Presentation exercises – Week 5

COMP 202, Winter 2022

Presentation date: Friday, February 11^{th}

Presentation exercise 1

It can be difficult, especially of late, to find a study space at McGill. Everywhere you look, rooms seem to be reserved and/or occupied. In this question we will write a study space finding simulator to simulate people trying to find a room.

First, write a function `roll_die()`. It takes no inputs and should return a randomly generated number between 1 and 6.

Then, write a function `is_available()` that takes three inputs: a string corresponding to a building, a string corresponding to a room number, and an hour of the day (an integer from 1 to 24). The function should return `True` if the given room (located in the given building) is available at the specified hour, and `False` otherwise.

A room is not available if any of the following conditions are met:

- if the room is in the McLennan or Redpath Library buildings.
- if the hour is before 8 am or after 12 pm.

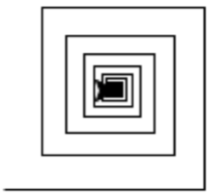
A room is available if none of the above are true and **all** of the following conditions are met:

- if the hour is between 8 am and 12 pm
- if, after rolling 3 dice, the sum of the three rolled numbers is 5 or less.

Then, write a function `find_available()` that takes no inputs. It asks the user to input a building, room number and hour of day (from 1 to 24), and then tells them if the room is available or not. If the room is not available, ask them to enter another room, building and hour. Keep asking until the room that they entered is available. At the end, inform them how many tries it took until they were able to find an available room.

Presentation exercise 2

Write a function `spiral(initial_length, angle, multiplier)` that uses Turtle to create a spiral drawing. The first line segment of the spiral should have length `initial_length` (an integer). The length of each succeeding line segment should be multiplied by the `multiplier` (a float) (thus each segment's length will be different from the last). Further, each two line segments should form angles of `angle` degrees. The spiral should end when the length of a line segment is less than 1 or greater than 1000. For example, calling `spiral(100, 90, 0.9)` should produce something like the following:



If you would like to make the spiral look a bit nicer, you could try to draw each line segment (or each pair of line segments) in a different color.

Presentation exercise 3

For this question you must work in a pair and present together. Make sure to evenly contribute to both the code and the presentation. The presentation should last 8-10 minutes (approx. 5 minutes per person).

A polygonal number is a number that can be represented by dots arranged in the shape of a regular polygon. For example, 10 is a triangular number, 16 is a square number, and 36 is both a triangular and a square number[1]:

Write a function (including docstring) called `polygonal` which takes as input two integers, the second of which represents the number of sides in a polygon. The function returns `True` if the first integer is a number that can be represented by a polygon with the given number of sides. For example, `polygon(10, 3)` returns `True`, while `polygon(10, 5)` returns `False`.

Presentation exercise 4

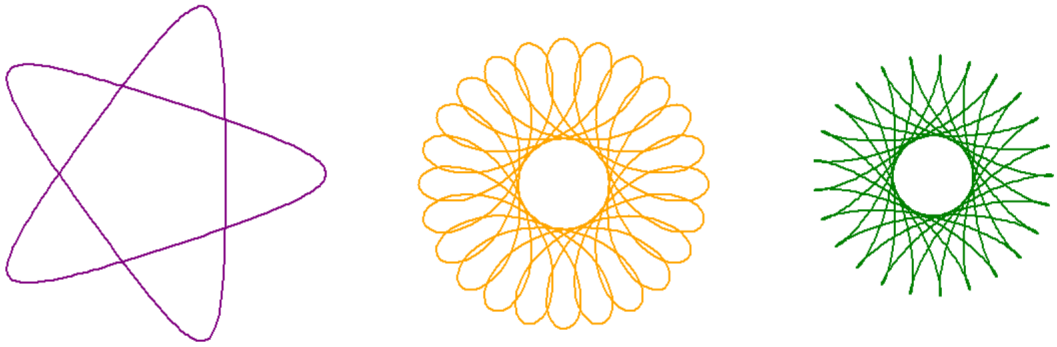
This question is for those of you with a stronger mathematical background.

An hypotrochoid is a curve traced by a point attached to a circle of radius r rolling around the inside of a fixed circle of radius R, where the point is a distance d from the center of the interior circle [2]. As kids you might have enjoyed tracing hypotrochoids yourselves! The parametric equations of an hypotrochoid are:

$$x = (R - r) \cdot \cos(\theta) + d \cdot \cos\left(\frac{R - r}{r} \cdot \theta\right)$$

$$y = (R - r) \cdot \sin(\theta) - d \cdot \sin\left(\frac{R - r}{r} \cdot \theta\right)$$

where $0 \leq \theta < 2\pi \cdot \frac{LCM(r,R)}{R}$, where LCM stands for least common multiple. Write a function called `draw_hypotrochoid` that takes as input a turtle, r, R, and d and draws the corresponding hypotrochoid. For example:



- The purple hypotrochoid has parameters: r = 75, R = 125, d = 125
- The green hypotrochoid has parameters: r = 85, R = 125, d = 80
- The orange hypotrochoid has parameters: r = 25, R = 120, d = 50

[1] https://en.wikipedia.org/wiki/Polygonal_number
[2] https://en.wikipedia.org/wiki/Hypotrochoid

Presentation exercise 5

For this question you must work in a pair and present together. Make sure to evenly contribute to both the code and the presentation. The presentation should last 8-10 minutes (approx. 5 minutes per person).

Dodgeball is a game in which two teams of players stand on the opposite end of a court and try to throw balls that hit opposing players in order to remove them from the game.

In this exercise we will write a simple turn-based, one-on-one dodgeball game in Python (one person on each team), with one player being human and the other player controlled by the computer. In a turn, each of the two players will choose one of three actions. Both chosen actions will then be revealed at the same time.

The three actions are: **throw** (throw the ball at the opponent), **dodge** (avoid a ball being thrown at you by your opponent), and **pick up** (pick a ball up off the ground - there are infinite balls on the ground). You can only throw a ball if you have previously picked one up, and you can only pick up a maximum of three balls at a time. If you throw a ball at the opponent and they have not chosen the dodge action, then you get one point. (If you throw and they dodge, nothing happens.) The game continues in this manner until 5 points have been scored. You will write three functions to accomplish this task.

- `point_gained(human_action, computer_action)`. This function takes two strings as arguments, corresponding to the action taken by each player (`'throw'`, `'dodge'`, or `'pickup'`). The return value is a string denoting which player (`'human'` or `'computer'`) has gained a point (or an empty string if no player has gained a point). If both players gain a point, then return a string with both player's names.
- `get_computer_action(num_player_balls, num_opponent_balls)`. This function will decide what action the computer will take. It takes two arguments: an integer representing the number of balls the player currently has in their possession, and the same for the opponent's number. The action will be decided as follows. If the player has picked up at least two balls, then the computer should dodge. Otherwise, if the computer has not picked up any ball yet, then either pickup or dodge should be randomly chosen. If the computer has already picked up three balls, then either throw or dodge should be randomly chosen. Finally, if the computer has picked up a ball or two but not three, then any of the three actions should be chosen from randomly. The function should return the string corresponding to the action (`'throw'`, `'dodge'`, or `'pickup'`).
- `get_human_action(num_player_balls, num_opponent_balls)`. This function will ask the user which action they want to take (either `'throw'`, `'dodge'`, or `'pickup'`), and then return that action string. It takes two arguments (an integer representing the number of balls the player currently has in their possession, and the same for the opponent's number). If the user does not enter one of the three valid actions, or tries to throw without having any balls in their possession, then the function should ask them again for input until they enter a valid one.
- `play_game()`. This function will call the other functions as necessary to play the game until 5 points have been scored by either the human or computer player. Make sure to print out descriptive messages that show what is going on in the game (what player has chosen which action, who has gained a point, etc.).

Presentation exercise 6

*"Pick the right door and you'll go free. Pick the wrong door and there *he'll* be."*

For this question you must work in a pair and present together. Make sure to evenly contribute to both the code and the presentation. The presentation should last 8-10 minutes (approx. 5 minutes per person).

The park was called Playland, and it was the best. You could laugh and scream, and get scared to death on rides, and stuff up on junk food, and ditch your parents, all in one night. And there was a haunted house, called Laughing In The Dark. It was only open one night a year, October 31.

Inside the haunted house it is very dark and maze-like. Write a function `haunted_house_maze` that lets a park guest try their luck at the maze. The guest will start at the north-west corner of the maze (position 0, 0 on the Cartesian plane) and must reach the south-east corner (position 4, 4). Keep track of their current position using two integer variables (x and y).

At each position, the user will have the option to move in one of the four cardinal directions (north, south, east or west). As it is very dark, the guest cannot tell if there is a wall in any direction or not. After the guest inputs a direction, determine if there is a wall by flipping a coin (50% chance that there is a wall). If there is a wall, inform the guest that they have crashed into a wall, and do not alter their position. (You do not need to keep track of where the interior walls are; as the house is haunted, the walls occasionally shift around.) If they try to go beyond the bounds of the maze (beyond the bounds of the square from (0, 0) to (4, 4)), likewise inform them that they have crashed, and do not alter their position.

Further, as it is very dark, the guest can occasionally get confused and move in a direction that they did not intend. After the guest enters a direction, and there was no wall in that direction, determine if they are confused by flipping two coins (25% chance to be confused). If they are confused, then randomly choose a different direction for them to move in and assume that there is no wall in that direction.

When the guest moves successfully (i.e., if there was no wall), then move them in that direction by updating their position variables (x and y). Continue this process of asking for movement as above until they reach position 4, 4 (the maze exit), at which point you should inform them they have found the maze exit. Then call the function `haunted_house_door_room`, which works as follows.

At the end of the haunted house is a heptagonal-shaped room containing 7 numbered doors, each painted in a different color. One door leads to the exit of the haunted house, five doors have walls behind them, and one door has an animatronic clown called **Zeebo** behind it. If the guest chooses the latter door, Zeebo will spring out and terrify the guest. Your function should let the guest choose which of the seven doors to try opening. The guest should always guess a door with a wall behind it for their first two guesses. On the third try, randomly choose two of the five remaining doors to be the exit and Zeebo's door, and let them choose from the remaining doors. If they choose correctly, congratulate them and end the program. If they choose a door with a wall behind it, let them try another door. If they choose Zeebo's door, crash the program by raising an exception, by writing the following line of code. We will explore the raise keyword later in the course.

```
raise Exception("Zeebo has found you! Muahahahahahaha!")
```

Note: This question is based on an episode of the 90s children's television show 'Are You Afraid of the Dark' entitled 'The Tale of Laughing In The Dark.'