

Contact

Dr. James Shackelford
shack@drexel.edu
Bossone 211

Office Hours: 3 – 4 pm (Tuesday)
Course Website: <http://learn.dcollege.net>

Textbook

Think Python
by Allen Downey
O'Reilly Press, 2015
ISBN-13: 978-1449330729
(Freely available in PDF format, check course website)



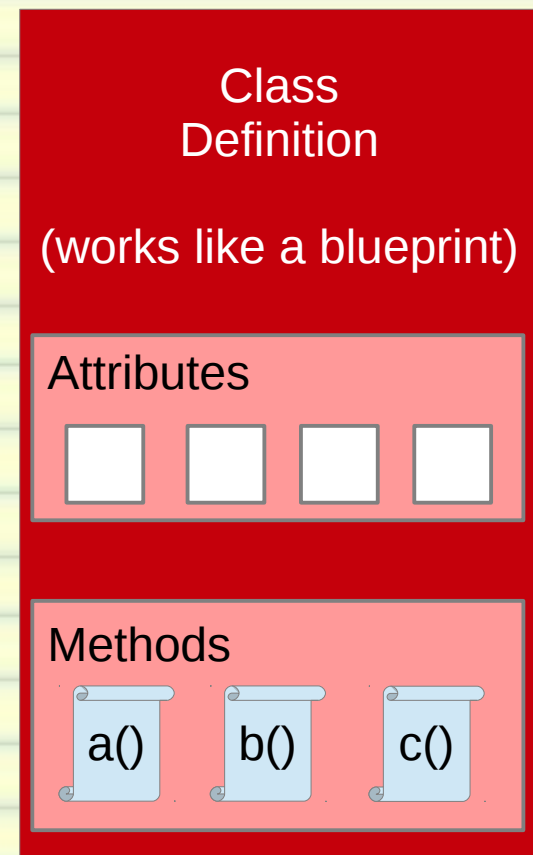
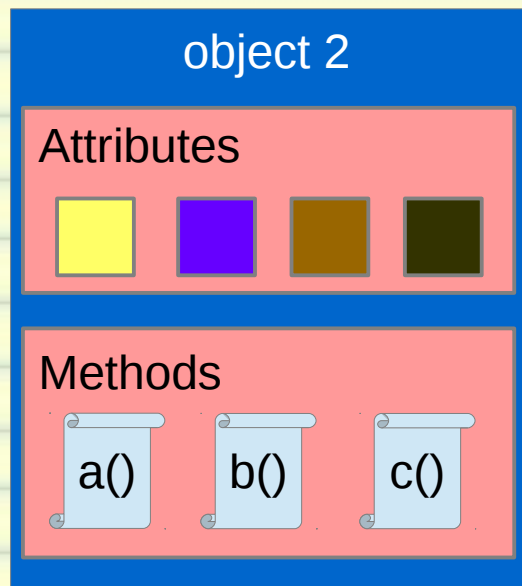
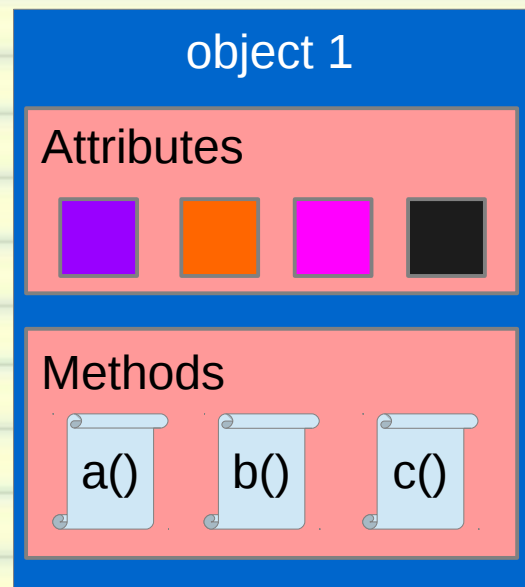
Grading

- 10% In-lab Programming Assignments
- 10% Take-Home Programming Assignments
- 35% Mid-term Exam
- 45% Final Exam

Introduction to Object Oriented Programming

Classes, Objects, Instances,
Methods, and Attributes

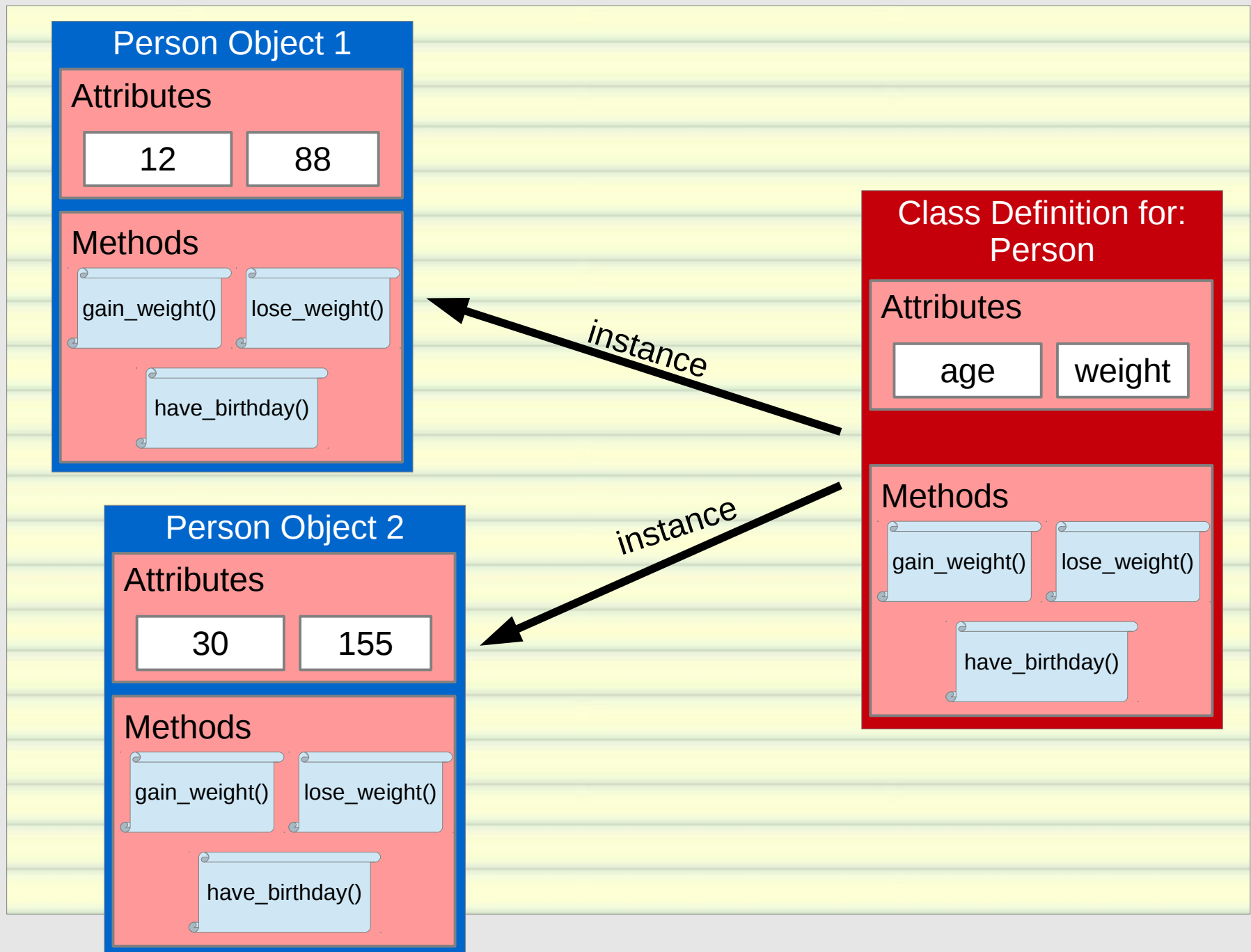
The Big Idea



instance

instance

The Big Idea



Class Definition

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```

class name

Class Definition

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```



class docstring

Class Definition

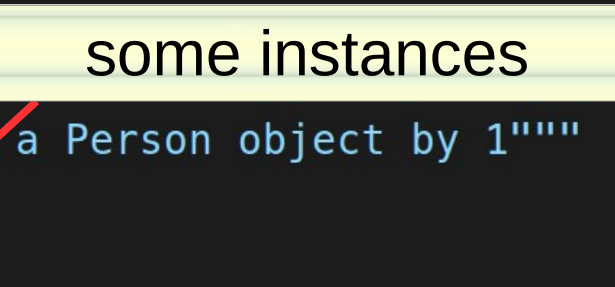
```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

methods



Class Definition

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```



A yellow rectangular box containing the text "some instances" is positioned to the right of the code. A red arrow points from this box to a red rectangular box that encloses the last two lines of code: `joe = Person(12, 88)` and `bob = Person(30, 155)`.

Class Definition

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```

called when object
is created

...Or...

instantiated

Class Definition

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```

The diagram illustrates the relationship between a class instantiation and a method call. Two red arrows originate from the arguments '12' and '88' in the `Person(12, 88)` call on line 24. One arrow points to the `age` parameter in the `__init__` method definition on line 8. The other arrow points to the `weight` parameter in the `__init__` method definition on line 8. Additionally, a red rectangle highlights the `Person(12, 88)` expression on line 24.

Class Definition

???

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```

Class Definition

this is the *instance*!

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```

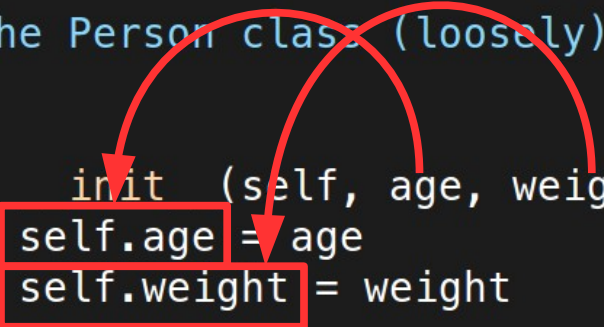

Instance/Object Attributes

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

these are instance
attributes

Initializing Objects

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def init (self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
```




we are
initializing them

Accessing Object Attributes

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    print joe.age
28    print bob.weight
```

accessing
attributes is easy



Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    joe.have_birthday()
28
```

calling methods
is also easy

Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    joe.have_birthday()
28    Person.have_birthday(joe)
```

btw. this is the same
(but uncommon)

Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    joe.have_birthday()
28    Person.have_birthday(joe)
```

here we are
explicitly passing
the instance

Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    joe.have_birthday()
28    Person.have_birthday(joe)
```

here, it is passed
implicitly

A yellow rectangular box with a thin black border contains the text "here, it is passed implicitly". Two red arrows originate from the box. One arrow points to the docstring of the `have_birthday` method in the `Person` class (line 21). The other arrow points to the `joe` object in the method call `joe.have_birthday()` on line 27, which is enclosed in a red rectangular box.

Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    bob.lose_weight(5)
28    Person.lose_weight(bob, 5)
```

same stuff going on
here, but with an
argument

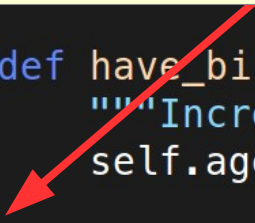
Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12        fine.
13        ...but if I want to define
14        more people, I need
15        rewrite the program
16
17        def have_birthday(self):
18            """Increase the age of a Person object by 1"""
19            self.age += 1
20
21        joe = Person(12, 88)
22        bob = Person(30, 155)
23
24        bob.lose_weight(5)
25        Person.lose_weight(bob, 5)
```

Calling Methods

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12        def lose_weight(self, amount):
13            """Decrease the weight of a Person object by amount"""
14
15            self.weight -= amount
16
17            def gain_weight(self, amount):
18                """Increase the weight of a Person object by amount"""
19
20                self.weight += amount
21
22        def have_birthday(self):
23            """Increase the age of a Person object by 1"""
24            self.age += 1
25
26        joe = Person(12, 88)
27        bob = Person(30, 155)
28
29        bob.lose_weight(5)
30        Person.lose_weight(bob, 5)
```

Problem
variables (i.e. names) must be defined at 'compile time'



Using Dictionaries of Instances

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def get_person(self):
13        """Return a Person object"""
14        return self
15
16    def __str__(self):
17        """Return a string representation of a Person object"""
18        return f'Person({self.age}, {self.weight})'
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    people = {}
25
26    people['joe'] = Person(12, 88)
27    people['bob'] = Person(30, 155)
28
29    people['bob'].have_birthday()
30    print people['bob'].age
```

Solution:
use a dictionary



Using Dictionaries of Instances

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Increase the weight of a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Decrease the weight of a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24 people = {}
25
26 people['joe'] = Person(12, 88)
27 people['bob'] = Person(30, 155)
28
29 people['bob'].have_birthday()
30 print people['bob'].age
```

Solution:

'keys' can be defined at 'runtime' (!!)



Using Dictionaries of Instances

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to the person"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from the person"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24 people = {}
25
26 people['joe'] = Person(12, 88)
27 people['bob'] = Person(30, 155)
28
29 people['bob'].have_birthday()
30 print people['bob'].age
```

Solution:

the 'value' associated
with each 'key' is an
instance of Person



Using Dictionaries of Instances

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to the person"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from the person"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    people = {}
25
26    people['joe'] = Person(12, 88)
27    people['bob'] = Person(30, 155)
28
29    people['bob'].have_birthday()
30    print people['bob'].age
```

Analysis:

So, `people['bob']` is an instance of `Person`



Using Dictionaries of Instances

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to the person"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from the person"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    people = {}
25
26    people['joe'] = Person(12, 88)
27    people['bob'] = Person(30, 155)
28
29    people['bob'].have_birthday()
30    print people['bob'].age
```

Analysis:

This calls the **have_birthday()** method for this Person instance



Now we just need a function that can:

- 1) create a Person instance
- 2) add the instance to a dictionary

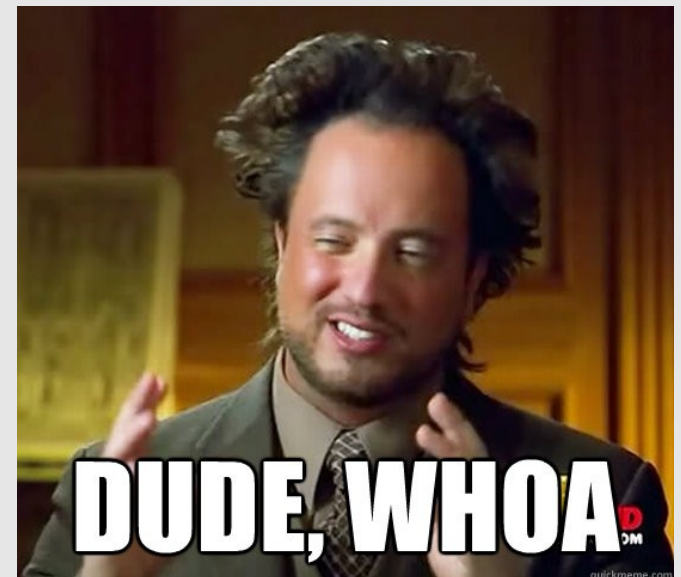
Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("    Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("    Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Solution:

This implements
both requirements

what's all that other stuff?

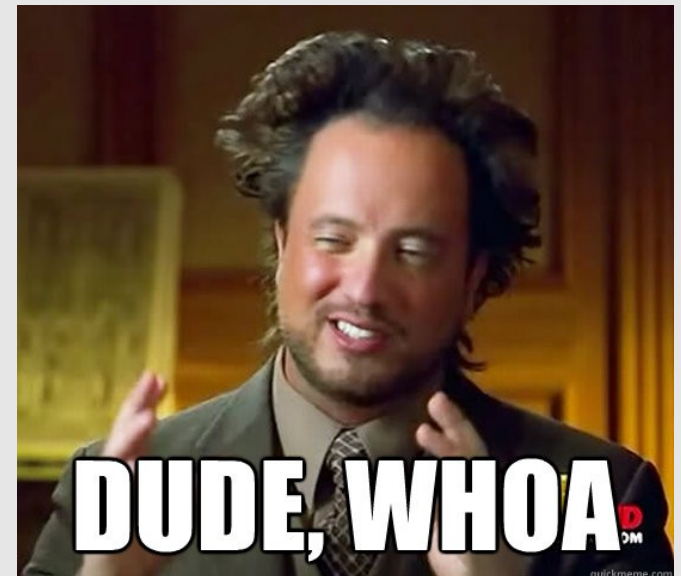


Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Getting User Input:

Used to store input from user about a Person

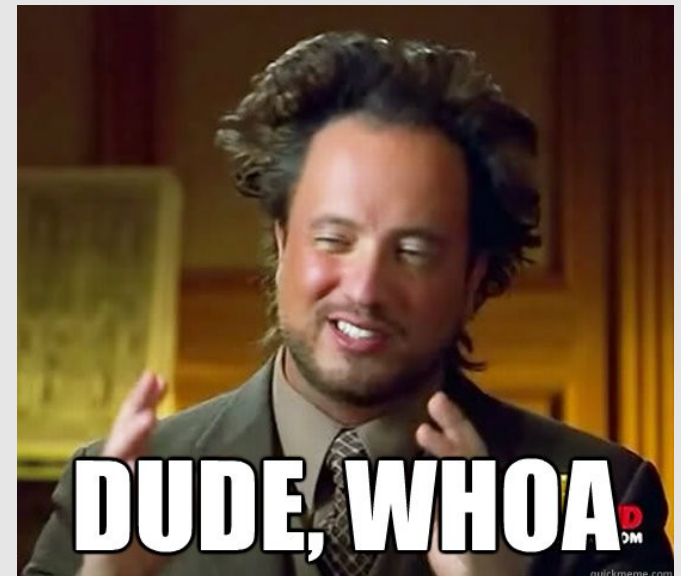


Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Getting User Input:

Get user input until they “just hit enter” at the **Name:** prompt without providing a name.

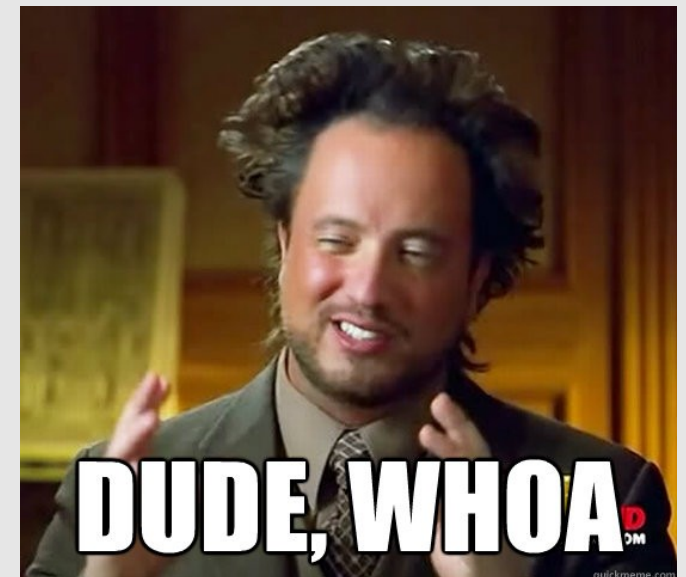


Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Getting User Input:

Collect person attributes from user via keyboard



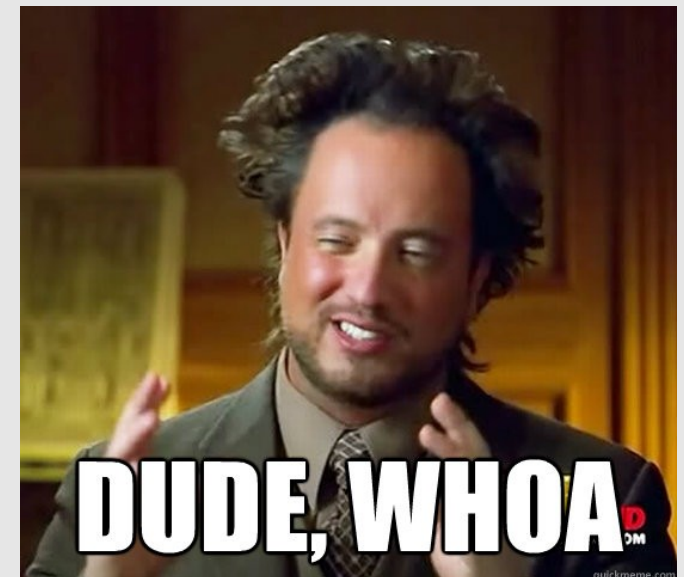
Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("    Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("    Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Adding the Person:

Converts person dictionary into a Person instance and adds it to the...

people dictionary !



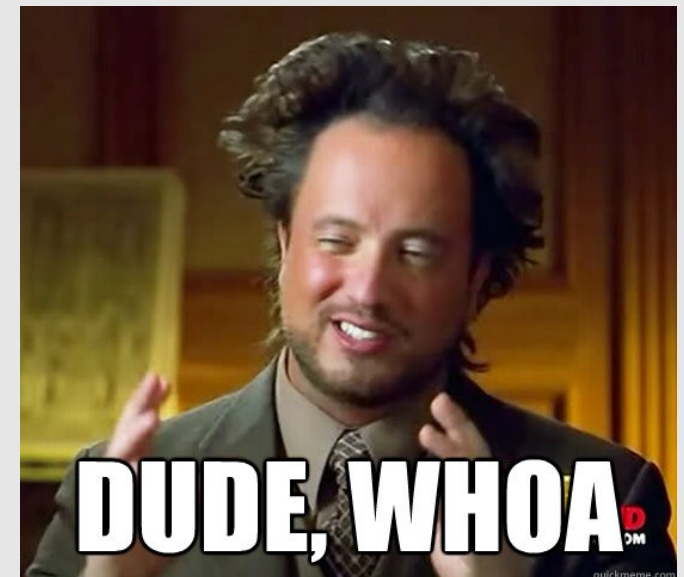
Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Adding the Person:

This is a **string** containing the name entered by the user

this string will be a **key**



Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Adding the Person:

Create a Person instance with the **age** and **weight** values entered by the user



Building Dictionaries of Instances from User Input

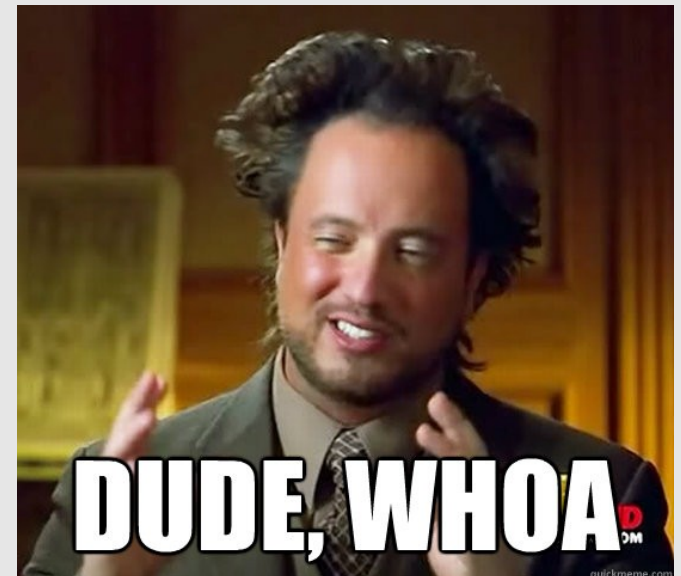
```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Adding the Person:

Insert the instance into the dictionary.

We have something like:

{'bob': <Person instance>}

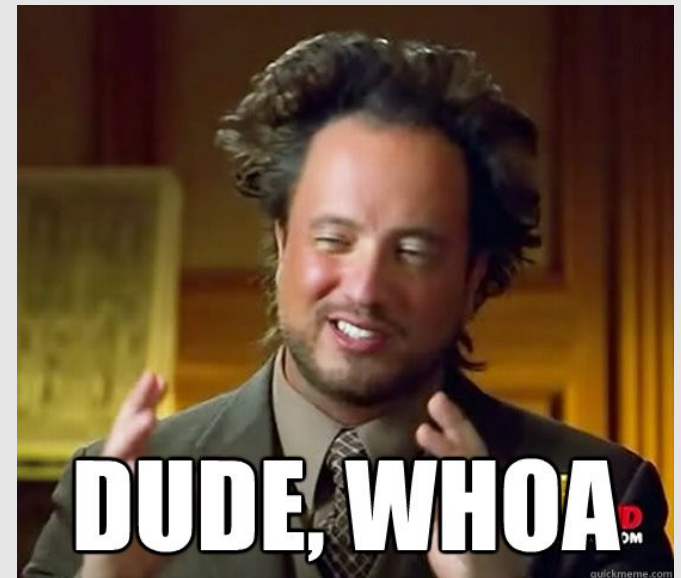


Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Listing People:

Simply cycles through the dictionary and prints all the People we have inserted



Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

Wait.

There is a pattern here.



Building Dictionaries of Instances from User Input

```
1 from our_example import Person
2
3
4 def add_person(people, p):
5     people[p['name']] = Person(p['age'], p['weight'])
6
7
8 def list_people(people):
9     for name, person in people.items():
10         print name
11         print '    age: %s' % person.age
12         print '    weight: %s' % person.weight
13
14
15 if __name__ == '__main__':
16     people = {}
17     person = {}
18     while True:
19         print '-----'
20         person['name'] = raw_input("  Name: ")
21         if person['name'] == '':
22             break
23
24         person['age'] = raw_input("    Age: ")
25         person['weight'] = raw_input("Weight: ")
26         add_person(people, person)
27
28     list_people(people)
```

**These functions all
operate on people**

**maybe People
should be an Object**

```
1 class Person:
2     """
3     This is the docstring for the Person class!!
4
5     The Person class (loosely) represent a person.
6     """
7
8     def __init__(self, age, weight):
9         self.age = age
10        self.weight = weight
11
12    def gain_weight(self, amount):
13        """Add 'amount' weight to a Person object"""
14        self.weight += amount
15
16    def lose_weight(self, amount):
17        """Remove 'amount' weight from a Person object"""
18        self.weight -= amount
19
20    def have_birthday(self):
21        """Increase the age of a Person object by 1"""
22        self.age += 1
23
24    joe = Person(12, 88)
25    bob = Person(30, 155)
26
27    joe.have_birthday()
28    Person.have_birthday(joe)
```


An example of encapsulation

```
1 from our_example import Person
2
3 class People:
4
5     def __init__(self):
6         self.people = {}
7
8     def add(self):
9         print '-----'
10        name = raw_input(" Name: ")
11        if name == '':
12            return False
13
14        age = raw_input(" Age: ")
15        weight = raw_input("Weight: ")
16
17        self.people[name] = Person(age, weight)
18        return True
19
20    def show(self):
21        for name, person in self.people.items():
22            print name
23            print '    age: %s' % person.age
24            print '    weight: %s' % person.weight
25
26
27 if __name__ == '__main__':
28     people = People()
29
30     while people.add():
31         pass
32
33     people.show()
```

People as a Class:

Encapsulates a dictionary
of Person instances

users of the People class
never access the dictionary
of instances directly

...they use the methods
provided by the People class
instead.