[320] Inheritance

Tyler Caraza-Harter

Review

Review Classes + Special Methods

```python
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self, mult, ucase):
        msg = "bark " * mult
        if ucase:
            msg = msg.upper()
        print(self.name + ": " + msg)

sam = Dog("Fido")
fido = Dog("Sam")
```

```python
fido.bark(5, False)                # 1
fido.bark(fido, 5, True)           # 2
fido.bark(fido, 5, True, None)     # 3
```

which call produces the following error?

```
TypeError: bark() takes 3 positional arguments but 4 were given
```

Review Classes + Special Methods

```python
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self, mult, ucase):
        msg = "bark " * mult
        if ucase:
            msg = msg.upper()
        print(self.name + ": " + msg)

sam = Dog("Fido")
fido = Dog("Sam")
```

```python
fido.bark(5, False)                 # 1
fido.bark(fido, 5, True)            # 2
fido.bark(fido, 5, True, None)      # 3
```

which call is correct?

Review Classes + Special Methods

```
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self, mult, ucase):
        msg = "bark " * mult
        if ucase:
            msg = msg.upper()
        print(self.name + ": " + msg)

sam = Dog("Fido")
fido = Dog("Sam")

fido.bark(5, False)                    # 1
```

what is printed?

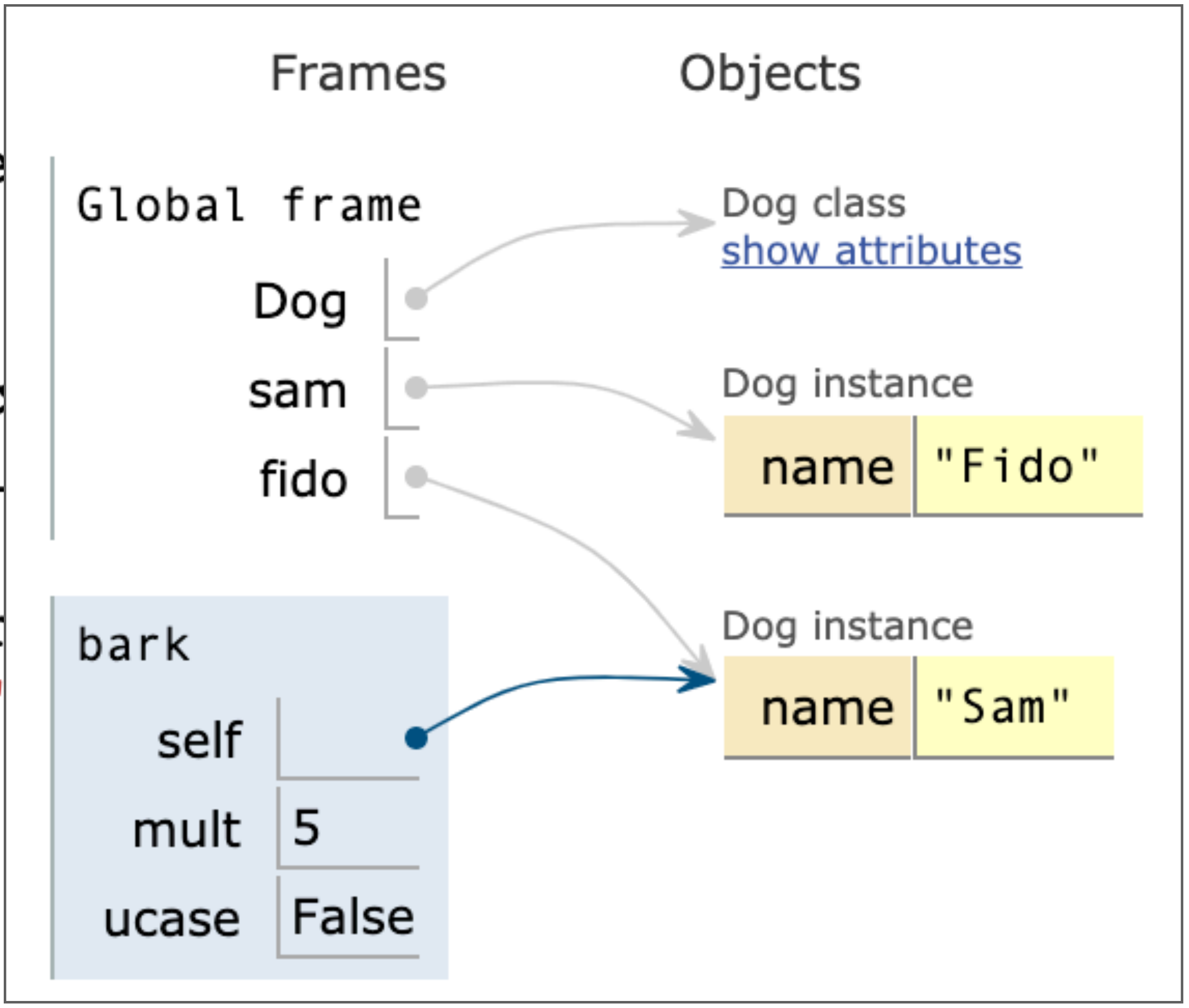
```
(1) Fido: bark bark bark bark bark
(2) Fido: BARK BARK BARK BARK BARK
(3) Sam: bark bark bark bark bark
```

Review Classes + Special Methods

```
class Dog:
    def __init__(self, name
        self.name = name

    def bark(self, mult, uc
        msg = "bark " * mul
        if ucase:
            msg = msg.upper
        print(self.name + "

sam = Dog("Fido")
fido = Dog("Sam")
```



```
fido.bark(5, False)                    # 1
```

what is printed?

```
(1) Fido: bark bark bark bark bark
(2) Fido: BARK BARK BARK BARK BARK
(3) Sam: bark bark bark bark bark
```

Review Classes + Special Methods

Special methods usually get called

1. explicitly
2. implicitly

What does **print(...)** use to represent an object?

1. __str__
2. __repr__
3. _repr_html_

What special method must be implemented for **sorting** to work?

1. __repr__
2. __order__
3. __lt__
4. __gt__

Review Classes + Special Methods

```
from math import *

class ContinuousList:
    def __init__(self, L):
        self.L = L

    def __getitem__(self, pos):
        assert 0 <= pos <= len(self.L) - 1
        idx1 = floor(pos) # round down
        idx2 = ceil(pos)  # round up
        v1 = self.L[idx1]
        v2 = self.L[idx2]
        diff = v2 - v1
        return v1 + (pos - idx1) * diff

clist = ContinuousList([7, 8, 9, 100, 200])
x = clist[3.2]
y = clist[1:3]
```

what will **x** be? (there **won't** be an error)

Review Classes + Special Methods

```
from math import *

class ContinuousList:
    def __init__(self, L):
        self.L = L

    def __getitem__(self, pos):
        assert 0 <= pos <= len(self.L) - 1
        idx1 = floor(pos) # round down
        idx2 = ceil(pos)  # round up
        v1 = self.L[idx1]
        v2 = self.L[idx2]
        diff = v2 - v1
        return v1 + (pos - idx1) * diff

clist = ContinuousList([7, 8, 9, 100, 200])
x = clist[3.2]
y = clist[1:3]
```

$\leftarrow$ what will **pos** be? (there **will** be an error)

Inheritance

Coding Examples

Principals

- method inheritance
- method resolution order
- overriding methods, constructor
- calling overridden methods
- abc's (abstract base classes)