

# INHERITANCE: SUBCLASS

inherits all attributes of Animal:

`__init__()`  
`age, name`  
`get_age(), get_name()`  
`set_age(), set_name()`  
`__str__()`

```
class Cat(Animal):
```

```
    def speak(self):
```

```
        print("meow")
```

```
    def __str__(self):
```

```
        return "cat:" + str(self.name) + ":" + str(self.age)
```

add new  
functionality via  
speak method

overrides `__str__`

- add new functionality with `speak()`
  - instance of type `Cat` can be called with new methods
  - instance of type `Animal` throws error if called with `Cat`'s new method
- `__init__` is not missing, uses the `Animal` version

# WHICH METHOD TO USE?

---

- subclass can have **methods with same name** as superclass
- for an instance of a class, look for a method name in **current class definition**
- if not found, look for method name **up the hierarchy** (in parent, then grandparent, and so on)
- use first method up the hierarchy that you found with that method name

```
class Person(Animal):
```

```
    def __init__(self, name, age):
```

```
        Animal.__init__(self, age)
```

```
        self.set_name(name)
```

```
        self.friends = []
```

```
    def get_friends(self):
```

```
        return self.friends
```

```
    def add_friend(self, fname):
```

```
        if fname not in self.friends:
```

```
            self.friends.append(fname)
```

```
    def speak(self):
```

```
        print("hello")
```

```
    def age_diff(self, other):
```

```
        diff = self.age - other.age
```

```
        print(abs(diff), "year difference")
```

```
    def __str__(self):
```

```
        return "person:" + str(self.name) + ":" + str(self.age)
```

*parent class is Animal*

*call Animal constructor  
call Animal's method  
add a new data attribute*

*new methods*

*override Animal's  
\_\_str\_\_ method*

```
import random
```

```
class Student(Person):
```

```
    def __init__(self, name, age, major=None):
```

```
        Person.__init__(self, name, age)
```

```
        self.major = major
```

```
    def change_major(self, major):
```

```
        self.major = major
```

```
    def speak(self):
```

```
        r = random.random()
```

```
        if r < 0.25:
```

```
            print("i have homework")
```

```
        elif 0.25 <= r < 0.5:
```

```
            print("i need sleep")
```

```
        elif 0.5 <= r < 0.75:
```

```
            print("i should eat")
```

```
        else:
```

```
            print("i am watching tv")
```

```
    def __str__(self):
```

```
        return "student:" + str(self.name) + ":" + str(self.age) + ":" + str(self.major)
```

bring in methods  
from random class

inherits Person and  
Animal attributes

adds new data

- I looked up how to use the  
random class in the python docs  
- random() method gives back  
float in [0, 1)