with the pilot. The tuple should contain all the partners that the pilot has trained with.

- `board(Jaeger)` takes as input a `Jaeger`. It returns the string `'<pilot name> is already on <Jaeger name>'` if the pilot is currently on board another Jaeger. Otherwise, the pilot boards the Jaeger and the string `'<pilot name> boards <Jaeger name>'` is returned.

- `alight()` takes no inputs. It returns the string `'<pilot name> is not on a Jaeger'` if the pilot is not currently on board a Jaeger. Otherwise, the pilot alights from the current Jaeger and the string `'<pilot name> alights from <Jaeger name>'` is returned.

The class `Jaeger` is initialized with a name (a string) and supports one method `drift()`. The method takes no inputs and returns a string based on the following conditions:

- `'<Jaeger name> has insufficient pilots'`, if there are fewer than 2 pilots on board.
- `'<pilot 1 name> and <pilot 2 name> are not compatible'`, if there exists any pair of pilots on board whose mutual compatibility level is lower than either of their compatibility thresholds.
- Otherwise, `'Drift successful. <Jaeger name> is operational'` is returned.

Provide an implementation for the classes `Pilot` and `Jaeger`.

For simplicity, you do not have to worry about data abstraction and can access the properties of both classes directly. Take careful note of the characters in the returned strings, especially spaces and punctuation.

Sample Execution:

```
raleigh = Pilot("Raleigh", 3)
yancy = Pilot("Yancy", 2)
mako = Pilot("Mako", 5)

gipsy = Jaeger("Gipsy Danger")
crimson = Jaeger("Crimson Typhoon")

>>> raleigh.show_partners()
()

>>> raleigh.train(yancy)
"Raleigh trains with Yancy"

>>> raleigh.show_partners()
(('Yancy', 1),)

>>> yancy.show_partners()
(('Raleigh', 1),)

>>> yancy.train(raleigh)
"Yancy trains with Raleigh"

>>> raleigh.board(gipsy)
"Raleigh boards Gipsy Danger"

>>> yancy.board(gipsy)
"Yancy boards Gipsy Danger"

>>> gipsy.drift()
"Raleigh and Yancy are not compatible"

>>> raleigh.train(yancy)
"Raleigh trains with Yancy"

>>> gipsy.drift()
"Drift successful. Gipsy Danger is operational"

>>> yancy.board(crimson)
"Yancy is already on Gipsy Danger"

>>> yancy.alight()
"Yancy alights from Gipsy Danger"
```

```
>>> gipsy.drift()
"Gipsy Danger has insufficient pilots"

>>> mako.alight()
"Mako is not on a Jaeger"

>>> mako.board(gipsy)
"Mako boards Gipsy Danger"

>>> gipsy.drift()
"Raleigh and Mako are not compatible"

>>> mako.train(raleigh)
"Mako trains with Raleigh"

>>> mako.show_partners()
(('Raleigh', 1),)

>>> raleigh.show_partners()
(('Yancy', 3), ('Mako', 1))

>>> mako.train(raleigh)
"Mako trains with Raleigh"

>>> mako.train(raleigh)
"Mako trains with Raleigh"

>>> mako.train(raleigh)
"Mako trains with Raleigh"

>>> mako.train(raleigh)
"Mako trains with Raleigh"

>>> gipsy.drift()
"Drift successful. Gipsy Danger is operational"

>>> yancy.board(gipsy)
"Yancy boards Gipsy Danger"

>>> gipsy.drift()
"Mako and Yancy are not compatible"

>>> yancy.train(mako)
"Yancy trains with Mako"

>>> yancy.train(mako)
"Yancy trains with Mako"
```

```
>>> yancy.train(mako)
"Yancy trains with Mako"

>>> yancy.train(mako)
"Yancy trains with Mako"

>>> yancy.train(mako)
"Yancy trains with Mako"

>>> gipsy.drift()
"Drift successful. Gipsy Danger is operational"
```

You are advised to solve this problem incrementally. Even if you cannot fulfil all the desired behaviours, you can still get partial credit for fulfilling the basic behaviours.

— E N D O F P A P E R —