1 Exercises

Exercise 1. If `s` is a string, `s.upper()` returns a copy of `s` converted to uppercase.

a. What does the following code fragment write?

```
s = 'Hello World'
s.upper()
stdio.writeln(s[6:11])
```

b. What does the following code fragment write?

```
s = 'Hello World'
s = s.upper()
stdio.writeln(s[6:11])
```

Exercise 2. Suppose we have a user-defined data type called `Circle` that represents a circle of radius r centered at (h, k) and supports the following API:

`Circle`

`Circle(h, k, r)`	constructs a circle c of radius r centered at (h, k); when no arguments are given, c is a unit circle centered at the origin
`c.area()`	returns the area of c
`c.contains(x, y)`	returns `True` if c contains† (x, y) and `False` otherwise
`c < d`	returns `True` if c is area-wise smaller than d, and `False` otherwise
`c == d`	returns `True` if c and d represent the same circle, and `False` otherwise
`str(c)`	returns a string representation of c, as `(h, k, r)`

† A point (x, y) is contained in a circle of radius r centered at (h, k) if $(x - h)^2 + (y - k)^2 \leq r^2$

a. Is the `Circle` data type immutable?

b. How do you create a `Circle` object `c1` representing a circle centered at $(1, 1)$ and having radius 2?

c. How do you create a `Circle` object `c2` representing a unit circle centered at the origin?

d. How do you obtain the area of `c1`?

e. How do you check if the point $(1.2, 2.2)$ is contained in `c1`?

f. How do you compare the areas of two circles represented by `Circle` objects `c` and `d` without invoking the `area()` method explicitly? What does the code translate to internally?

g. How do you check if two `Circle` objects `c` and `d` represent the same circle? What does the code translate to internally?

h. How do you obtain the string representation of `c1`? What does the code translate to internally?

i. Provide code that creates a list `a` of 100 `Circle` objects, each representing a circle centered at the origin and having a random radius from the interval $[0, 1)$.

j. Provide an expression that uses `map` and `reduce` to calculate the sum of the areas of the circles stored in the list `a` from the previous part.

Exercise 3. Write a program called `filter.py` that accepts three floats h, k, and r as command-line arguments, creates a `Circle` object `c` representing a circle centered at (h, k) and having radius r, reads in pairs (x, y) of floats from standard input representing points on a 2D plane, and writes the fraction of points that fall inside the circle `c`. For example

```
>_ ~/workspace/programs
$ python3 filter.py 0 0 3
1 2
3 4
1 5
```

```
1 3
<ctrl-d>
0.25
```

2 Solutions to Exercises

Solution 1.

a. `World`

b. `WORLD`

Solution 2.

a. Yes

b. `c1 = Circle(1, 1, 2)`

c. `c2 = Circle()` or `c2 = Circle(0, 0, 1)`

d. `c1.area()`

e. `c1.contains(1.2, 2.2)`

f. `c < d` which translates to `c.__lt__(d)` internally

g. `c == d` which translates to `c.__eq__(d)` internally

h. `str(c1)` which translates to `c1.__str__()` internally

i.

```
circles = []
for i in range(100):
    c = Circle(r = stdrandom.uniform(0, 1))
    circles.append(c)
```

j. `reduce(lambda x, y: x + y, map(lambda x: x.area(), circles))`

Solution 3.

filter.py

```
import stdio
import sys
from circle import Circle

def main():
    h = float(sys.argv[1])
    k = float(sys.argv[2])
    r = float(sys.argv[3])
    c = Circle(h, k, r)
    total, inside = 0, 0
    while not stdio.isEmpty():
        x = stdio.readFloat()
        y = stdio.readFloat()
        total += 1
        inside += 1 if c.contains(x, y) else 0
    stdio.writeln(1.0 * inside / total)

if __name__ == '__main__':
    main()
```