CPE/CSC 101

## Today

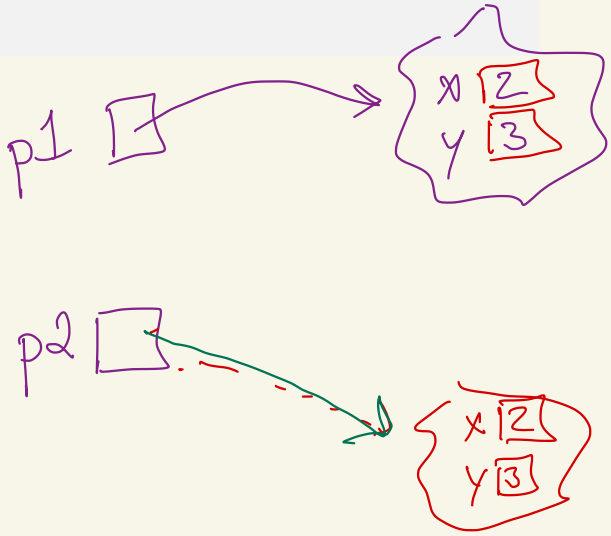
- Object Equality
- Boolean Operators

- Assignment #1 revisions by 10pm 1/20/2021

- Quiz #2

```
class Point:
    def __init__(self, x: float, y: float):
        self.x = x
        self.y = y

p1 = Point(2, 3)
p2 = Point(2, 3)
```




Python

1 == 7 False

7 == 7 True

p1 == p2 False

↑ should this be True?

equal?

math

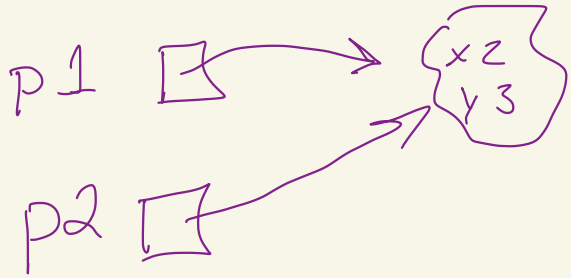
1 = 7

7 = 7

p1 = p2 ?

Python

```
p1 = Point(2, 3)
p2 = p1
```




```
p1 == p2    True
```

↖ default implementation that compares for identical references

```
p1 is p2        (for objects)
```

↖ identity

# Boolean Operators

bool - True, False

**Math**

$2 < x < 7$

**Python**

`2 < x and x < 7`

`2 < x` is T/F — op1

`x < 7` is T/F — op2

`and` is true only when both operands are true

## Truth Table

| op1 | op2 | op1 and op2 |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

$x = 4$

cond = x < 7 and x > 2

```
# Determine if two circles both
    overlap a third circle
# input: circle to check for overlap
#        with two others as Circle
# input: another Circle
# input: a third Circle
# result: bool
def overlap_3(circle1: Circle,
              circle2: Circle,
              circle3: Circle) -> bool:
    return (circles_overlap(circle1,
                            circle2)
            and circles_overlap(circle1,
                                circle3))
```

## Precedence

higher

*, /

+, -

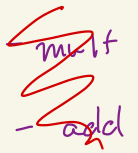
<, >=, ==, != (not equal)

not

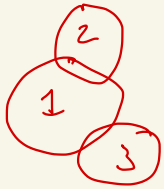
and

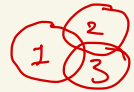
or

lower



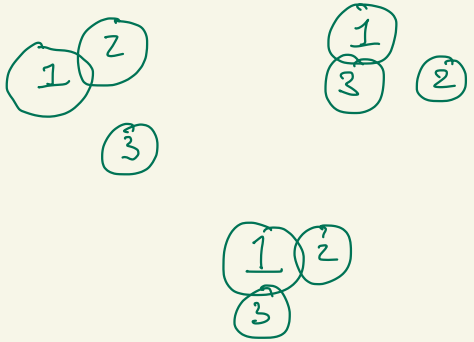
return circles_overlap(circle1, circle2 and circle3)



# or Truth

| $op_1$ | $op_2$ | $op_1$ or $op_2$ |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |




```
def overlap_some(circle1: Circle,
                 circle2: Circle,
                 circle3: Circle)
  -> bool:
    return (circles_overlap(circle1,
                            circle2)
            or
            circles_overlap(circle1,
                            circle3))
```

## not

| op | not op |
|---|---|
| T | F |
| F | T |


one of

```
def overlaps_only_one(circle1: Circle,
                      circle2: Circle,
                      circle3: Circle)
    -> bool:
    overlaps2 = circles_overlap(circle1,
                                circle2)
    overlaps3 = circles_overlap(circle1,
                                circle3)

    return (overlap2 or overlaps3)
            and
            not (overlaps2 and
                 overlaps3)

    return ((overlaps2 and not overlaps3)
            or
            (overlaps3 and not overlaps2))
```

( (overlaps2 and not overlaps3)
or
(overlaps3 and not overlaps2)

| overlaps2 | overlaps3 | not overlaps2 | not overlaps3 | overlaps2 and not overlaps3 |
|---|---|---|---|---|
| T | T | F | F | F |
| T | F | F | T | T |
| F | T | T | F | F |
| F | F | T | T | F |

## or

| | | |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

## XOr (exclusive or)

^ ← Python

| op1 | op2 | op1 ^ op2 |
|---|---|---|
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |

```
class Point:
    def __init__(self, x:float, y:float):
        self.x = x
        self.y = y

    def __eq__(self, other:object)->bool:
        return ( type(other) is Point
                 and self.x == other.x
                 and self.y == other.y
               )
```

p1 == p2

same type

near equality comparison in lab

p1 == 4