

Project 3

PYTHON OBJECT MODEL
OBJECT ORIENTED DESIGN

Graphics

- Download the file graphics.py from the link:

http://brd4.braude.ac.il/~samyz/cgi-bin/view_file.py?file=PYTHON/projects/graphics.py

- This file implements our graphical environment. Specifically, it defines a canvas window on which we can draw points, lines, rectangles, and other geometrical shapes
- There is no need to read or understand the code in this module
- It is based on the Tkinter module, which is the simplest graphics environment that comes with any Python distribution and is therefore always available
- If you want to experiment with graphics programming, you may start with:

<http://www.tkdocs.com/tutorial/>

The **Point** Abstract Data Type

- `p = Point(x,y)` [constructor]
 - ◆ Create a new point `p` from two integers: `x`, `y`
 - ◆ Our domain is the two-dimensional plane for abstract circuit design (CAD system)
- `p.x = x` coordinate [field]
- `p.y = y` coordinate [field]
- `p.move(dx, dy)` [mutator]
 - ◆ Move the point `p` to new coordinates: `x+dx`, `y+dy`
- `p.draw()` [accessor]
 - ◆ Draw the point on the screen
- `p.text(t)` [accessor]
 - ◆ Draw a texts string `t` above the point

P(80,130)
■

P(170,180)
■

P(230,250)
■

Test Driven Development

- Reminder: in test driven methodology you write your tests before the implementation of your ADT !!!
- After implementation, your tests should run and **PASS** after each modification you make to your implementation (“nightly test regression”)
- The following tests are your “insurance policy” that your implementation is correct. The more tests you write, the more you’re insured

```
# Testing our Point ADT: test 1
def test1():
    print "==== Testing The Point Class ====="
    p1 = Point(20,20)
    p2 = Point(50,60)
    print "Testing the Python print statement on Point p1:"
    print p1
    print "Testing the Python print statement on Point p2:"
    print p2
    print "Test 1: PASSED"
```

Test Driven Development

- Here is a more formal and practical test

```
def test2():  
    p1 = Point(20,20)  
    p2 = Point(50,60)  
    assert p1.x == 20 and p1.y == 20  
    assert p2.x == 50 and p2.y == 60  
    p1.move(100, 200)  
    p2.move(100, 200)  
    assert p1.x == 120 and p1.y == 220  
    assert p2.x == 150 and p2.y == 260  
    print "Test 2: PASSED"
```

Point Implementation

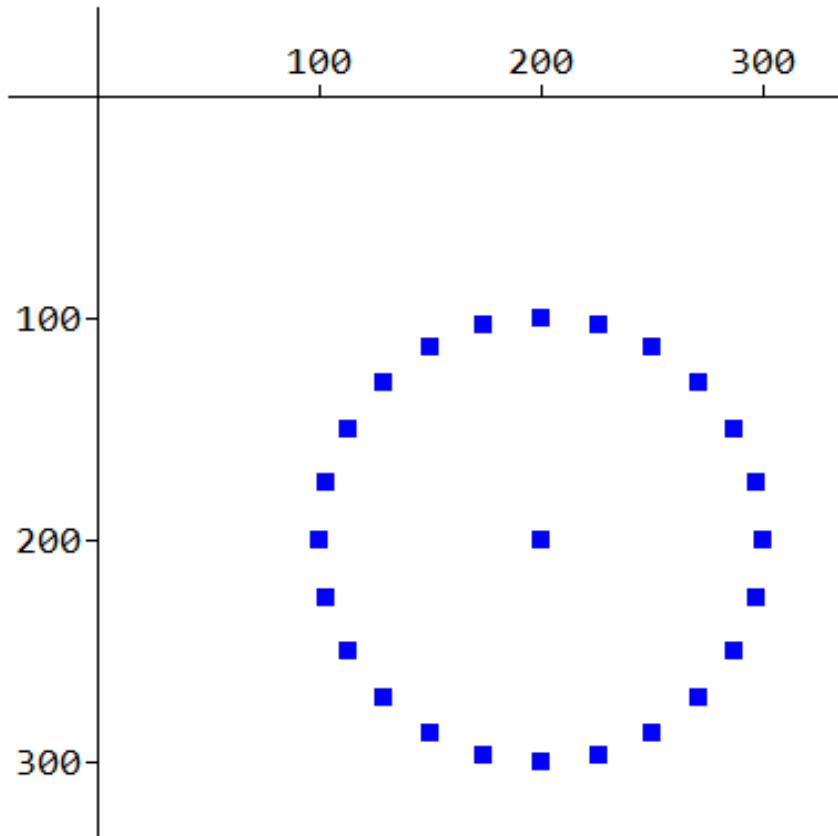
- Download the file point.py from the link:

http://brd4.braude.ac.il/~samyz/cgi-bin/view_file.py?file=PYTHON/projects/point.py

- We shall spend a few minutes in lab for reading and discussing the code before you start your work

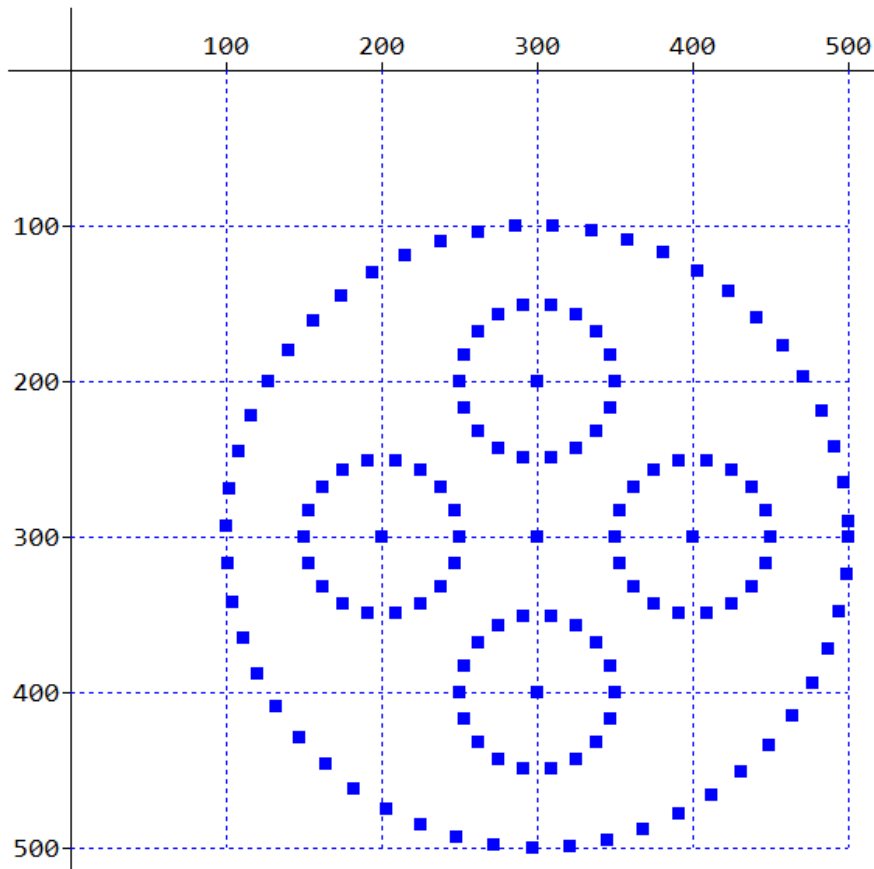
Problem 1

- Write a function `make_ring()` which draw 24 points on a circle with center=(200,200) and radius=100:



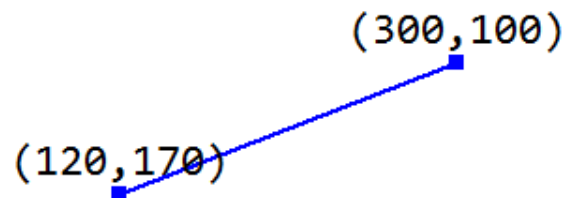
Problem 2

- Design and write a class Ring which can be used to draw rings as follow
- Hint:
 - ◆ the big circle center is: (300,300), radius=200, and it has 48 points
 - ◆ The small circle radius=50, has 18 points, and the centers are easy to compute



The **Line** Abstract Data Type (1)

- `l = Line(p1,p2)` [constructor]
 - ◆ Create a new line object `l` from two point objects: `p1`, `p2`
- `l.p1` get the first point [field]
- `l.p2` get the second point [field]
- `l.move(dx, dy)` [mutator]
 - ◆ Move the line `l` by `dx` units horizontally and `dy` units vertically
- `l.length()` [accessor]
 - ◆ Return line length – the distance between the points `p1` and `p2`
- `l.middle()` [accessor]
 - ◆ Return the middle point of this line (as a Point object!)
- `l.draw()` [accessor]
 - ◆ Draw the line on a canvas



Line Implementation

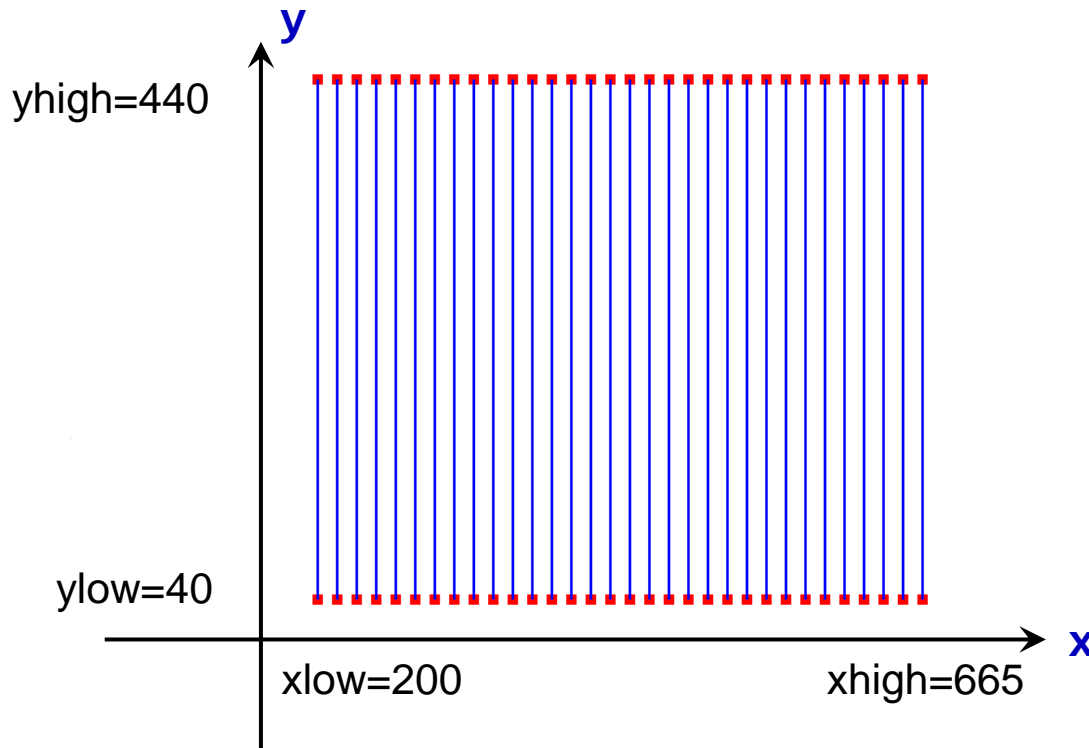
- Download the file line.py from the link:

http://brd4.braude.ac.il/~samyz/cgi-bin/view_file.py?file=PYTHON/projects/line.py

- This file contains an implementation of the Line class

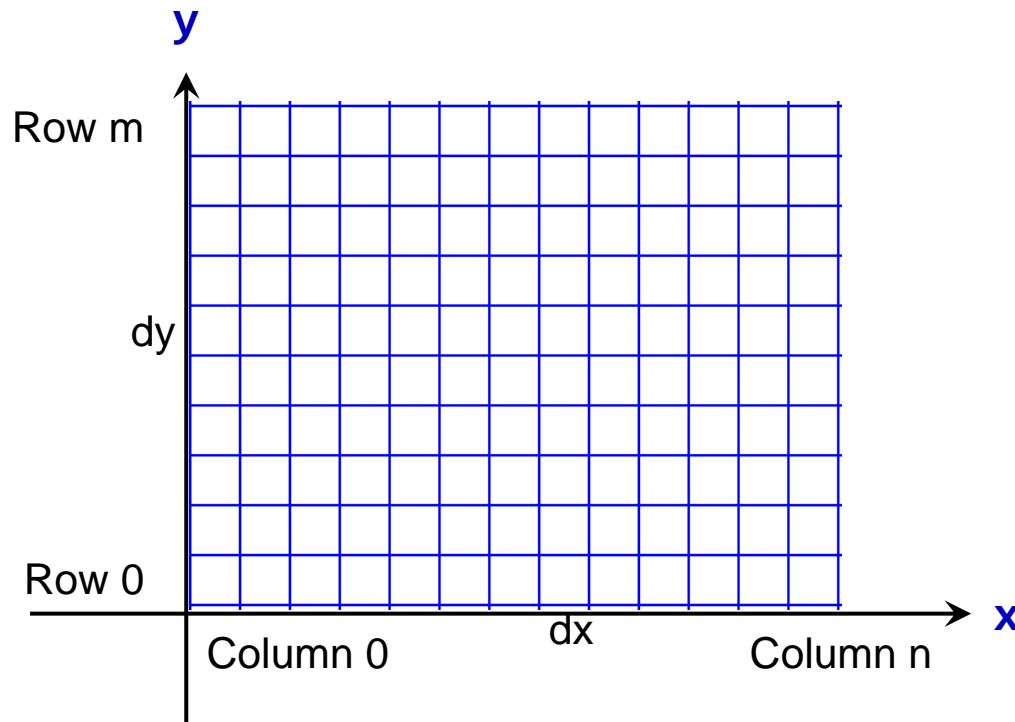
Problem 3: VLSI BUS

- A simple **VLSI BUS** consists of a well structured group of lines (sometimes called “signals” or “bits”)
- Write a function `draw_bus()` for drawing a 32 bits **BUS** with the following characteristics: `xlow = 200`, `xhigh=665`, `dx=15`, `ylow=40`, `yhigh=440`
- Make sure to draw the points too!

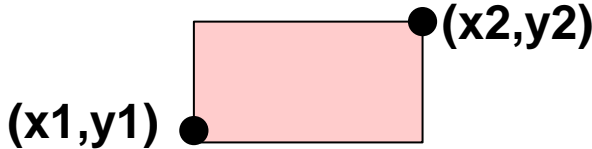


Problem 4: VLSI GRID

- A simple **VLSI GRID** consists of an equally spaced horizontal and vertical lines as in the bottom figure
- Write a short function **draw_grid(m,n,dx,dy)** for drawing an **m×n** grid such that the distance between vertical lines is dx , and distance between horizontal lines is dy



The **Rectangle** Abstract Data Type

- `r = Rectangle(x1,y1,x2,y2)` [constructor]
 - ◆ Create a new rectangle `r` from four integers: `x1`, `y1`, `x2`, `y2`
 - ◆ Our domain is the two-dimensional plane for abstract circuit design (CAD system)
- `r.draw()`

A diagram showing a light pink rectangle. The bottom-left corner is marked with a black dot and labeled `(x1,y1)`. The top-right corner is marked with a black dot and labeled `(x2,y2)`.
- `r.x1` = `x1` coordinate field [field]
- `r.y1` = `y1` coordinate field [field]
- `r.x2` = `x2` coordinate field [field]
- `r.y2` = `y2` coordinate field [field]
- `r.move(dx, dy)` [mutator]
 - ◆ Move the rectangle `r` to new coordinates: `x1+dx`, `y1+dy`, `x2+dx`, `y2+dy`
- `r.area()` [accessor]
- `r.width()` [accessor]
- `r.height()` [accessor]
- `r.center()` [accessor]

Test Driven Development

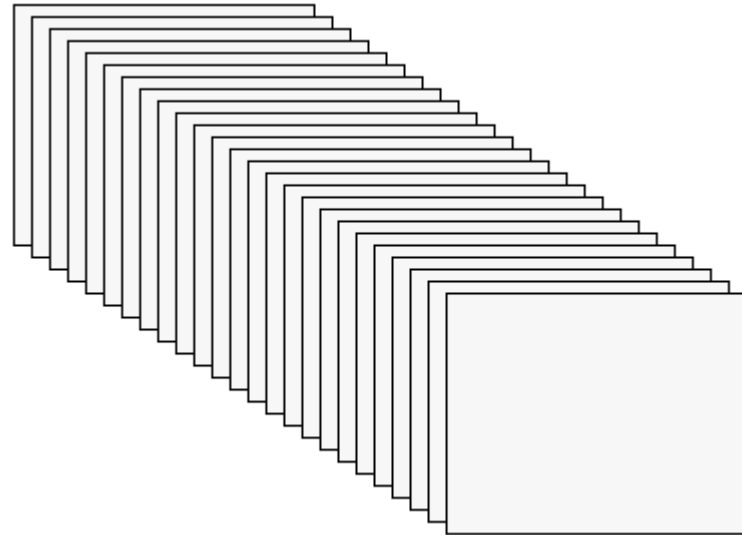
- Download the rectangle module from:
http://brd4.braude.ac.il/~samyz/cgi-bin/view_file.py?file=PYTHON/projects/rectangle.py
- Here is a simple code for testing the Rectangle class
Make sure it runs and is PASSED

```
# Testing our Rectangle ADT: test 1

def test1():
    r = Rectangle(30, 20 ,80, 70)
    assert r.area() == 2500
    assert r.width() == 50
    assert r.height() == 50
    r.move(15,25)
    assert r.x1 == 45
    assert r.y2 == 95
    assert r.area() == 2500
    print "Test PASSED"
```

Problem 5: Graphical Application

- Look at the simple Python implementation of the Rectangle ADT at project #3 section in the Python course web site:
<http://brd4.braude.ac.il/~samyz/PYTHON/#project3>
- Note that this implementation also contains a draw() method !
- **Problem 10:**
Use this implementation to write a short script that produces the following effect:



Hint:

```
r = Rectangle(10, 10 ,160, 130)
```

```
dx = 9, dy = 6, there are 25 rectangles
```

The solution is test3() in the above file ... but try first before you look it up!

Problem 6: Textfile Class

- Download the file `textfile.py` from the link:

http://brd4.braude.ac.il/~samyz/cgi-bin/view_file.py?file=PYTHON/projects/textfile.py

- This file implements the **Textfile** class for analyzing words frequency in large text files
- Read the usage description at the beginning of the file
- Use the Textfile class to find the 10 most used words in the book:
<http://brd4.braude.ac.il/~samyz/PYTHON/projects/jude.txt>
- Also indicate how many times each of these words appear in the book?
- Make sure to write a function that can be reused for other books ...

Problem 7: most common words

- Write a function `most_common_words(file, n)` which accepts a text file name and an integer `n` and prints the `n` most frequent words in the file and their frequency count:

```
file = "D:/BRAUDE/PYTHON/Projects/proj1/proj1.txt"
```

```
most_frequent_words(file, 10)
```

```
1.  the      88
2.  of       57
3.  a        52
4.  in       47
5.  is       34
6.  and      32
7.  are      27
8.  to       25
9.  numbers  16
10. cards    15
```

```
Hint: start with  tf = Textfile(file)
```