Calling Functions & Creating Objects

CPE101
Winter 2019 @ CPSLO
By Toshi

Point Class

```
class Point:

    #self denotes this object itself
    def __init__(self, x, y):
        self.x = x
        self.y = y

    #official string representation
    def __repr__(self):
        return “Point(%s, %s)” % (self.x, self.y)

    #define equality
    def __eq__(self, other):
        return type(other) == Point and\
            self.x == other.x and \
             self.y == other.y
```

- Continuing a line to the next line
  - Add \ at the end of the line to continue to the next line.
  - Except within parentheses and brackets.
    - ( ) [ ] { }

Some useful Python syntax

- Continuing a line to the next line
  - Add \ at the end of the line to continue to the next line.
  - Except within parentheses and brackets.
    - ( ) [ ] { }

Line Class

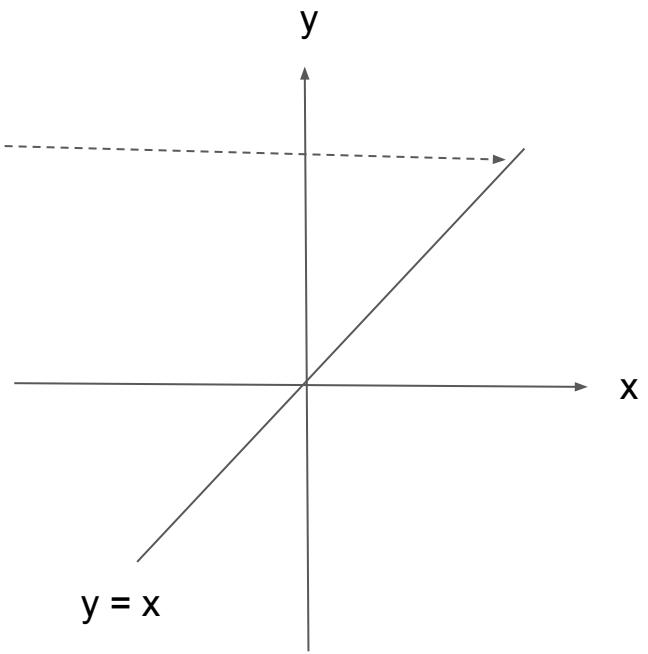
```
class Line:
    # y = a*x + b
    def __init__(self, a, b):
        self.a = a
        self.b = b

    #official string representation
    def __repr__(self):
        return “Line(%s * x + %s)” % (self.a, self.b)

    #define equality
    def __eq__(self, other):
        return type(other) == Point and\
            self.a == other.a and \
             self.b == other.b
```

poiont_on_line function

```
from point import Point
from line import Line

def point_on_line(line, point):
    '''checks if the point is on the line.
    Args:
        line(Line): A line
        point(Point): A point
    Returns:
        bool: True or False
    '''
    return point.y == line.a * point.x + line.b
```

Main function

```
#this function actually make a function call
def main():
    line = Line(1, 0)
    x = input("Enter x coordinate of a point: ")
    y = input("Enter y coordinate of a point: ")

    #input() returns str, so need to be converted to int
    x = int(x)
    y = int(y)

    #call the function with 2 arguments
    result = point_on_line(line, Point(x, y))
    print(result)
    #check if the calculation is correct
    assert result == (y == line.a * x + line.b)
```



Making main() called when the __name__ is ‘__main__’

```
#if the program is the top code (not an import)
if __name__ == '__main__':
    #call main function
    main()
```

lines_cross function

```
def lines_cross(line1, line2):
    '''checks if two lines cross
    Args:
        line1 (Line): a 2D line
        line2 (Line): other 2D line
    Returns:
        bool: True or False
    '''
    return (line1.a - line2.a) != 0
```

lines_cross_at function

```
def lines_cross_at(line1, line2):
    '''Computes a point where two lines cross
    Args:
        line1(Line): a line
        line2(Line): other line
    Returns:
        Point or None: the point where the two lines cross, or None if no crossing.
    '''
    if (line1.a - line2.a) != 0:
        x = line2.b - line1.b / (line1.a - line2.a)
        y = line1.a * x + line1.b
        return Point(x, y)
    return None
```