INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad
CS5107: Programming Lab
Lab 2: The plot thickens

27 Dec, 2020

Time: 1 week

Max points: 100

1. Some instructions for you codes:
   - Codes should be compatible with *Python3*.
   - Code for each question should be placed in separate file (*Q2.py, Q3.py, Q4.py and Q5.py*).
   - Codes should be properly commented.

2. Create a class `PieChart` such that [45]
   - It should be possible to create an object of this type from a dictionary mapping labels to positive numbers.

```
p = PieChart({'Frogs': 10, 'Dog': 25})
```

   - Exception should be thrown if any label is not a string or any value is not a positive numeric.

```
p = PieChart({1, 23})
```

Expected output:

```
<class 'Exception'>
Label should be string
```

```
p = PieChart({'Frog': '30'})
```

Expected output:

```
<class 'Exception'>
Value should be a postive numeric
```

```
p = PieChart({'Frog': -10})
```

Expected output:

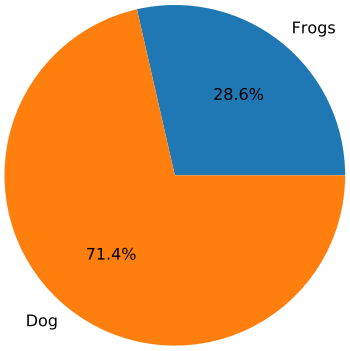
```
<class 'Exception'>
Value should be a postive numeric
```

Question 2 continues on the next page...

- It should have method **show** to display/plot the actual pie-chart corresponding to this object.

```
p = PieChart({'Frogs': 10, 'Dog': 25})
p.show()
```

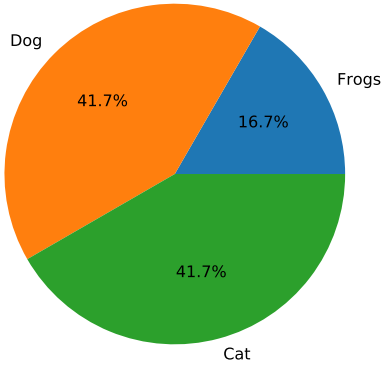
Expected output:



- Overload the + operator so that one can add a tuple consisting of a label and value to the *PieChart* object.

```
p = PieChart({'Frogs': 10, 'Dog': 25})
p = p + ('Cat', 25)
p.show()
```

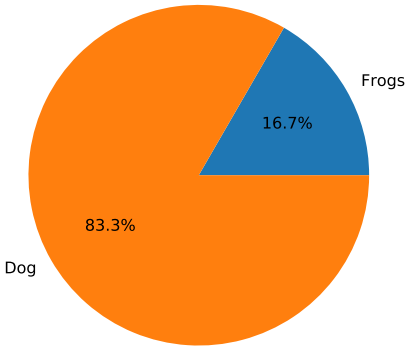
Expected output:



Question 2 continues on the next page...

```
p = PieChart({'Frogs': 10, 'Dog': 25})
p = p + ('Dog', 25)
p.show()
```

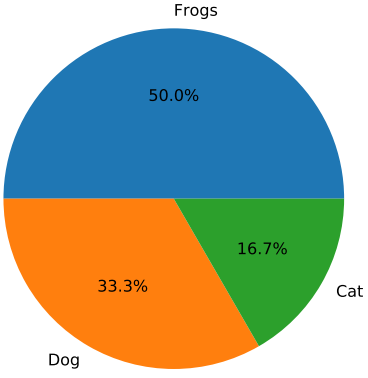
Expected output:



- An appropriate exception should be thrown if the tuple does not have 2 elements, or first tuple element is not a string, or second tuple element is not a positive numeric.
- Overload the + operator so that one can add together two *PieChart* objects.

```
p = PieChart({'Frogs': 10, 'Dog': 20})
p = p + PieChart({'Frogs': 20, 'Cat': 10})
p.show()
```

Expected output:

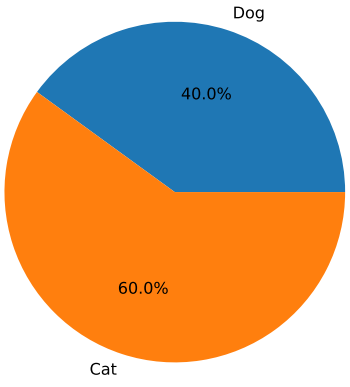


Continued on next page...

- Overload the − operator so that one can delete a label (and its corresponding value) from the *PieChart* object.

```
p = PieChart({'Frogs': 10, 'Dog': 20, 'Cat':30})
p = p - 'Frogs'
p = p - 'Lions'
p.show()
```

Expected output:



3. Let $X_1, X_2, \ldots$ be a sequence of i.i.d. Bernoulli random variables with mean μ and variance σ^2. Define $\tilde{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i$. Then $\frac{\tilde{X}_n - \mu}{\sigma/\sqrt{n}}$ looks like *standard normal distribution* as $n \to \infty$. This a famous result and is known as the *Central Limit Theorem (CLT)* For details refer the Wikipedia page on CLT. Using the FuncAnimation function of *matplotlib*, create an animation to demonstrate CLT. A reference animation is available at animation link. Try to match the reference animation as much as possible. [20]

4. Create a class `Sines` such that [20]

   - It should have method `addSine` that allows one to add any sine function with unit amplitude to this an object of type `Sine`. `addSines` should take the phase offset (in degree) as its argument.

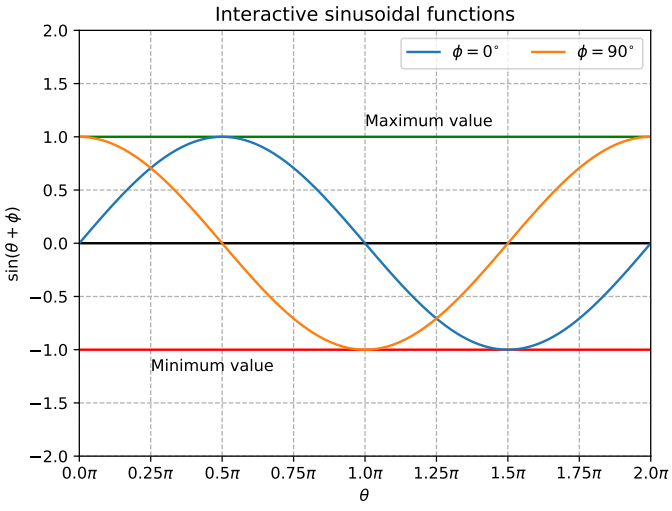
```
s = Sines()
s.addSine(0)
s.addSine(90)
```

   - It should have method `show` to display/plot all the sine functions added to an object of this type. Your plots should match the expected format as much as possible.

```
s = Sines()
s.addSine(0)
s.addSine(90)
s.show()
```

Question 4 continues on the next page...

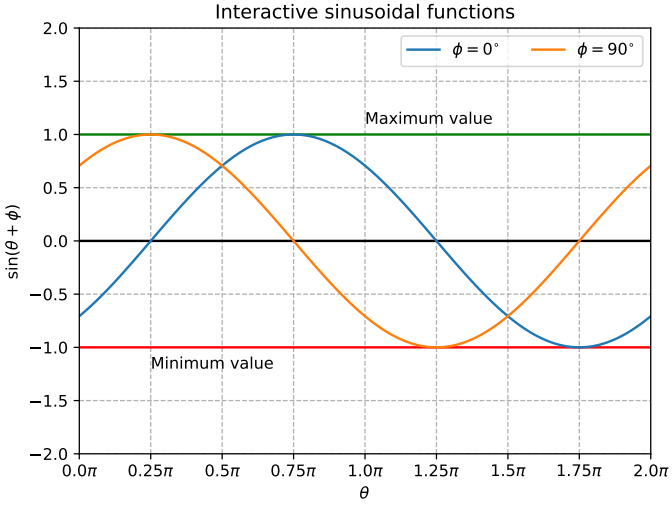
Expected output:



- It should have method `shiftRight` to shift all sine functions to the right. This method should take the shift amount (in degree) as its argument.

```
s = Sines()
s.addSine(0)
s.addSine(90)
s.shiftRight(45)
s.show()
```

Expected output:

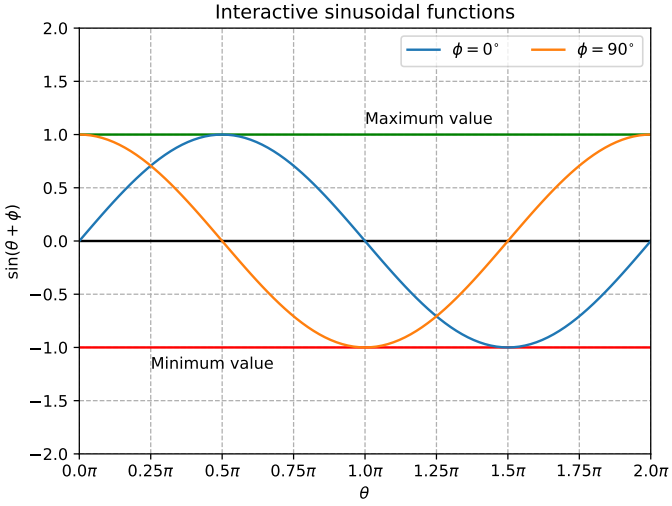


Continued on next page...

- It should have method `shiftLeft` to shift all sine functions to the left. This method should take the shift amount (in degree) as its argument.

```
s = Sines()
s.addSine(0)
s.addSine(90)
s.shiftRight(45)
s.shiftLeft(45)
s.show()
```

Expected output:



5. To the above class, add a method `interact` that will allow one to interact with the plot using the following keys. [15]

   - A press of *d* key will move the x-axis (and the curves) to the right a constant speed.
   - A press of *a* key will move the x-axis (and the curves) to the left a constant speed.
   - A press of *spacebar* will pause/unpause the movement.
   - A press of *r* key will reset the curves to their initial position.

```
s = Sines()
s.addSine(0)
s.addSine(30)
s.addSine(45)
s.shiftRight(45)
s.interact()
```

Expected output: Refer the sample interaction video.