previous up next

Tartalomjegyzék

- 1 Exercises
  - 1.1 Shapes on a canvas
    - 1.1.1 Shapes
    - 1.1.2 Inheritance and constructors
    - 1.1.3 Canvas
  - 1.2 Iterables

Exercises

Work in Spyder!

Shapes on a canvas

Shapes

Write a class called **Shape**.

- Let it have two members: **x** and **y**, the coordinates of the shape on the plane (center of mass).
- Define a **move** method, with one parameter **v**: a list of length 2, a vector to translate the shape with. After this method the coordinates should be changed.

Define the following classes as children of **Shape**:

- **Ellipse** with additional parameters (except the *(x, y)* coordinates): **a** and **b** the *x* and *y* axes radii
- **Rectangle** with additional parameters (except the *(x, y)* coordinates) **a** and **b** the length of the sides

Write an **area** method for both, which calculates the area!

Define an **equation** method for printing the equation of the **Ellipse**! Something like:

```
((x-1)/2)^2 + ((y-2)/3)^2 = 1
```

Inheritance and constructors

If the child class (e.g. Ellipse) you want to call the parents class' constructor then you have two ways:

```
class B(A):
    def __init__(self, x, y, a, b):
         A.__init__(self, x, y)
         # OR
         super(B, self).__init__(x, y)
```

The first explicitly calls the parents `__init__`

```
A.__init__(self, x, y)
```

The second one calls the parent of **B** which happens to be **A**:

```
super(B, self).__init__(x, y)
```

Canvas

Define a class called **Canvas** (vászon)

- Its only member variable should be a list of Shapes: **shapes**. This stores several Shape objects.
- Define an **add** method which adds a new shape to the canvas!
- Make this class **iterable**! Write the **__iter__(self)** and the **next(self)** methods, as seen on the lecture.
- Define a **crop** method in the following way.
    - Its two parameters should be two coordinates: a top-left corner and a bottom-right corner.
    - The function should return the list of Shapes which entirely fit in that rectangle.
    - For this it is best to implement a **box()** method on both **Ellipse** and **Rectangle** which returns a bounding box of the shape:

a list of two coordinates, the top-left corner and a bottom-right corner of the object.

Iterables

Write an iterable class like **range**, but without returning a whole list, but storing only the actual element.

```
class Range(object):
    def __init__( ... ):
        ...
    def __iter__( ... ):
        ...
    def next( ... ):
        ...
```

- Its constructor should have one parameter: a number or a string. The iteration should go up that number, from 0 with 1 steps.
- If the number is not positive, then the iteration should take 0 steps.
- If you get a string then try to convert it to a number. It it cannot be converted, then raise a **ValueError** exception.
    - If it is a valid integer, then calculate with that.
- If you get the string **"http://wiki.math.bme.huinf"http://wiki.math.bme.hu** then make the iteration go endless (infinite loop)!

previous up next