

Rabbit GETTER METHODS

```
class Rabbit(Animal):
    tag = 1
    def __init__(self, age, parent1=None, parent2=None):
        Animal.__init__(self, age)
        self.parent1 = parent1
        self.parent2 = parent2
        self.rid = Rabbit.tag
        Rabbit.tag += 1
    def get_rid(self):
        return str(self.rid).zfill(3)
    def get_parent1(self):
        return self.parent1
    def get_parent2(self):
        return self.parent2
```

method on a string to pad
the beginning with zeros
for example, 001 not 1

- getter methods specific
for a Rabbit class
- there are also getters
get_name and get_age
inherited from Animal

WORKING WITH YOUR OWN TYPES

```
def __add__(self, other):  
    # returning object of same type as this class  
    return Rabbit(0, self, other)
```

recall Rabbit's `__init__(self, age, parent1=None, parent2=None)`

- define **+ operator** between two `Rabbit` instances
 - define what something like this does: `r4 = r1 + r2`
where `r1` and `r2` are `Rabbit` instances
 - `r4` is a new `Rabbit` instance with age 0
 - `r4` has `self` as one parent and `other` as the other parent
 - in `__init__`, **parent1 and parent2 are of type `Rabbit`**

SPECIAL METHOD TO COMPARE TWO Rabbits

- decide that two rabbits are equal if they have the **same two parents**

booleans

```
def __eq__(self, other):  
    parents_same = self.parent1.rid == other.parent1.rid \  
                   and self.parent2.rid == other.parent2.rid  
    parents_opposite = self.parent2.rid == other.parent1.rid \  
                      and self.parent1.rid == other.parent2.rid  
    return parents_same or parents_opposite
```

- compare ids of parents since **ids are unique** (due to class var)
- note you can't compare objects directly
 - for ex. with `self.parent1 == other.parent1`
 - this calls the `__eq__` method over and over until call it on `None` and gives an `AttributeError` when it tries to do `None.parent1`