# CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2018

Swarthmore College

# Outline Nov 26:

- Continue: writing classes
- Snowflake class
- ScreenSaver class
- Hand back Quiz 4

Notes

- Lab 9 due tonight!!
- Extra office hours today 2:30-4:30pm (room 249)
- Lab 10 due Saturday
- Quiz 5 this Friday, study guide posted soon

# Recap Classes

# Recap Die class

- Defining the Constructor: builds an instance of the class (self), and initializes all instance variables (self.xxx)

```
class Die:

    def __init__(self, num_sides):
        """Construct a new die with the given number of sides."""
        self.sides = num_sides
        self.value = 1 # default starting value
```

# Recap Die class

- Defining the Constructor: builds an instance of the class (self), and initializes all instance variables (self.xxx)

```
class Die:

    def __init__(self, num_sides):
        """Construct a new die with the given number of sides."""
        self.sides = num_sides
        self.value = 1 # default starting value
```

- Using the Constructor: assign the new object to a variable, making the “self” placeholder a concrete instance

```
def main():
    # create 8-sided dice
    die1 = Die(8)
    die2 = Die(8)
```

# Recap Die class

- Defining Methods: always use “self” as the first argument (placeholder for the instance). Getters are a type of method that return instance variables or their derivatives.

```
def getValue(self):
    """Getter for the die's current value."""
    return self.value

def roll(self):
    """Choose a new random value for the die, i.e. roll it."""
    self.value = random.randrange(1,self.sides+1)
```

# Recap Die class

- Defining Methods: always use “self” as the first argument (placeholder for the instance). Getters are a type of method that return instance variables or their derivatives.

```
def getValue(self):
    """Getter for the die's current value."""
    return self.value

def roll(self):
    """Choose a new random value for the die, i.e. roll it."""
    self.value = random.randrange(1,self.sides+1)
```

- Using Methods: instance.method(…), don’t use self

```
# roll both until we get the same value
same = False
while not same:
    die1.roll()
    die2.roll()
    print(die1)
    print(die2)
    print()
    # check if the values are the same
    same = (die1.getValue() == die2.getValue())
```

# Recap Die class

- Defining the __str__ method: no print(..) statements! Build and return a single string. (no arguments besides self)

```
def __str__(self):
    """String representation of the die (with current value)."""
    return "%d-sided die, current value: %d" % (self.sides, self.value)
```

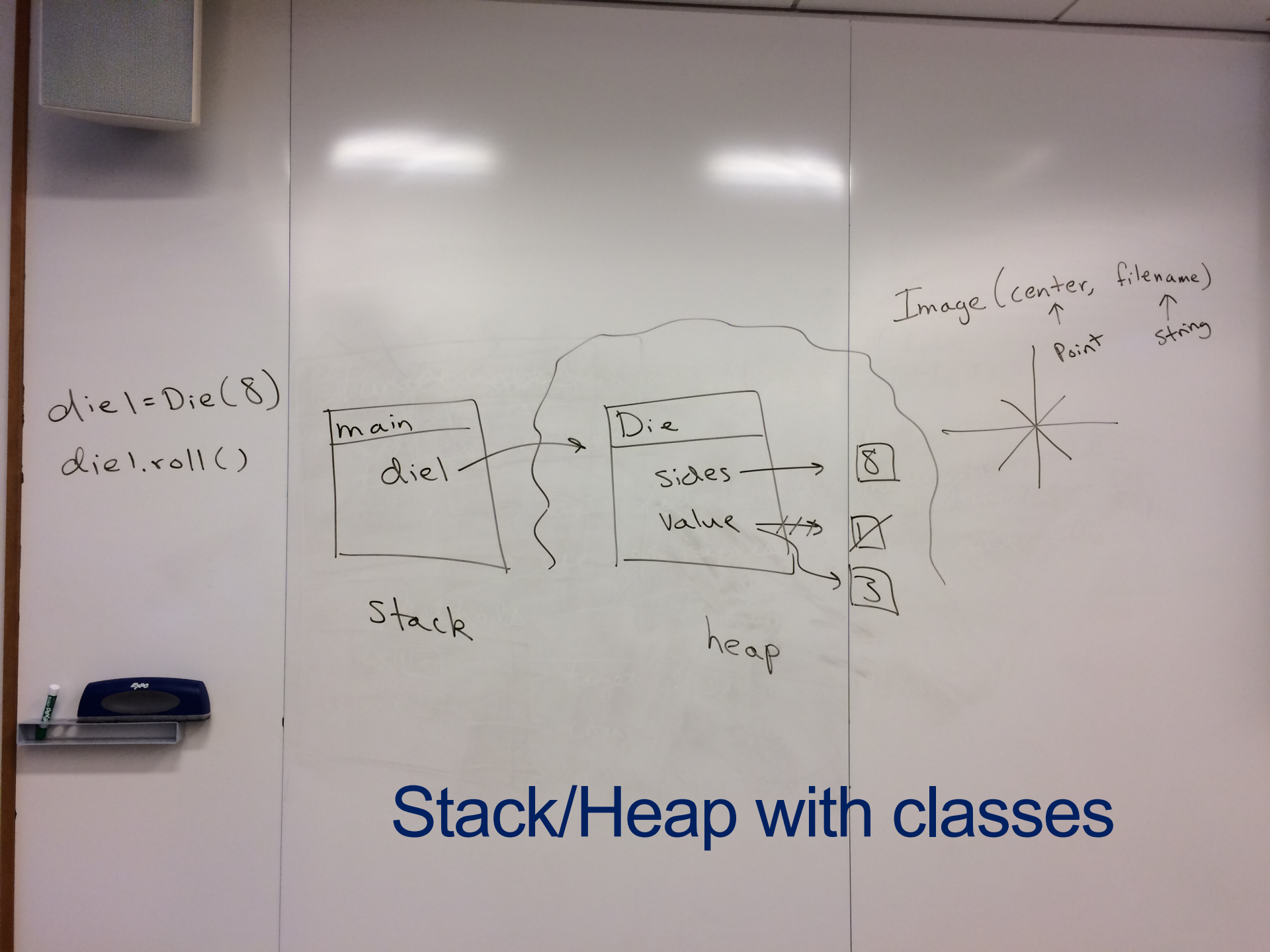
# Recap Die class

- Defining the __str__ method: no print(..) statements! Build and return a single string. (no arguments besides self)

```
def __str__(self):
    """String representation of the die (with current value)."""
    return "%d-sided die, current value: %d" % (self.sides, self.value)
```

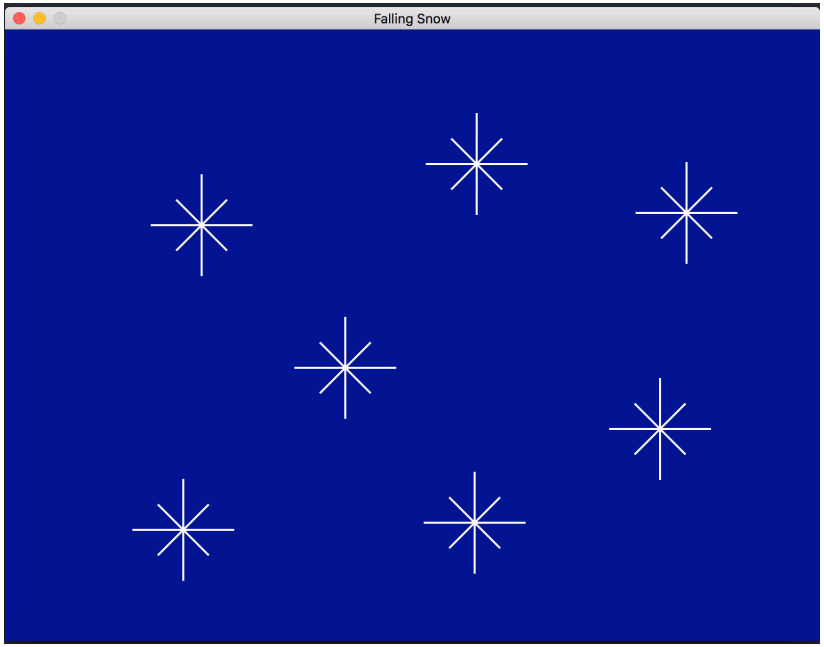
- Using the __str__ method: simply call print(instance)!

```
print(die1)
print(die2)
```




# Stack/Heap with classes

# Today




- Finish: snowflake.py

- Start: screen_saver.py