Presentation 18

Object Oriented Design

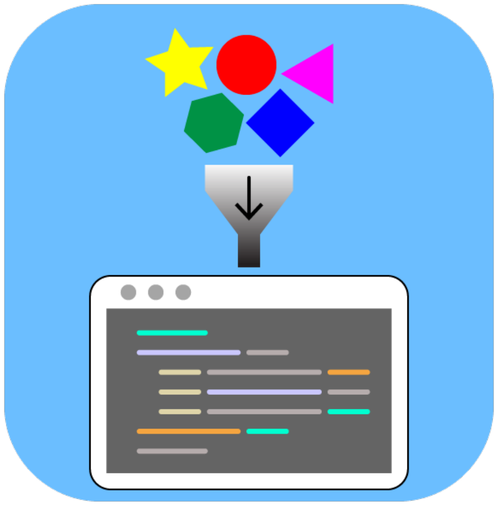
Announcements for Today

Assignments

- Survey for A4 still open
- A5 is posted right now
  - Short written assignment
  - Due next **Tuesday**
- A6 is **also** posted now
  - Due **Sunday** before semis
  - Designed for two weeks
  - But it gets hard at end
  - Get started early!

Video Lessons

- **Videos 20.9-20.10** today
- Also **Lesson 21** for today
- **Lesson 22** for next time



Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    def __repr__(self):
        return 'Example['+str(x)+']'
```

```
>>> a = Example(3)
>>> str(a) # a.__str()__
```

What is the result?

A: '3'

B: 'Value 3'

C: 'Example[3]'

D: **Error**

E: I don’t know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    def __repr__(self):
        return 'Example['+str(x)+']'
```

```
>>> a = Example(3)
>>> str(a)
```

What is the result?

A: '3'

[✓ B: 'Value 3']

C: 'Example[3]'

D: **Error**

E: I don’t know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    def __repr__(self):
        return 'Example['+str(x)+']'
```

```
>>> a = Example(3)
>>> repr(a)
```

What is the result?

A: '3'

B: 'Value 3'

C: 'Example[3]'

D: **Error**

E: I don't know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    def __repr__(self):
        return 'Example['+str(x)+']'
```

No self

```
>>> a = Example(3)
>>> repr(a)
```

What is the result?

A: '3'

B: 'Value 3'

C: 'Example[3]'

[✓ D: **Error**]

E: I don't know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    def __repr__(self):
        return ('Example['+
                      str(self.x)+']')
```

```
>>> a = Example(3)
>>> repr(a)
```

What is the result?

A: '3'

B: 'Value 3'

C: 'Example[3]'

D: **Error**

E: I don't know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    def __repr__(self):
        return ('Example['+
                    str(self.x)+']')
```

```
>>> a = Example(3)
>>> repr(a)
```

What is the result?

A: '3'

B: 'Value 3'

[✓ C: 'Example[3]']

D: **Error**

E: I don’t know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    # No __repr__ definition
```

```
>>> a = Example(3)
>>> repr(a)
```

What is the result?

A: '<Example object ...>'

B: 'Value 3'

C: 'Example[3]'

D: **Error**

E: I don’t know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    def __str__(self):
        return 'Value '+str(self.x)

    # No __repr__ definition
```

```
>>> a = Example(3)
>>> repr(a)
```

What is the result?

[✓ A: '<Example object ...>']

B: 'Value 3'

C: 'Example[3]'

D: **Error**

E: I don't know

Warm-Up: **str and repr**

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    # No __str__ definition


    def __repr__(self):
        return ('Example['+
                    str(self.x)+']')
```

```
>>> a = Example(3)
>>> str(a)
```

What is the result?

A: '<Example object ...>'

B: 'Value 3'

C: 'Example[3]'

D: **Error**

E: I don’t know

Warm-Up: **str and repr**

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    # No __str__ definition


    def __repr__(self):
        return ('Example['+
                     str(self.x)+']')
```

```
>>> a = Example(3)
>>> str(a)
```

What is the result?

A: '<Example object ...>'

B: 'Value 3'

[✓ C: 'Example[3]']

D: **Error**

E: I don’t know

Warm-Up: str and repr

```
class Example(object):
    """A simple class"""

    def __init__(self,x):
        self.x = x

    # No __str__

    def __repr__(self):
        return ('Example['+
                    str(self.x)+']')
```

```
>>> a = Example(3)
>>> str(a)
```

What is the result?

e object ...>'

3]'

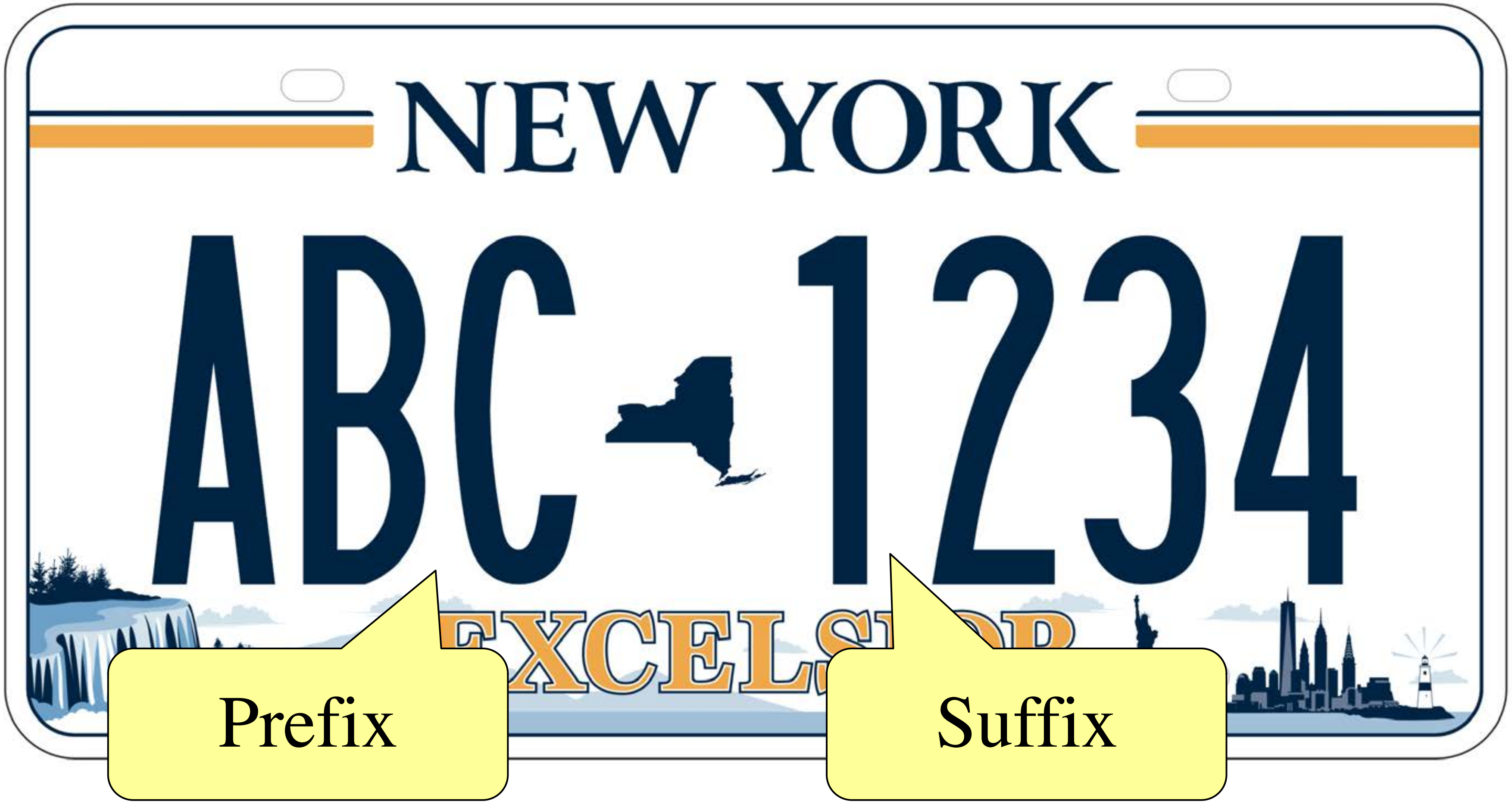
D: **Error**

E: I don't know

str falls back to **repr**

repr has no fall back

Class License



Design Getters and Setters

```
class License(object):
    """A class representing a license plate.

    CLASS ATTRIBUTES (NO GETTERS/SETTERS):
      Attribute USED:  The license plates used so far (initially empty)
      Invariant: USED is a list of (prefix,suffix) pairs"""
    # Attribute_owner: The name of the owner (MUTABLE)
    # Invariant: owner is a nonempty string or None
    # Attribute _prefix: The first half of the licence (IMMUTABLE)
    # Invariant: _prefix is a str of 3 upper case letters
    # Attribute _suffix: The second half of the licence (IMMUTABLE)
    # Invariant: _suffix is an int in 0..9999 inclusive
```

Complete the Initializer

```
class License(object):
    …
    def __init__(self, ???):
        """Initializes a license plate with the given prefix and suffix.

        No license plate can be created if it has the same prefix and suffix as
        an existing plate (this will cause an AssertionError). On creation, the
        pair (prefix,suffix) is added to the class attribute USED to ensure this

        Precond: prefix is a string of 3 upper case letters
        Precond: suffix is an int in 0..9999, inclusive
        Precond: owner a nonempty string or None (Optional; default None)"""
```

Complete the Initializer

```
class License(object):
    ...
    def __init__(self, ???):
        """Initializes a license

        No license plate can b
        an existing plate (this
        pair (prefix,suffix) is

        Precond: prefix is a st
        Precond: suffix is an i
        Precond: owner a non
```

What are params?

A: (prefix,suffix)

B: (self,prefix,suffix)

C: (self,prefix,suffix,owner)

D: (self,prefix,suffix,owner=None)

E: Unsure

Complete the Initializer

```
class License(object):
    ...
    def __init__(self, ???):
        """Initializes a license

        No license plate can b
        an existing plate (this
        pair (prefix,suffix) is

        Precond: prefix is a st
        Precond: suffix is an i
        Precond: owner a non
```

What are params?

A: (prefix,suffix)

B: (self,prefix,suffix)

C: (self,prefix,suffix,owner)

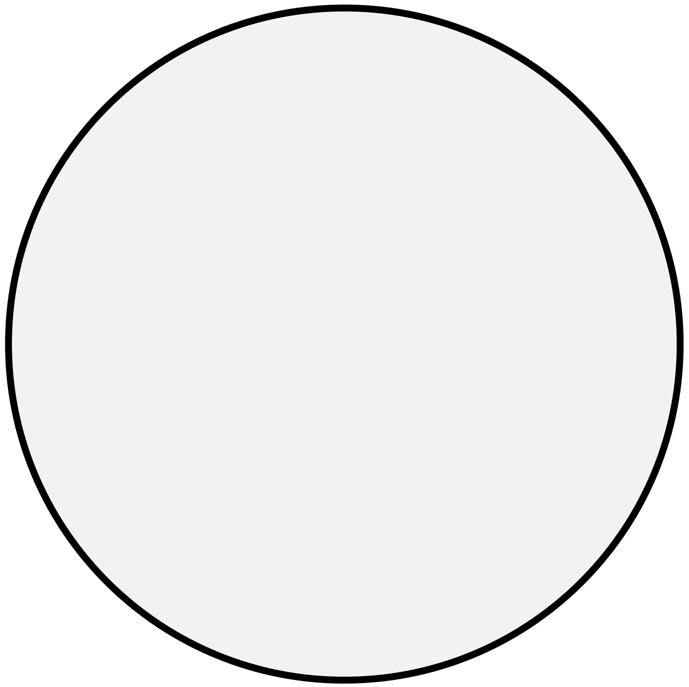
[✓ D: (self,prefix,suffix,owner=None)]

E: Unsure

Implement the __str__ Method

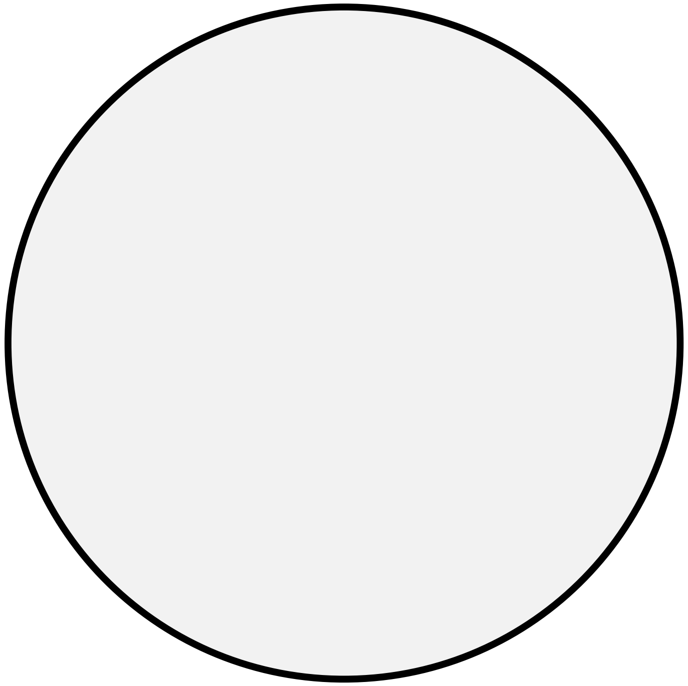
```python
class License(object):
    ...
    def __str__(self):
        """Returns a string representation of this license plate.

        The string is of the form prefix-suffix. The suffix is padded with
        leading 0s to have three characters. If the plate has an owner, the
        owner follows the string in parentheses. Otherwise, nothing is added
        to the string.

        Example: 'ABC-001' if no owner
                 'XYZ-093 (Bob)' for owner Bob"""
```

Design Time: A (2D) Circle



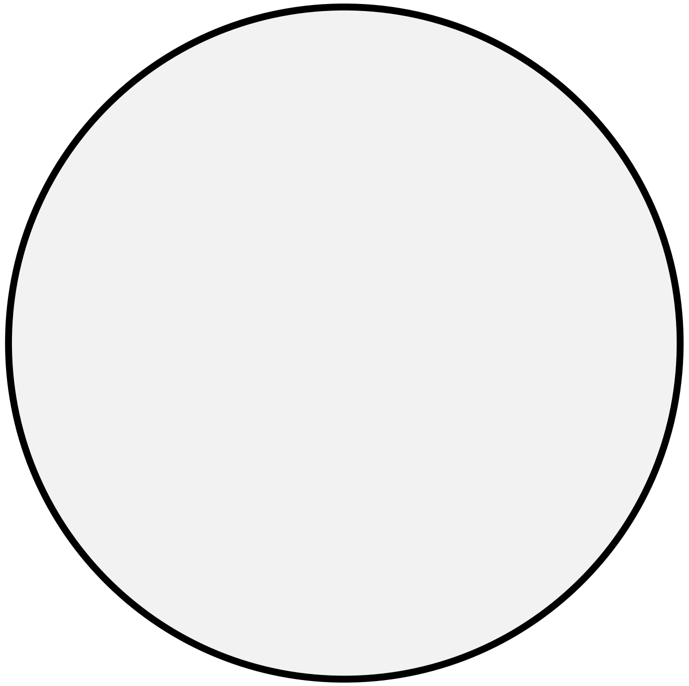
What are the Attributes?

Design Time: A (2D) Circle



What are the Invariants?

Design Time: A (2D) Circle



What are the Methods?

Questions?