Lists

Solar System Builder

Solar System Builder
Over the next few exercises, we will construct a program that allows a user to create a simple solar system using these variables:

```
planet_x = []  # list of planet centre x-coordinates
planet_y = []  # list of planet centre y-coordinates
planet_diameter = 50  # planet diameter in pixels
```

Exercise 1
Initialize `planet_x` and `planet_y` with three planets whose (x, y) coordinates are as follows:

$(250, 250)$

$(100, 400)$

$(400, 100)$

Solution

```
def setup():

    # create 500x500 canvas with black backdrop
    size(500,500)
    background(0,0,0)

    global planet_x
    global planet_y

    # initialize the sun & two other planets
    planet_x = [250,100,400]
    planet_y = [250,400,100]
```

This code can be found in `cmpt140-ch14-py/cmpt140_ch14_solar_system/cmpt140_ch14_solar_system.pyde`

Exercise 2
Write Processing code which draws all the planets in the lists.

the canvas should be black

all the planets have the same diameter, given by `planet_diameter`

the first planet should be yellow (it's the sun)

all other planets should be blue

Solution

```
def draw():
    global planet_x
    global planet_y
    global planet_diameter

    background(0,0,0)

    # draw the sun
    fill(255,255,0)
    ellipse(planet_x[0],planet_y[0],planet_diameter,planet_diameter)

    # all other planets
    for i in range(1,len(planet_x)):
        fill(0,0,255)
        ellipse(planet_x[i],planet_y[i],planet_diameter,planet_diameter)
```

This code can be found in `cmpt140-ch14-py/cmpt140_ch14_solar_system/cmpt140_ch14_solar_system.pyde`

Exercise 3
Write Processing code that:

> On a mouse click, appends a new planet's (x, y) coordinates to the lists. The new planet's coordinates are the same as the mouse coordinates

> On pressing the 'r' key, deletes the most recently added planet's coordinates from the lists, except that we cannot delete the very last planet (it's the sun!)

Solution

```
def mousePressed():
    global planet_x
    global planet_y

    # add mouse click coords as new planet to draw
    planet_x.append(mouseX)
    planet_y.append(mouseY)

def keyPressed():
    global planet_x
    global planet_y

    # remove last drawn planet (do not remove the sun)
    if key == 'r' and len(planet_x) > 1:
        planet_x.pop()
        planet_y.pop()
```

This code can be found in `cmpt140-ch14-py/cmpt140_ch14_solar_system/cmpt140_ch14_solar_system.pyde`

Exercise 4
Write Processing code which shifts the location of all planets by 20 pixels in the appropriate direction for these keypresses:

'w' shifts all the planets upwards

'a' shifts all the planets left

's' shifts all the planets downwards

'd' shifts all the planets right

Solution

```
def keyPressed():
    global planet_x
    global planet_y

        ⋮

    # shift solar system map up, left, down, or right
    elif key == 'w':
        for i in range(len(planet_y)):
            planet_y[i] = planet_y[i] - 20
    elif key == 'a':
        for i in range(len(planet_x)):
            planet_x[i] = planet_x[i] - 20
    elif key == 's':
        for i in range(len(planet_y)):
            planet_y[i] = planet_y[i] + 20
    elif key == 'd':
        for i in range(len(planet_x)):
            planet_x[i] = planet_x[i] + 20
```

This code can be found in `cmpt140-ch14-py/cmpt140_ch14_solar_system/cmpt140_ch14_solar_system.pyde`

Exercise 5
Write Processing code which displays the **number of planets** in the top-left corner of the canvas. Underneath this counter, display the **index**, **x-coordinate**, and **y-coordinate** for every planet in the list.

Solution

```
def draw():

        ⋮

    display_planet_info(planet_x,planet_y)

def display_planet_info(planet_x,planet_y):
    """
    shows the coordinates of all planets in top-left overlay
    planet_x: list of all planets' x-coordinates
    planet_y: list of all planets' y-coordinates
    """

    # show count of planets in top-left corner
    fill(255,255,255)
    text("planet count:",10,10)
    text(len(planet_x),100,10)

    # list all planets' indices and coordinates
    for i in range(len(planet_x)):
        text(i,10,40+i*15)
        text(planet_x[i],30,40+i*15)
        text(planet_y[i],60,40+i*15)
```

This code can be found in `cmpt140-ch14-py/cmpt140_ch14_solar_system/cmpt140_ch14_solar_system.pyde`

The `text()` coordinates used are semi-arbitary and likely differ from the ones used here. As long as the text is visible in the upper-left corner of the canvas and does not overlap, then the coordinates used should be fine.

More List Exercises

Exercise 6
What does the following Processing code do?

```
scores = [440,985,255,400,379,933,816,983,968,167,
          519,979,654,653,734,271,405,227,129,333,
          660,593,971,597,875]

for i in range(5,len(scores)+1,5):
    score_range = scores[0:i]
    avg_score = sum(score_range) / len(score_range)
    print(i,avg_score)
```

Answer: The running average for the list of scores is printed to the screen for every five new scores introduced into the calculation.

Exercise 7

The police are working on a case. The suspect's first name starts with a "J" and works with a "Michael Thorton".

Write Processing function `find_suspects` which takes a list parameter `employees` and returns another list containing names of possible suspects.

> `employees` is a list of strings; each string is in `"FirstName LastName"` format
>
> If there is no `"Michael Thorton"` in the list, then no one is a suspect.
>
> Otherwise, anyone whose name starts with "J" is a suspect
>
> Hint: We can index strings in the same way as lists!

Example Input:

```
employees = ["Michael Thorton", "Jason Bourne", "Jack Bauer"]
```

Example Output:

```
["Jack Bauer","Jason Bourne"]
```

Solution

```
def find_suspects( employees ):
    """
    returns list of possible suspects from company's list of employees
    employees: list of company's employees to check
    return: list of suspects with first name starting with "J"
            in lexicographical order
    """
    matching = []  # to contain employees with first names start with "J"

    # check if our lead works for this company
    if "Michael Thorton" in employees:
        # for every employee, check their first name
        for e in employees:
            if e[0] == "J":
                matching.append(e)
        # lexicographical ordering for matching employees
        matching.sort()
```

This code can be found in `cmpt140-ch14-py/cmpt140_ch14_agents/cmpt140_ch14_agents.pyde`