ECE-203 – Programming for Engineers

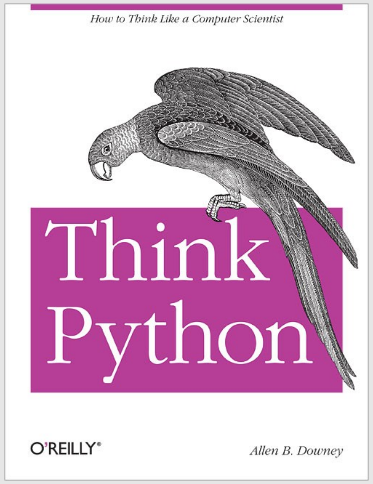
Contact

Dr. James Shackleford
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

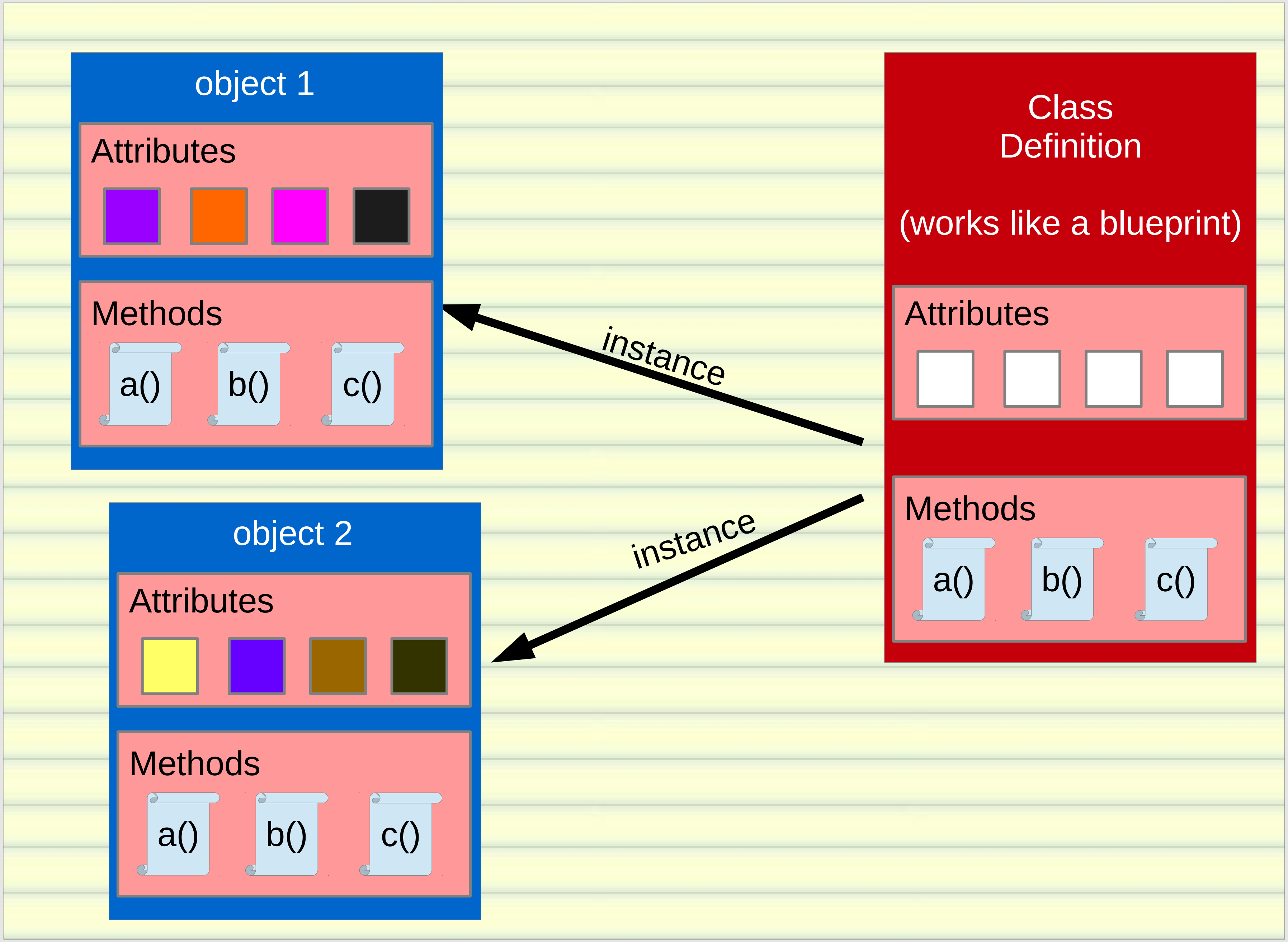
- 10% In-lab Programming Assignments
- 10% Take-Home Programming Assignments
- 35% Mid-term Exam
- 45% Final Exam

New Topic

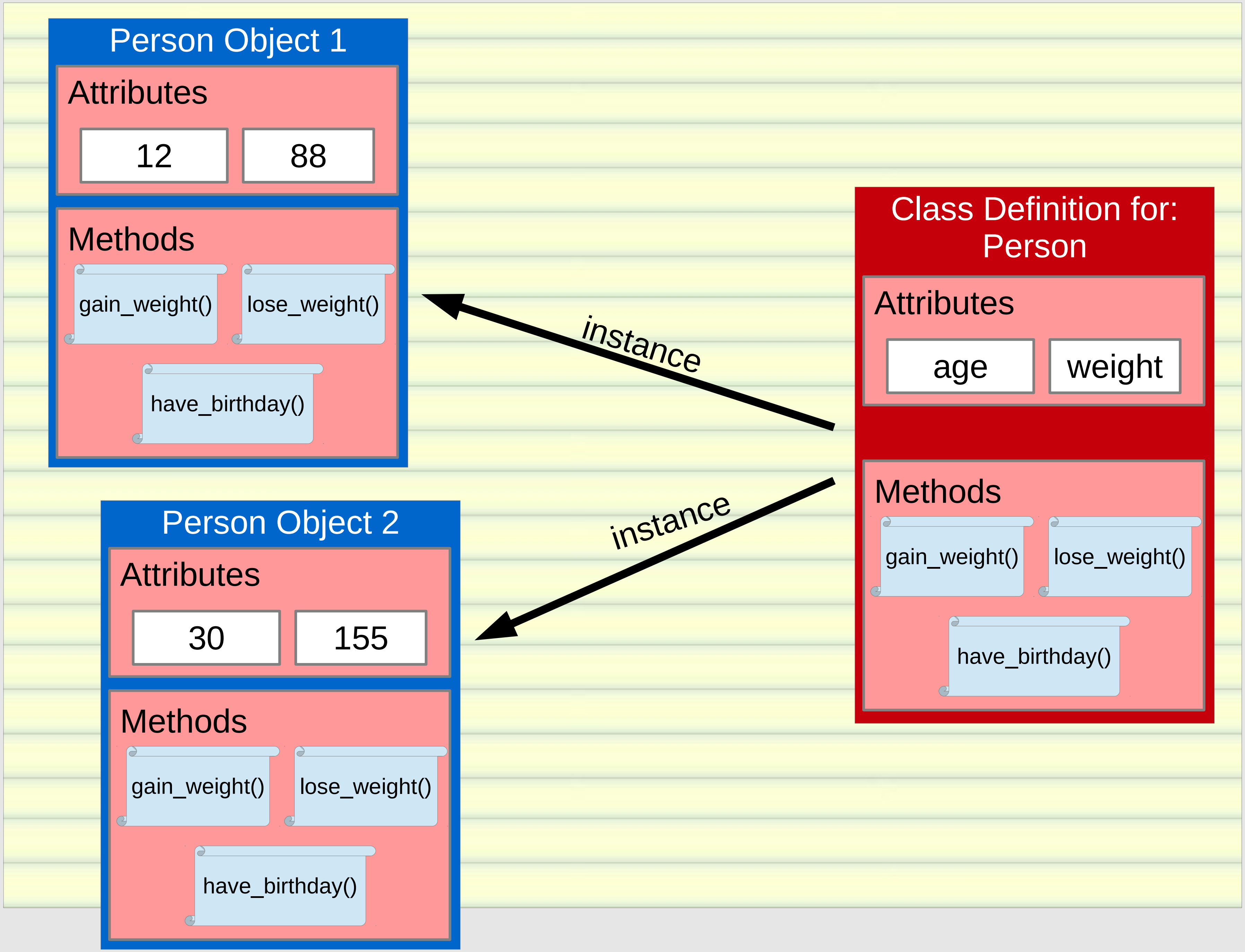
Introduction to
Object Oriented Programming

Classes, Objects, Instances,
Methods, and Attributes

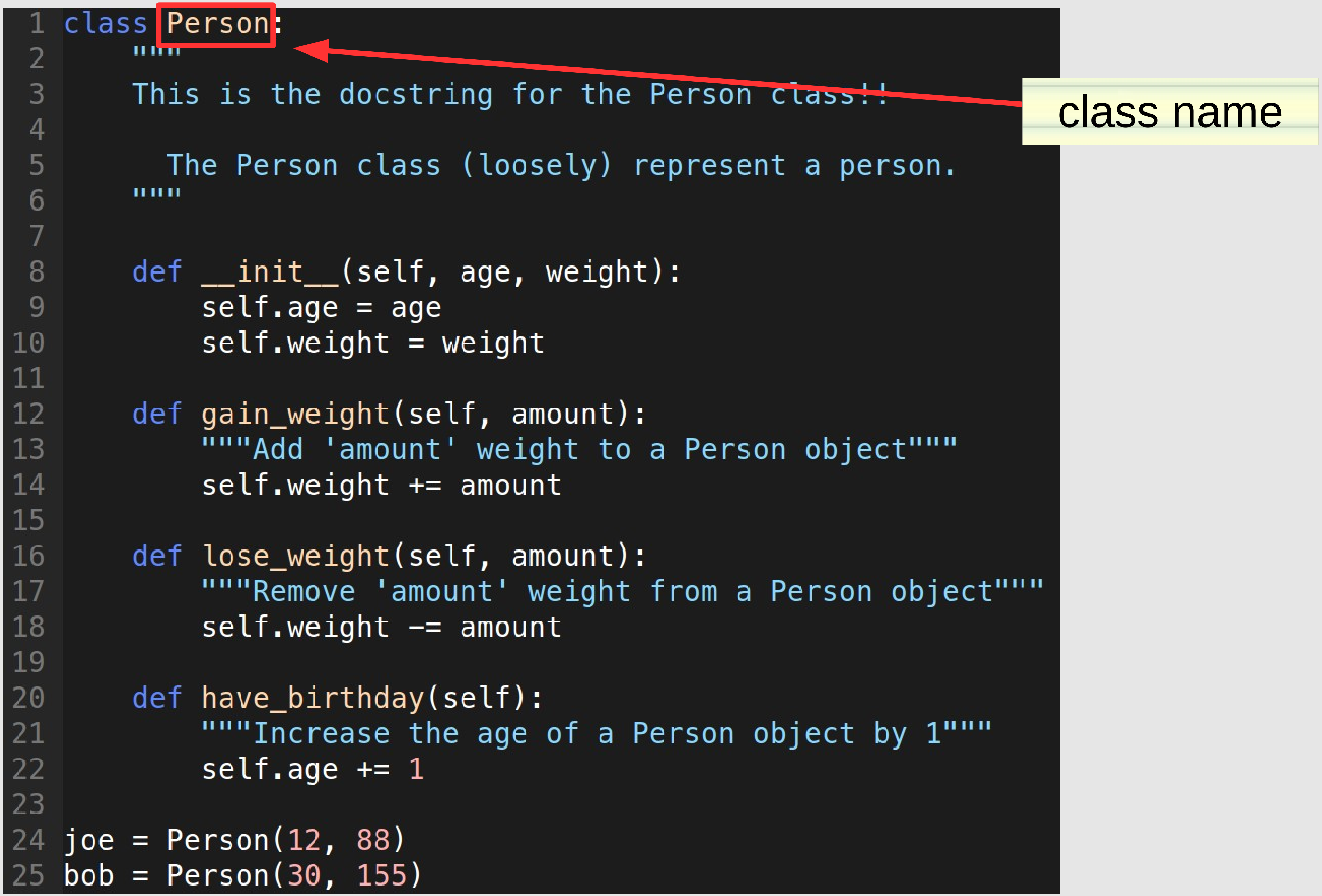
The Big Idea


object 1
Attributes
Methods
a()
b()
c()
object 2
Attributes
Methods
a()
b()
c()
instance
instance
Class
Definition
(works like a blueprint)
Attributes
Methods
a()
b()
c()

The Big Idea

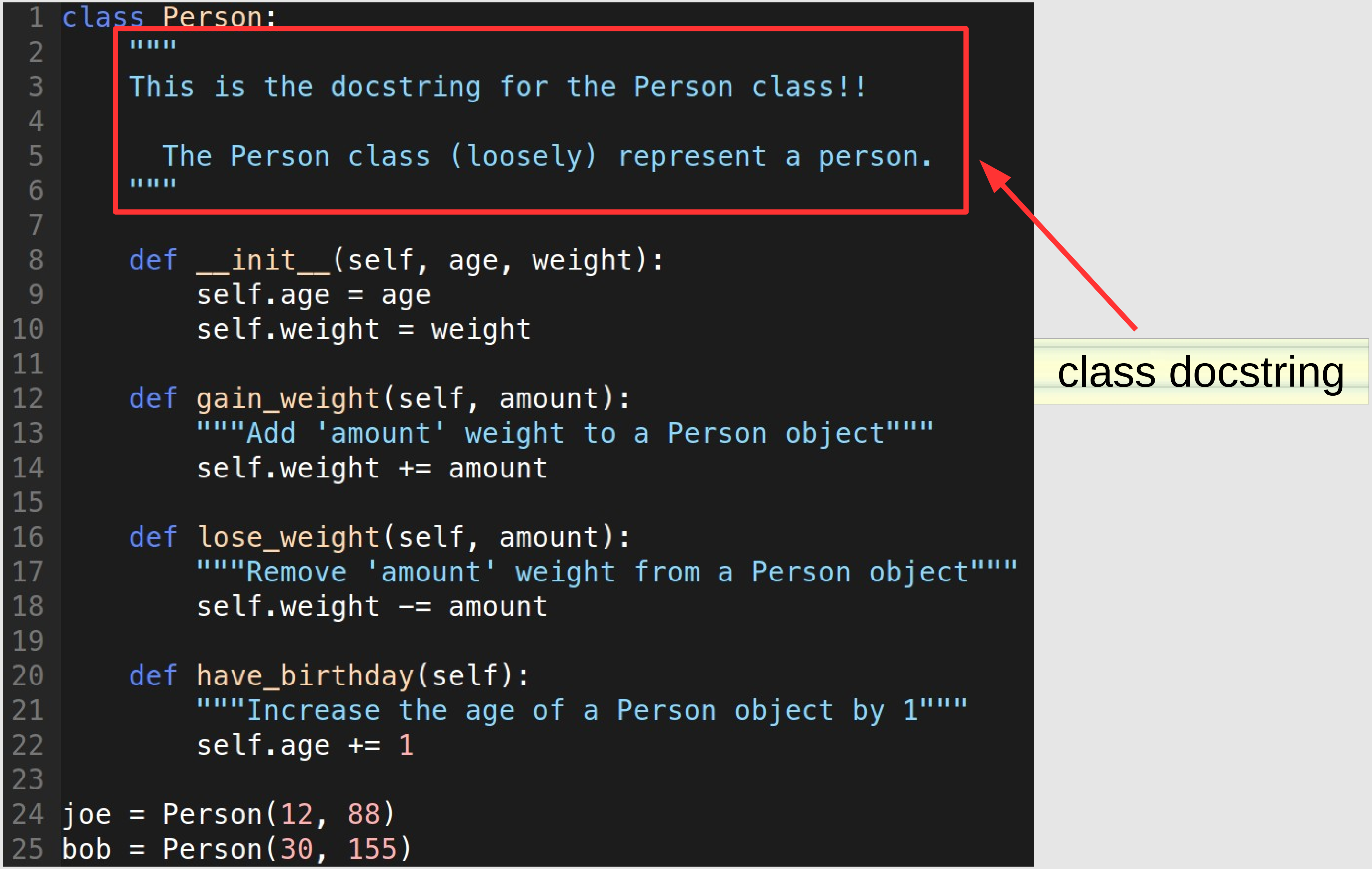

Person Object 1
Attributes
12
88
Methods
gain_weight()
lose_weight()
have_birthday()
Person Object 2
Attributes
30
155
Methods
gain_weight()
lose_weight()
have_birthday()
instance
instance
Class Definition for: Person
Attributes
age
weight
Methods
gain_weight()
lose_weight()
have_birthday()

Class Definition



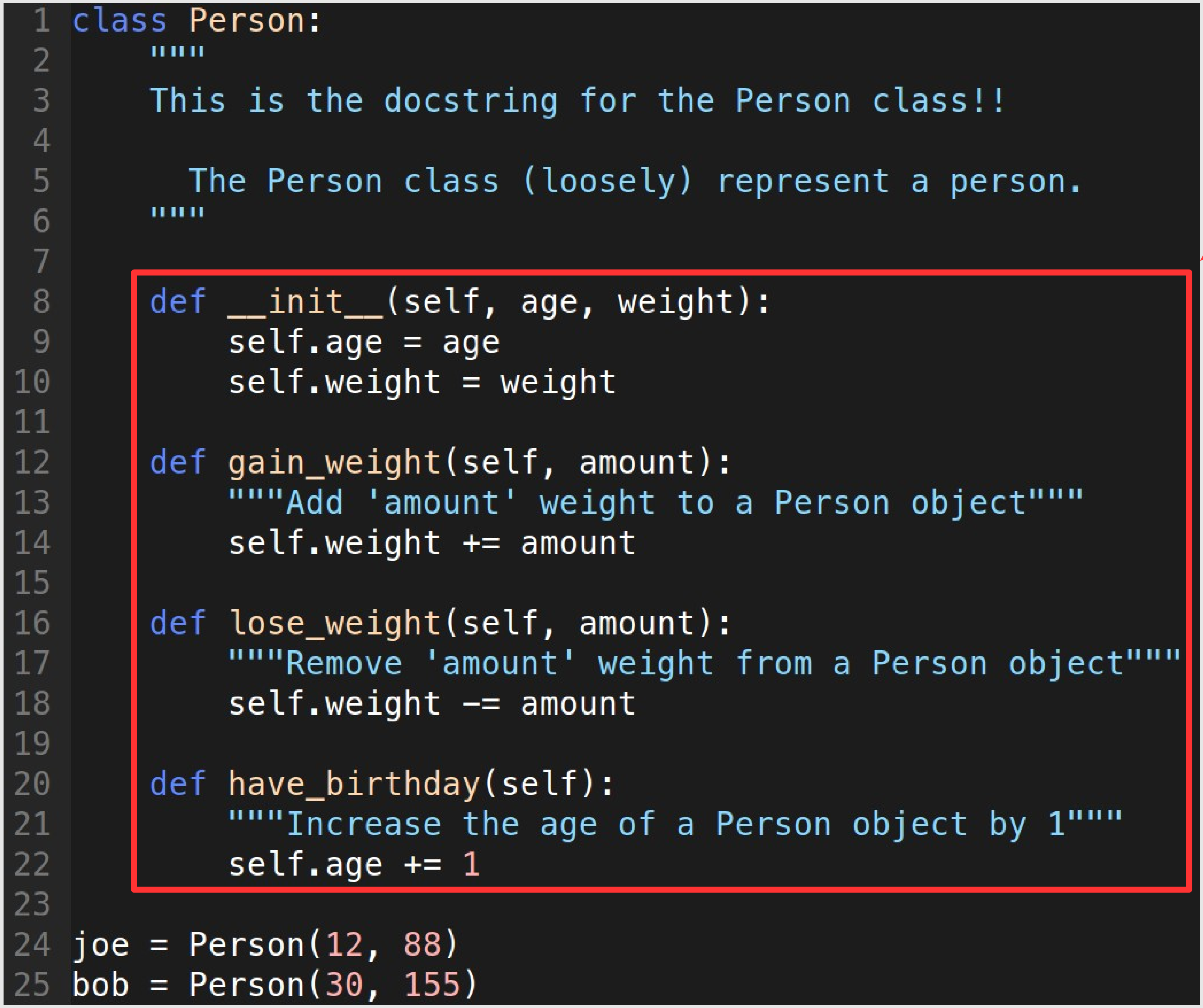
```python
class Person:
    """
    This is the docstring for the Person class!!

     The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

Class Definition



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

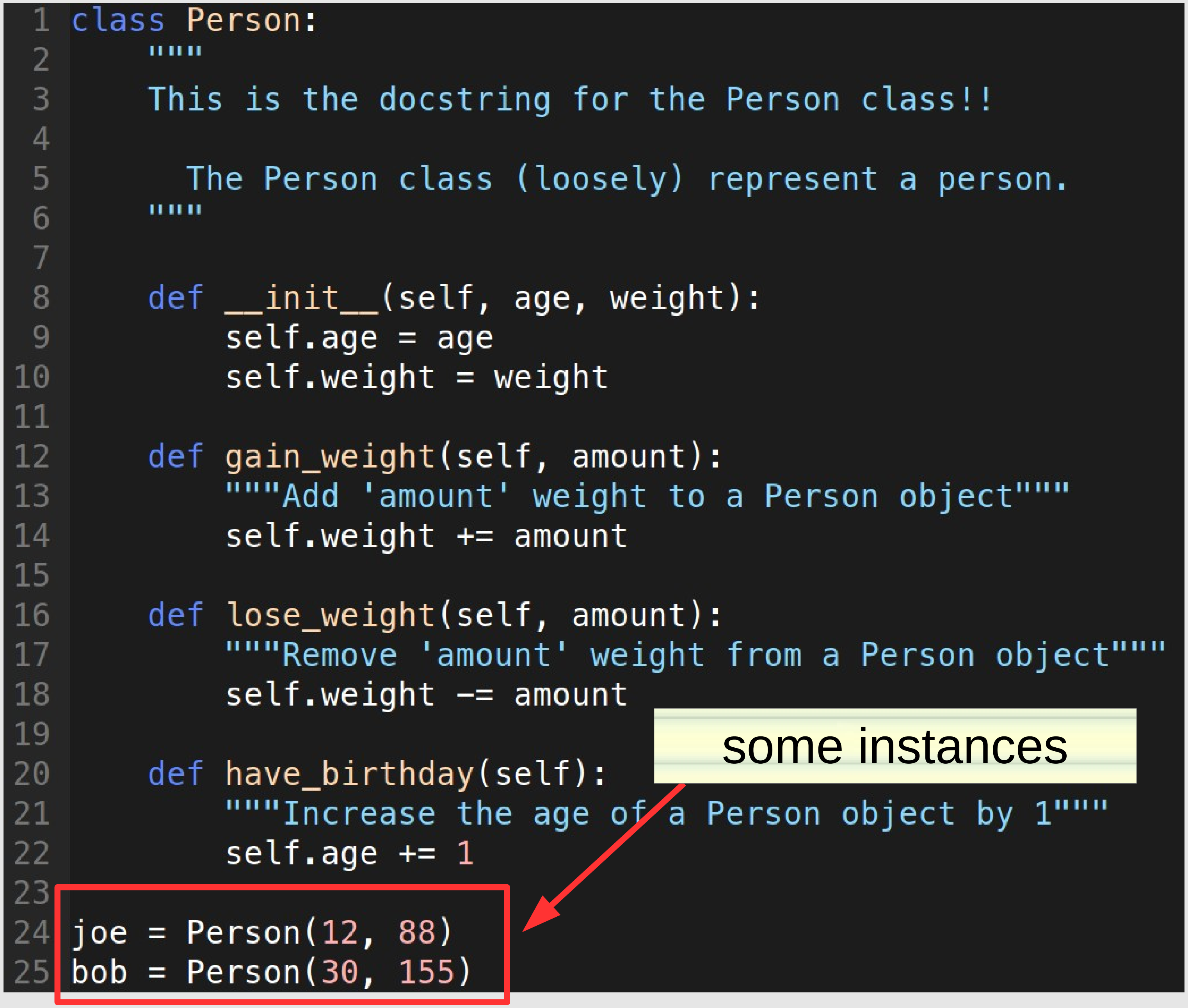
Class Definition



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

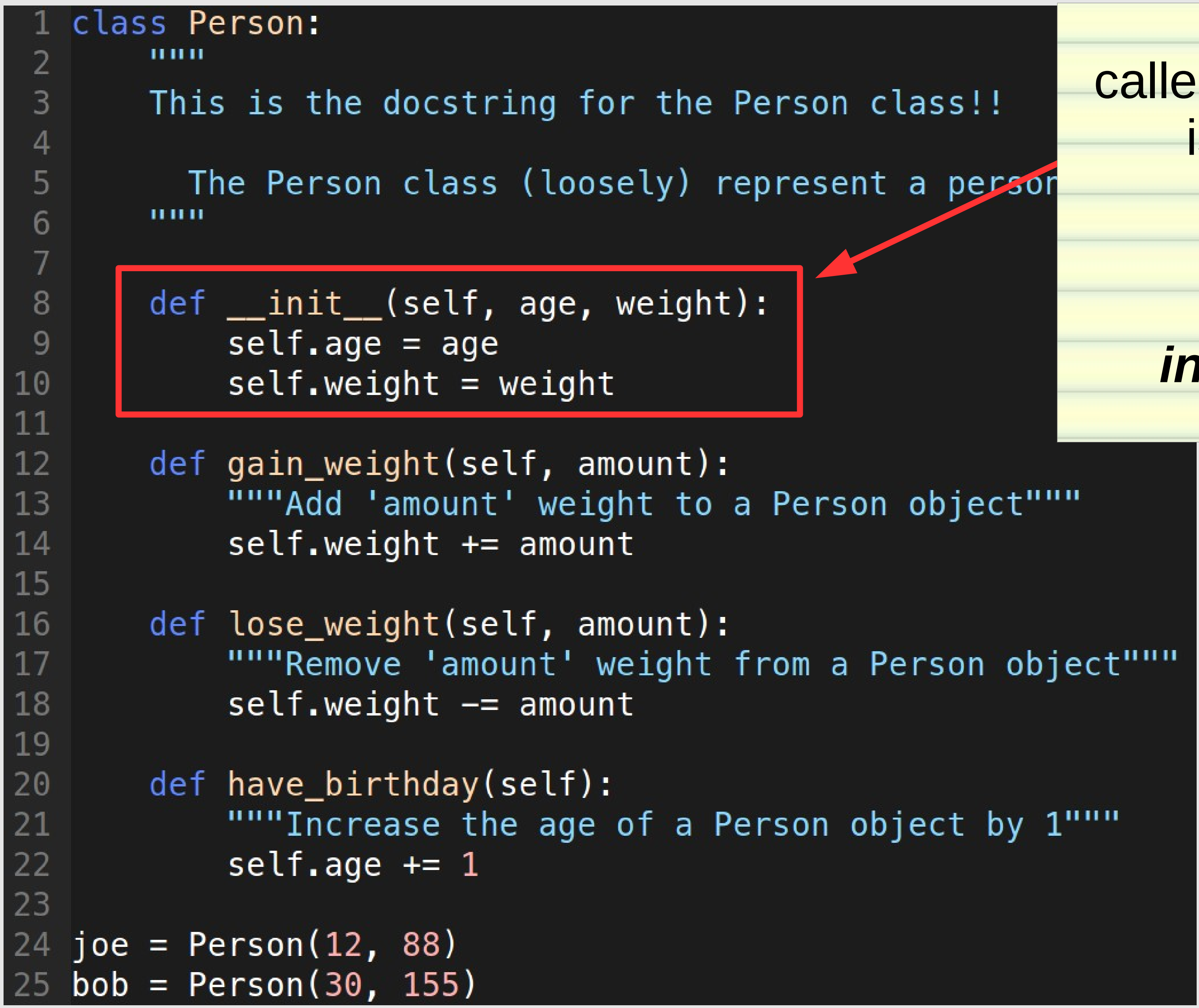
methods

Class Definition



```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

Class Definition



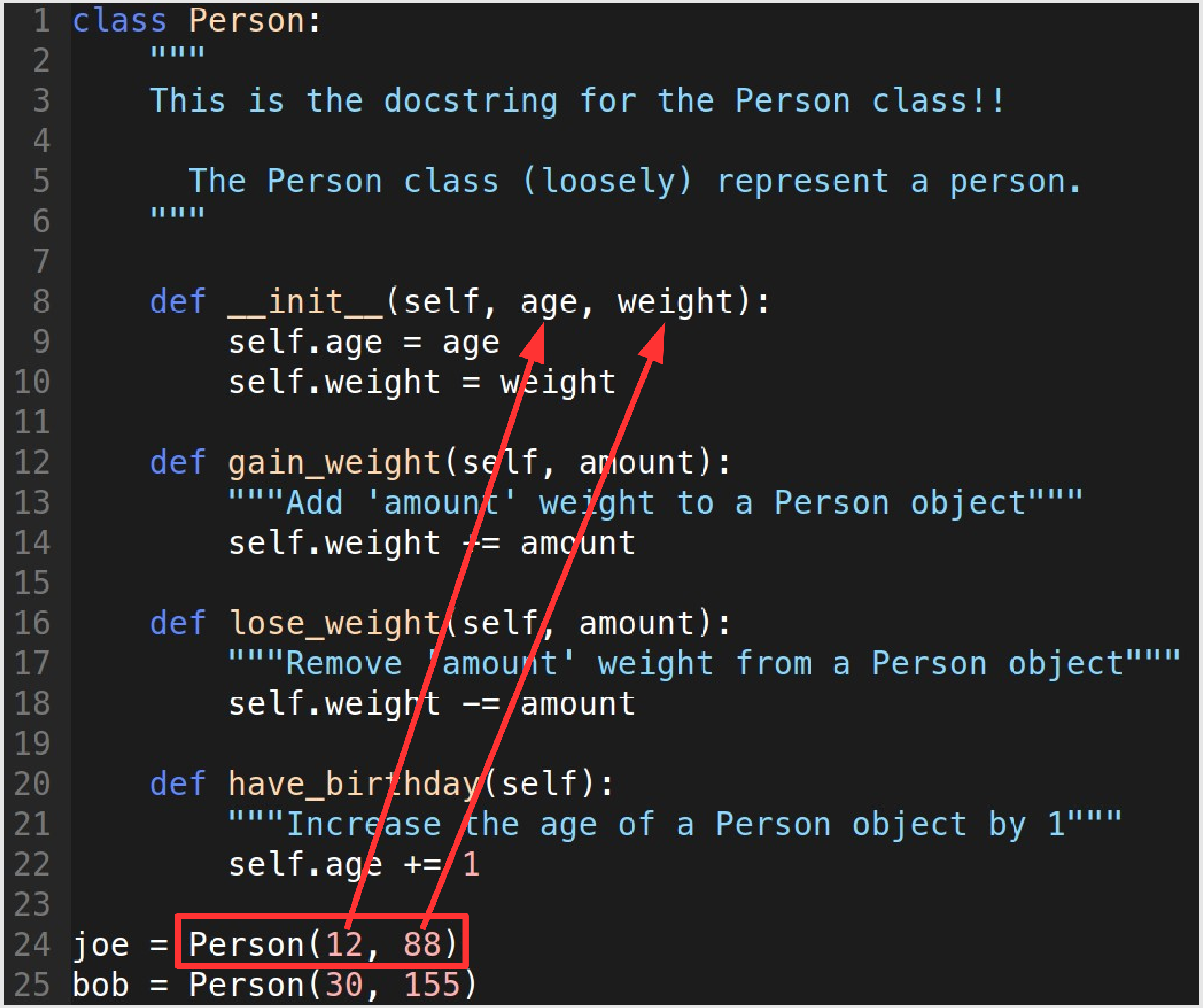
```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a perso
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

called when object is **created**

...or...

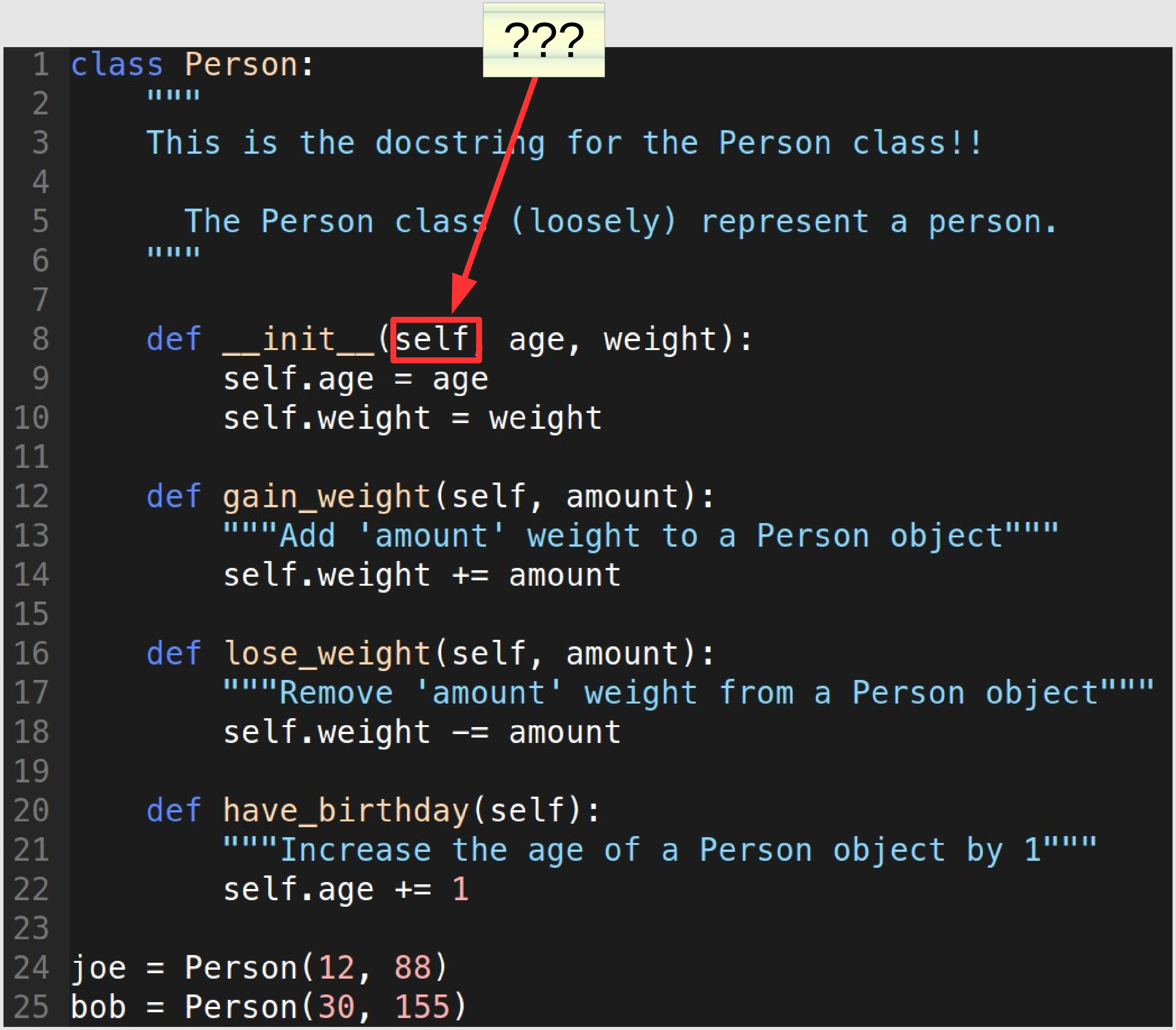
instantiated

Class Definition



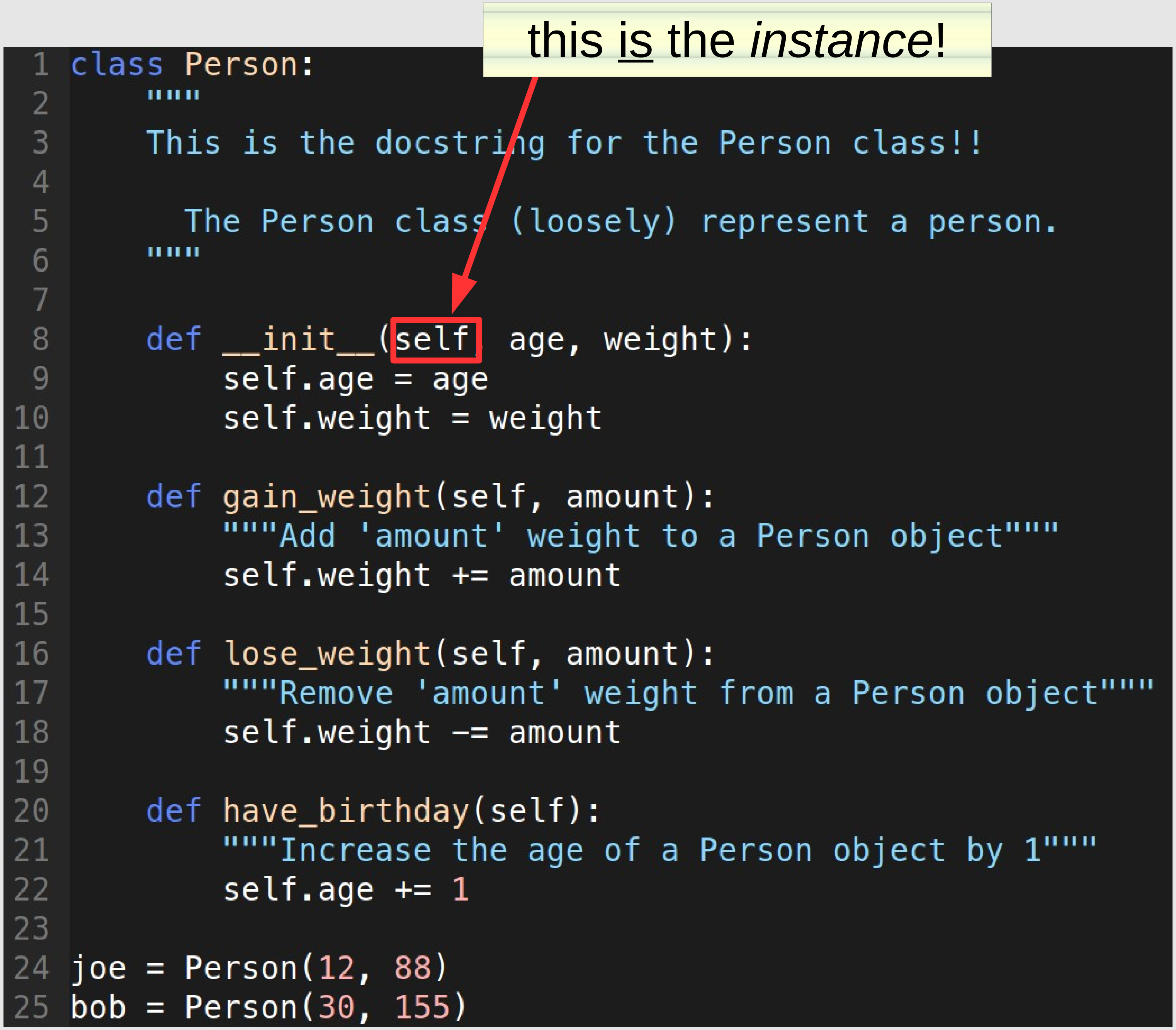
```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

Class Definition



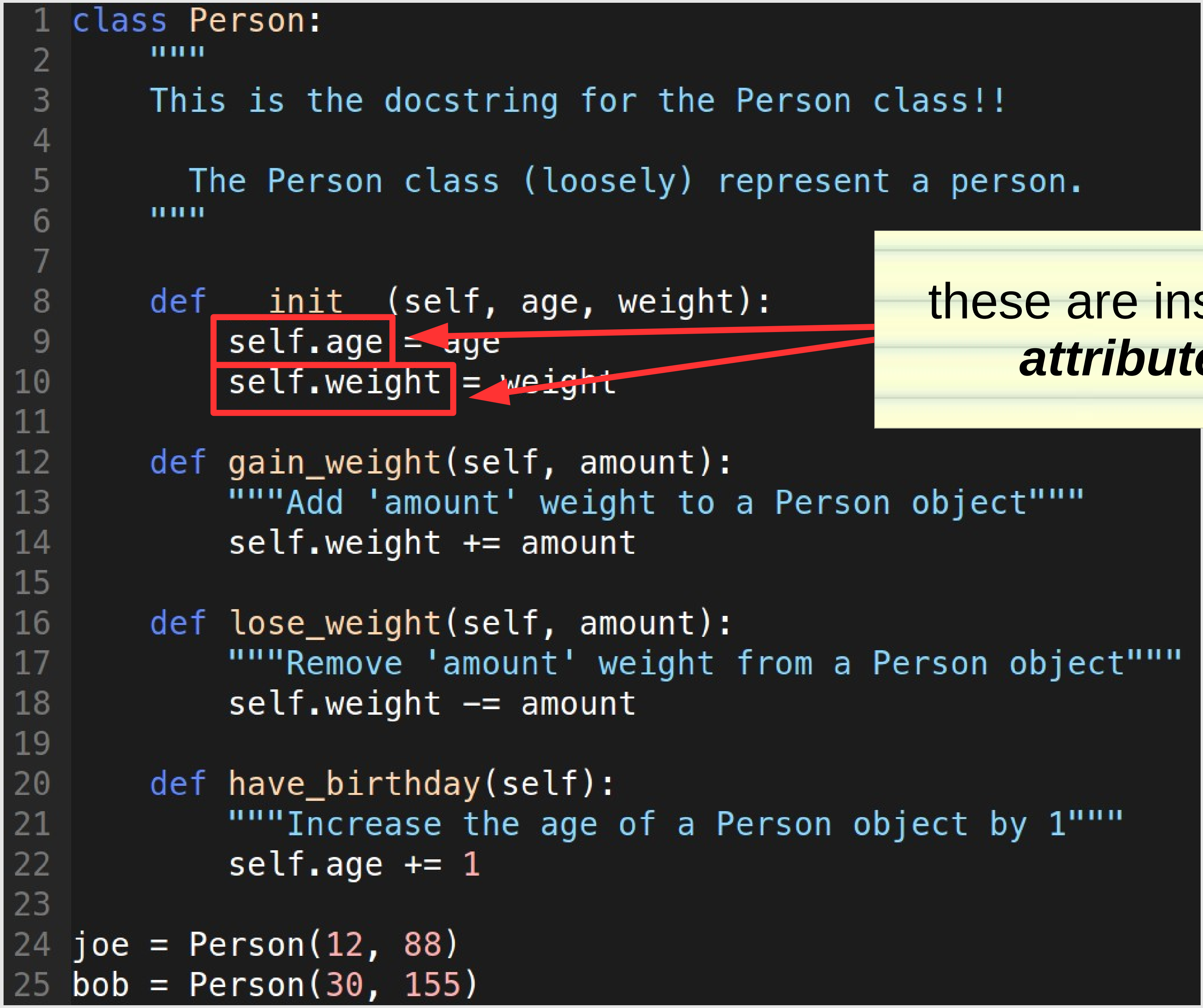
```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

Class Definition


this is the *instance*!

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

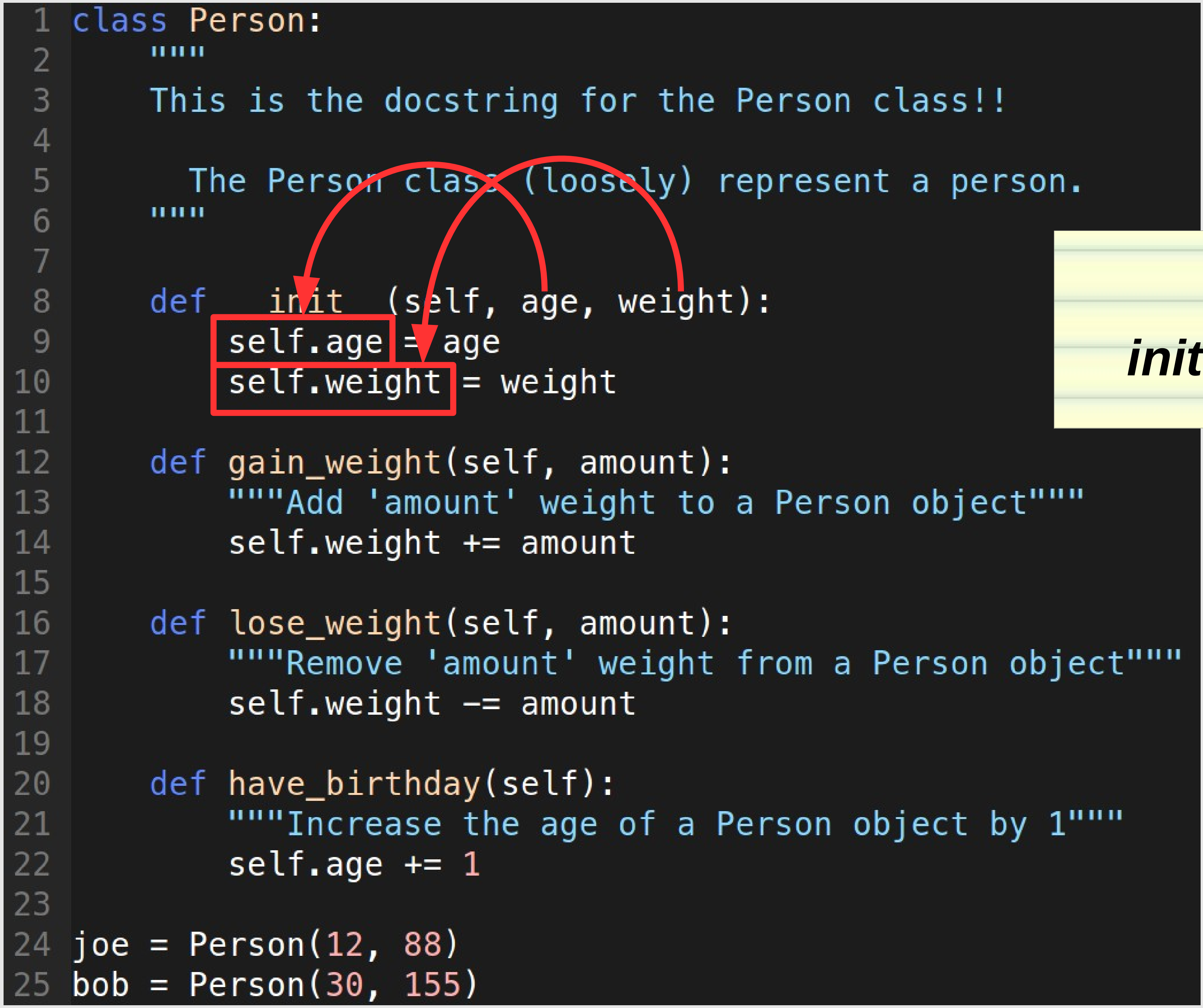
Instance/Object Attributes



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

these are instance ***attributes***

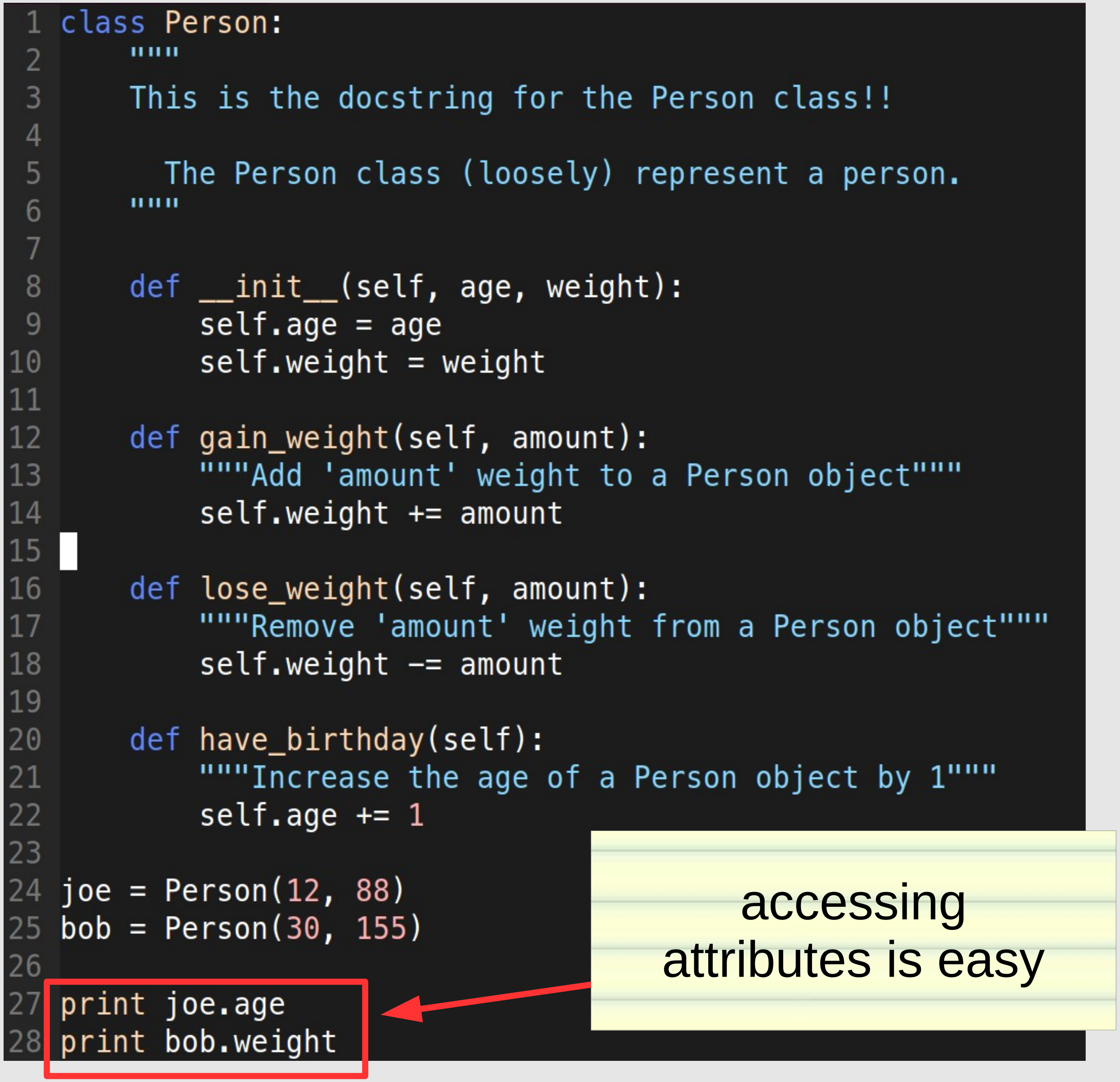
Initializing Objects



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

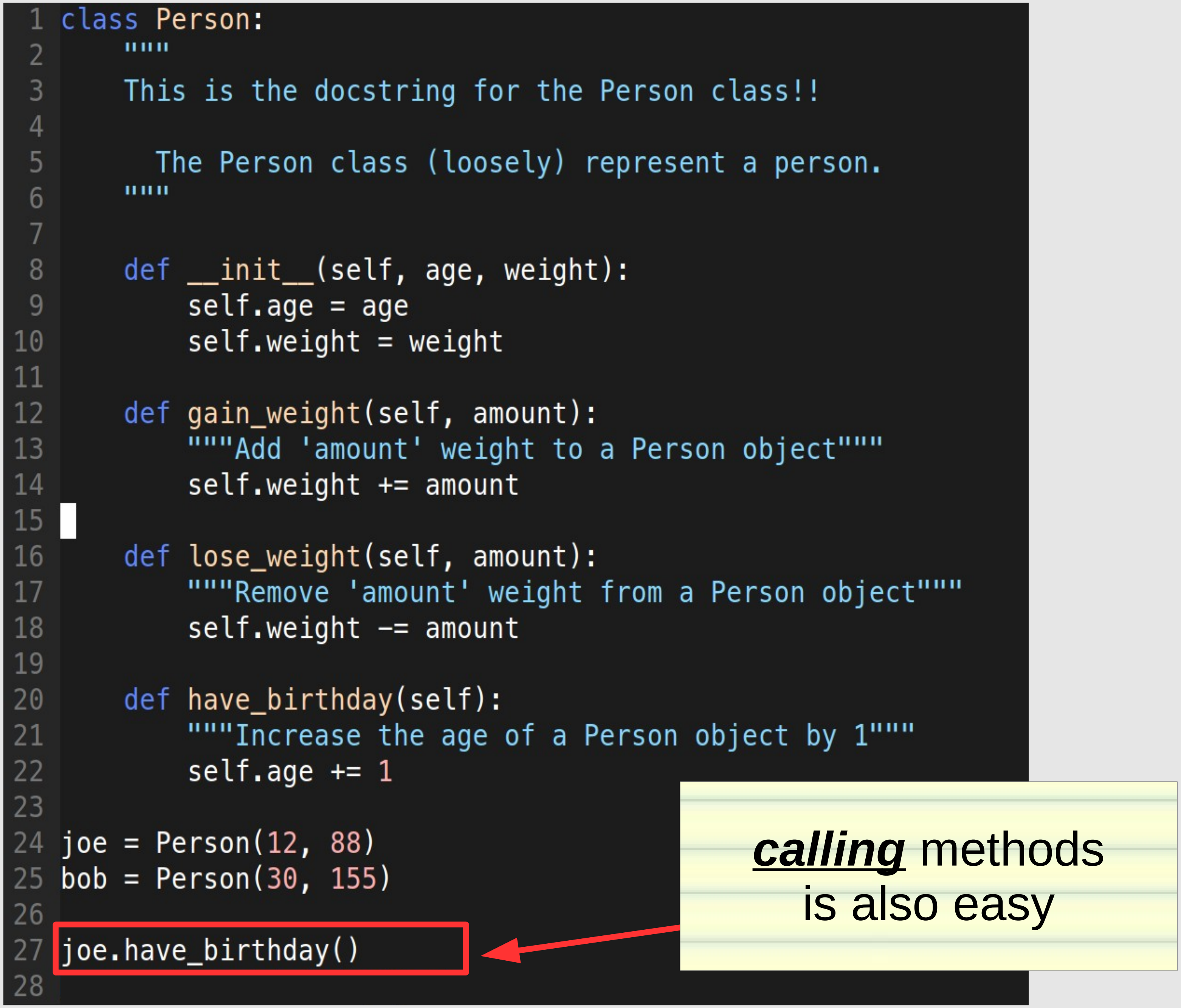
we are ***initializing*** them

Accessing Object Attributes



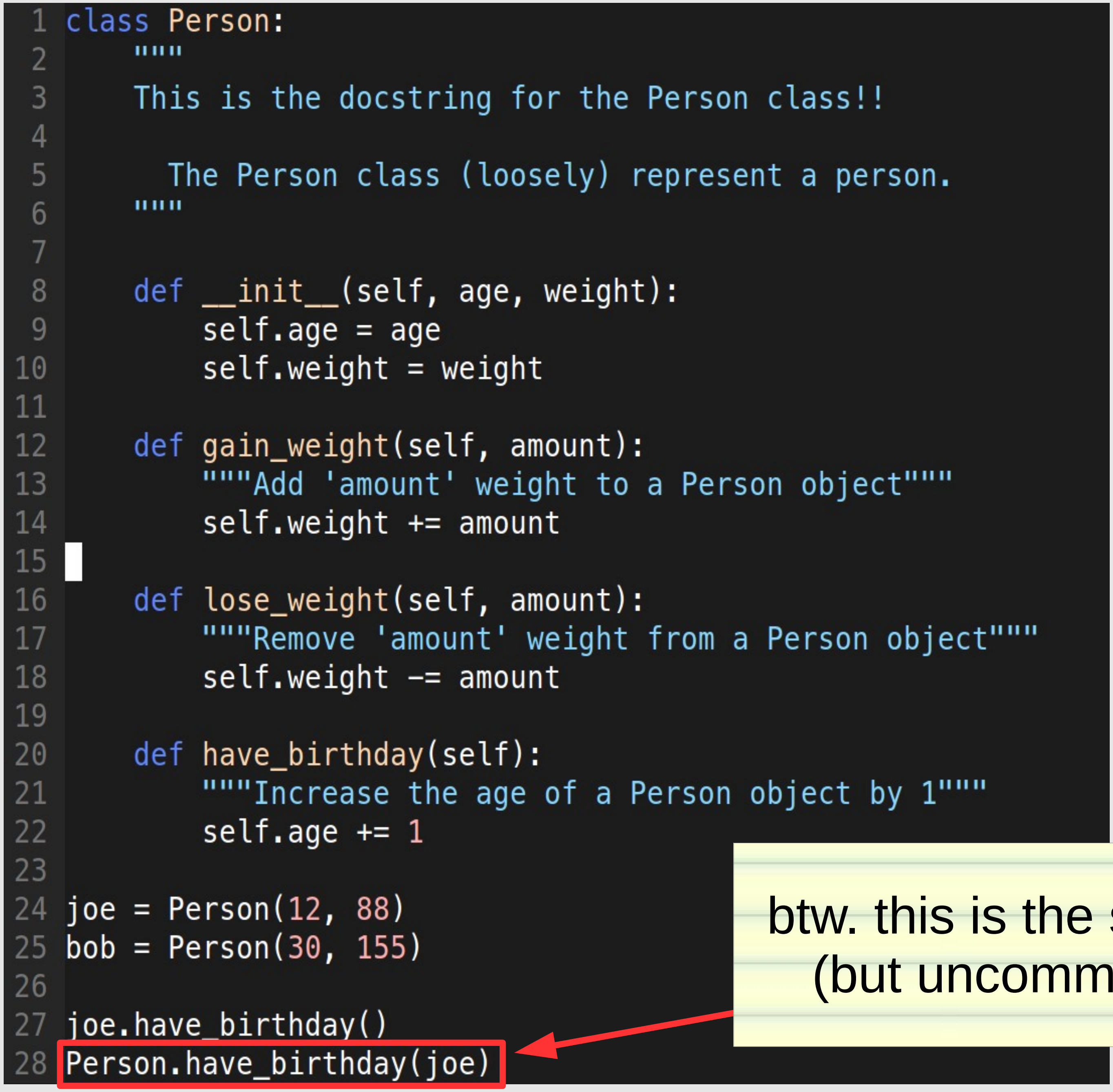
```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

print joe.age
print bob.weight
```

Calling Methods



```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()

```

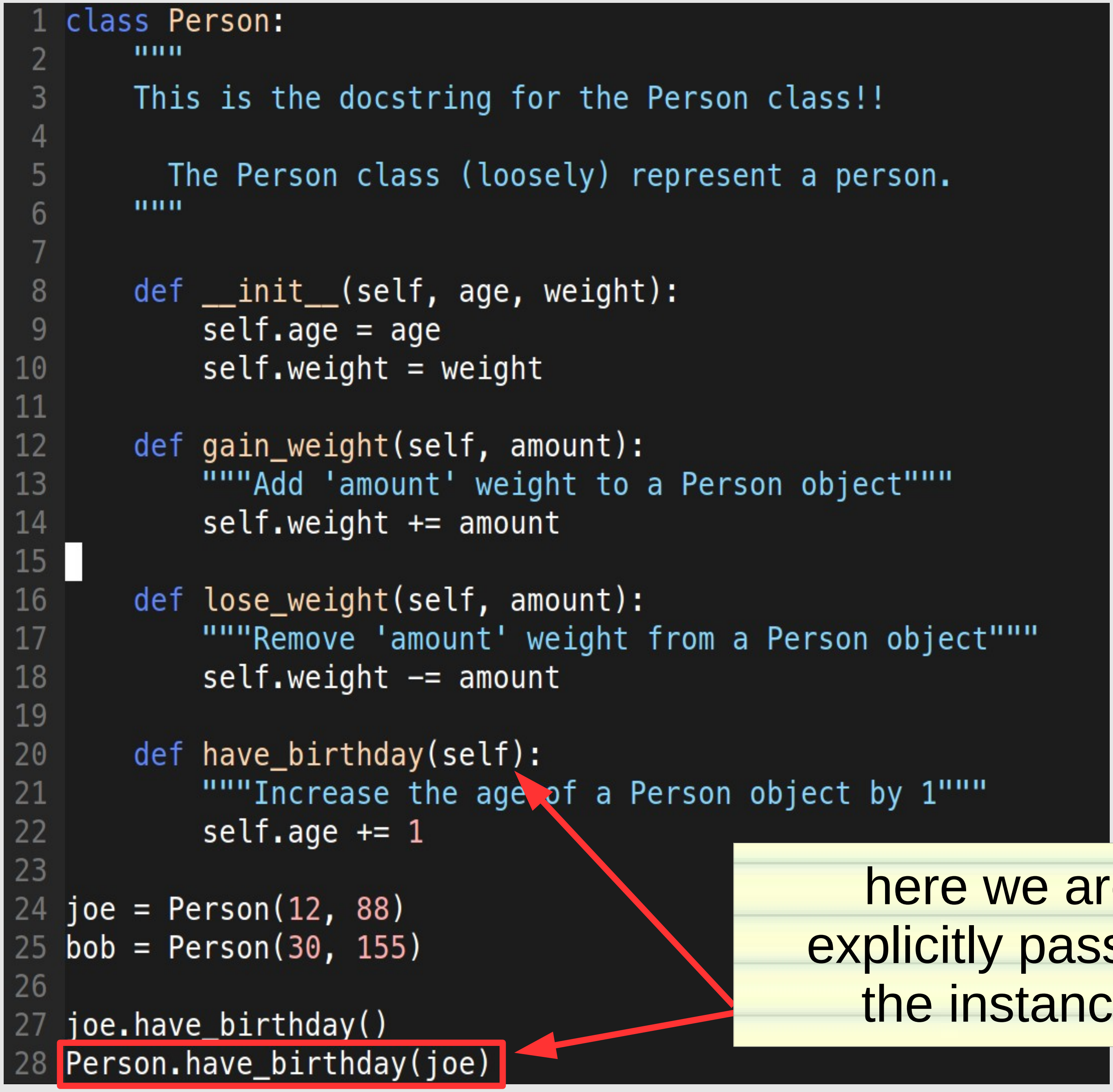
Calling Methods



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

btw. this is the same (but uncommon)

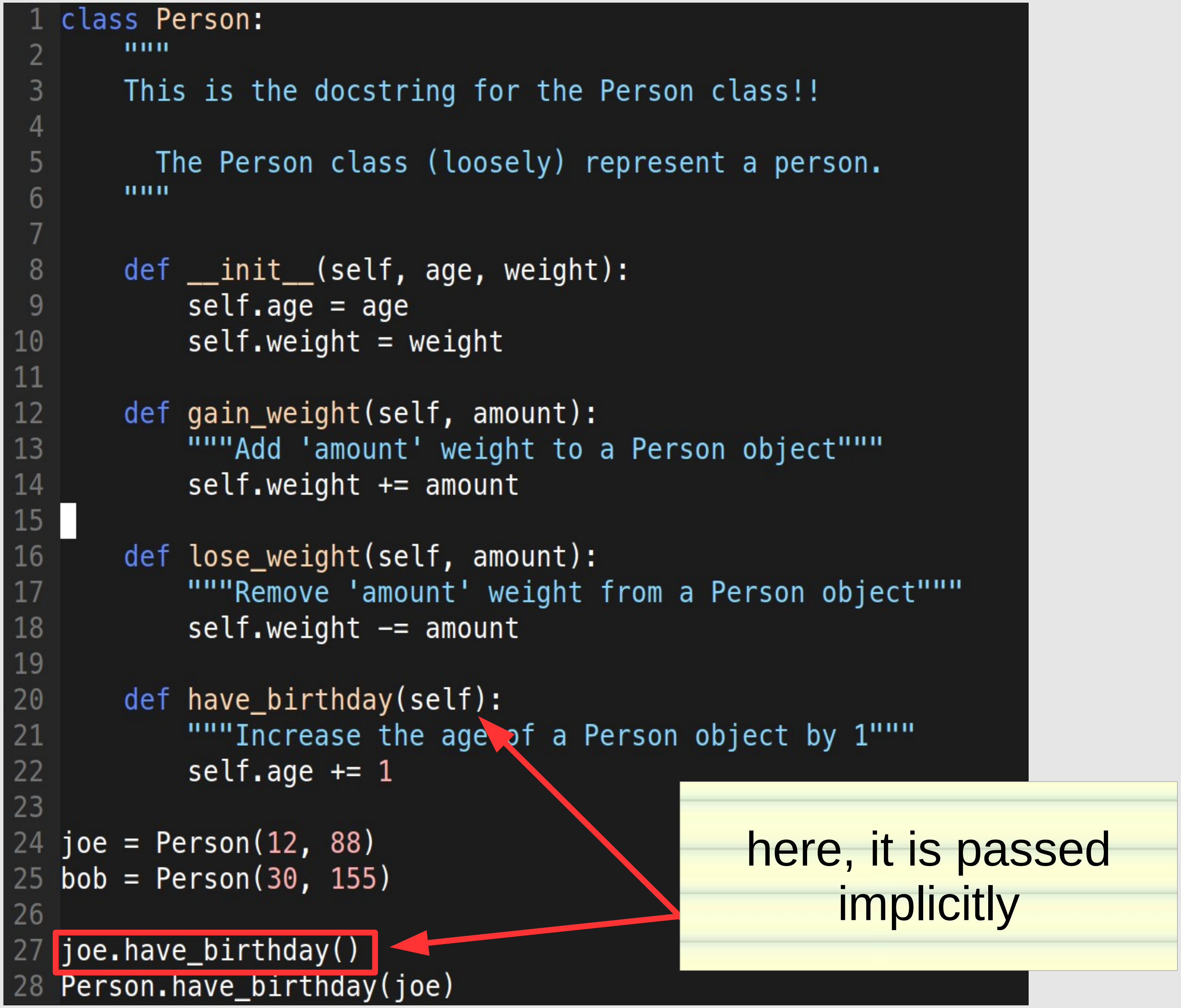
Calling Methods



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

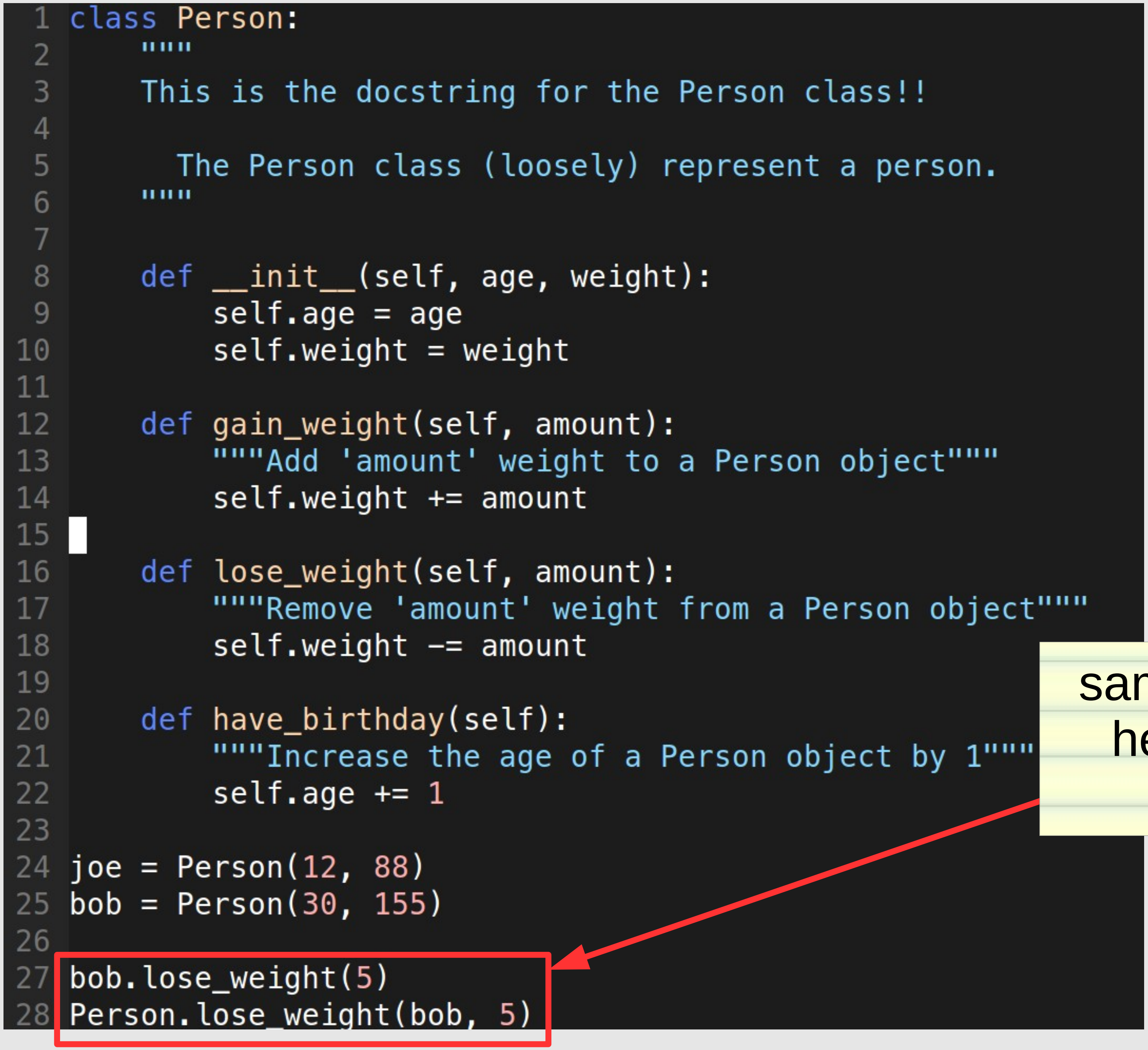
here we are explicitly passing the instance

Calling Methods



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

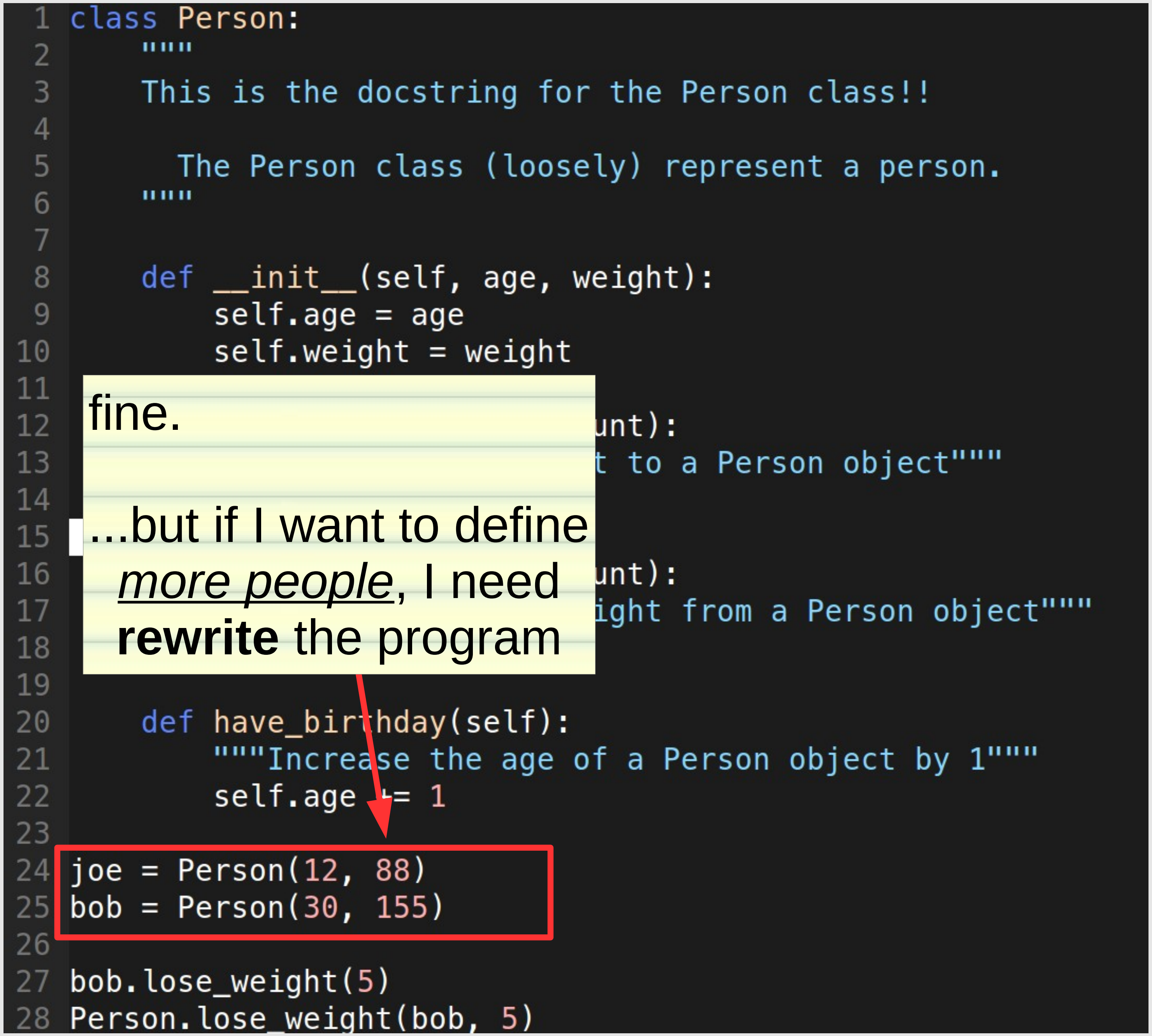
Calling Methods



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

bob.lose_weight(5)
Person.lose_weight(bob, 5)
```

same stuff going on here, but with an argument

Calling Methods

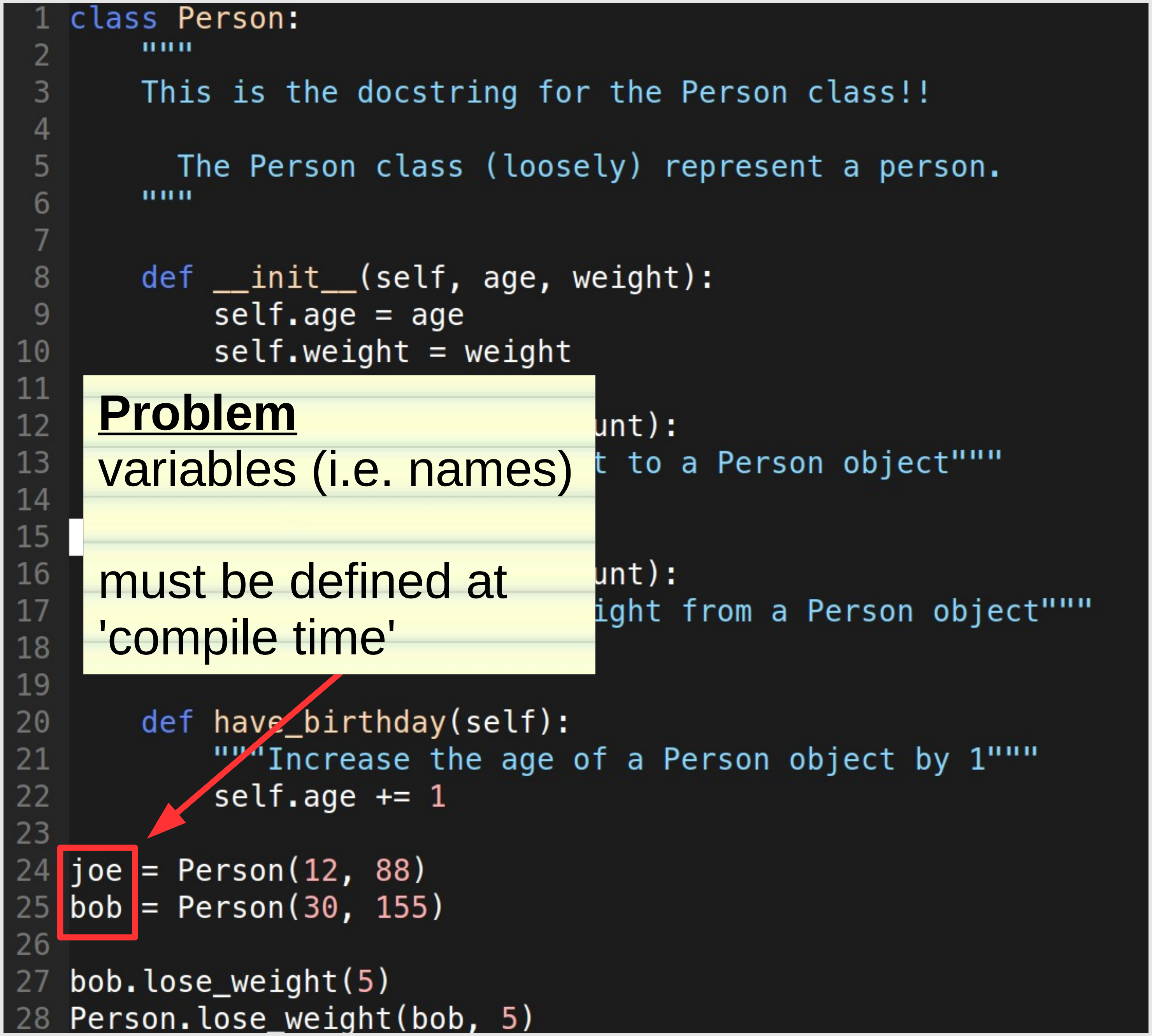


```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

                                 unt):
                                 t to a Person object"""


                                 unt):
                                 ight from a Person object"""


    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

bob.lose_weight(5)
Person.lose_weight(bob, 5)
```

fine.

...but if I want to define *more people*, I need **rewrite** the program

Calling Methods



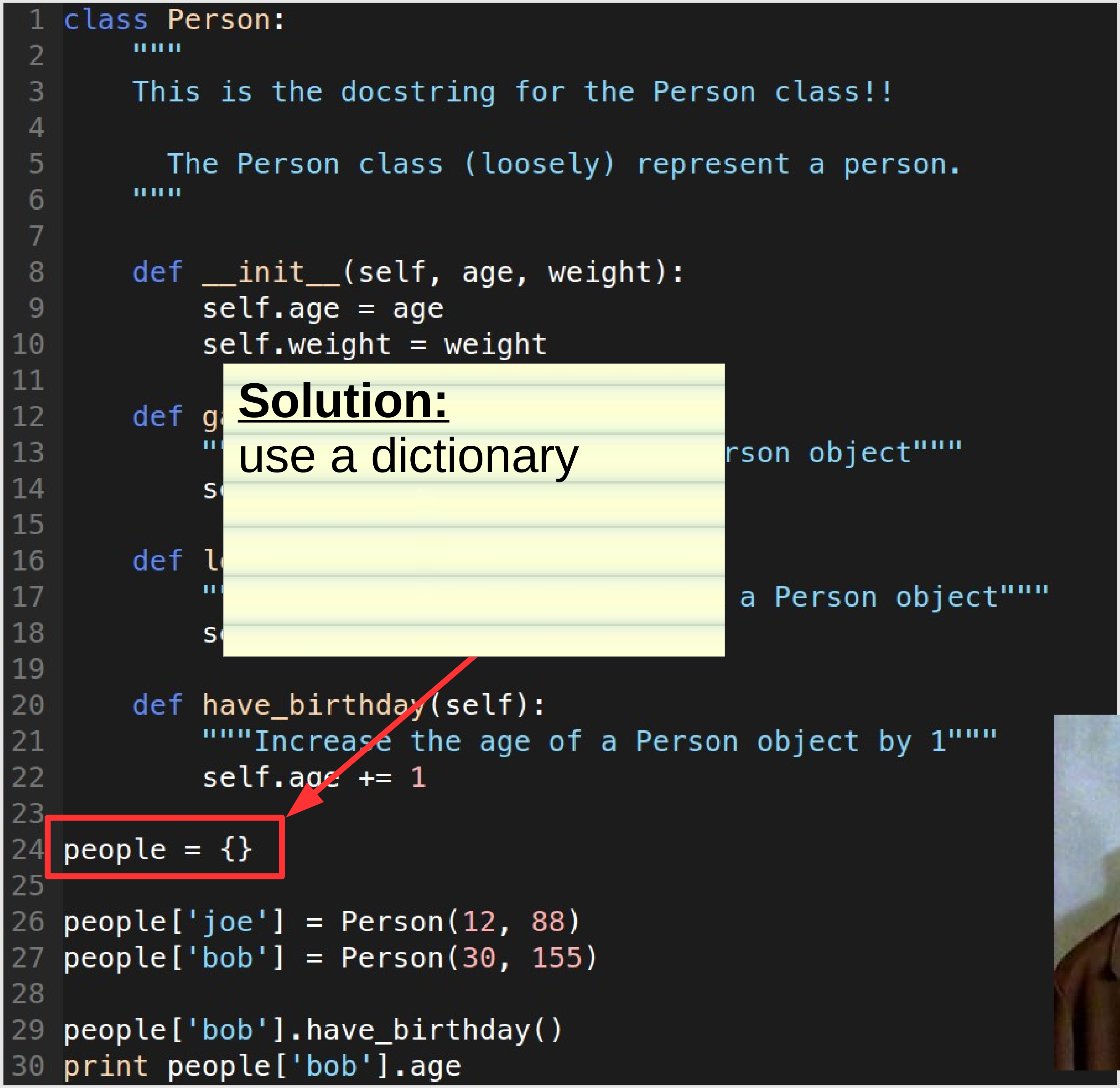
```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

                                              unt):
                                              t to a Person object"""


                                              unt):
                                              ight from a Person object"""


    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

bob.lose_weight(5)
Person.lose_weight(bob, 5)
```

Problem

variables (i.e. names)

must be defined at 'compile time'

Using Dictionaries of Instances

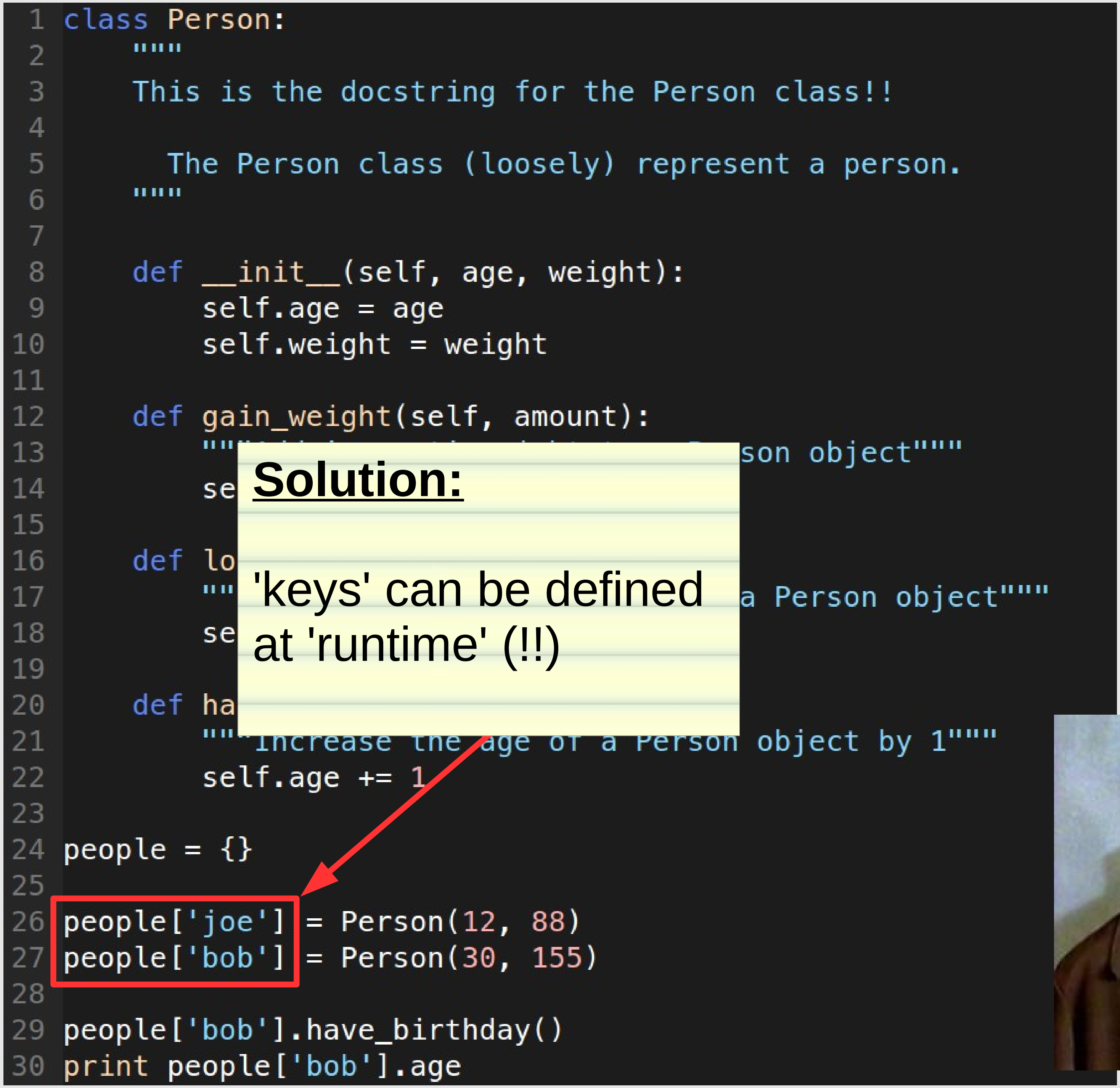


```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def g
        "                                  rson object"""
        s

    def l
        "                                    a Person object"""
        s

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

people = {}

people['joe'] = Person(12, 88)
people['bob'] = Person(30, 155)

people['bob'].have_birthday()
print people['bob'].age
```

Solution:

use a dictionary

Using Dictionaries of Instances

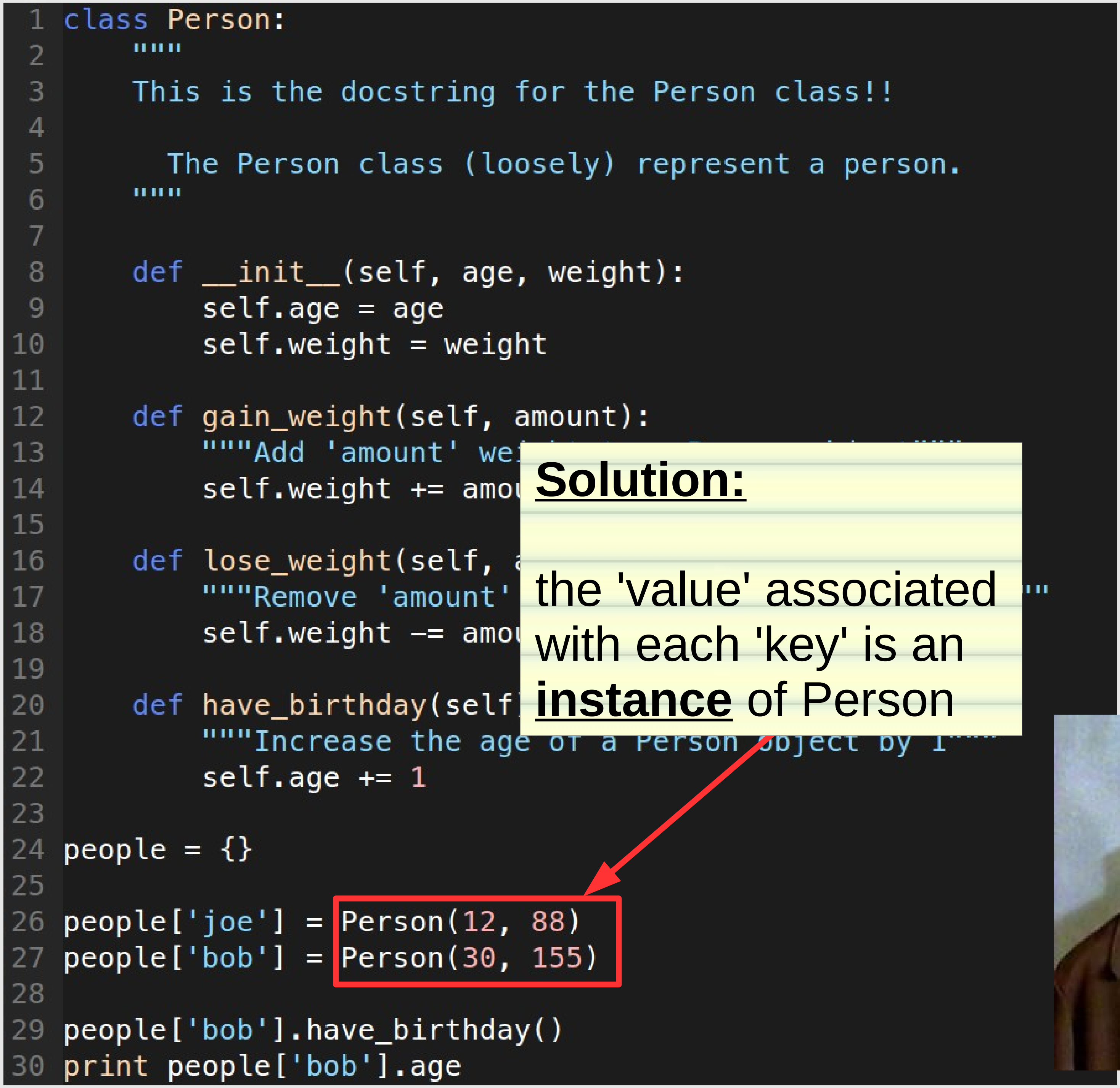


```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """                          son object"""
        se

    def lo
        ""                           a Person object"""
        se

    def ha
        """Increase the age of a Person object by 1"""
        self.age += 1

people = {}

people['joe'] = Person(12, 88)
people['bob'] = Person(30, 155)

people['bob'].have_birthday()
print people['bob'].age
```

Solution:

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

Using Dictionaries of Instances

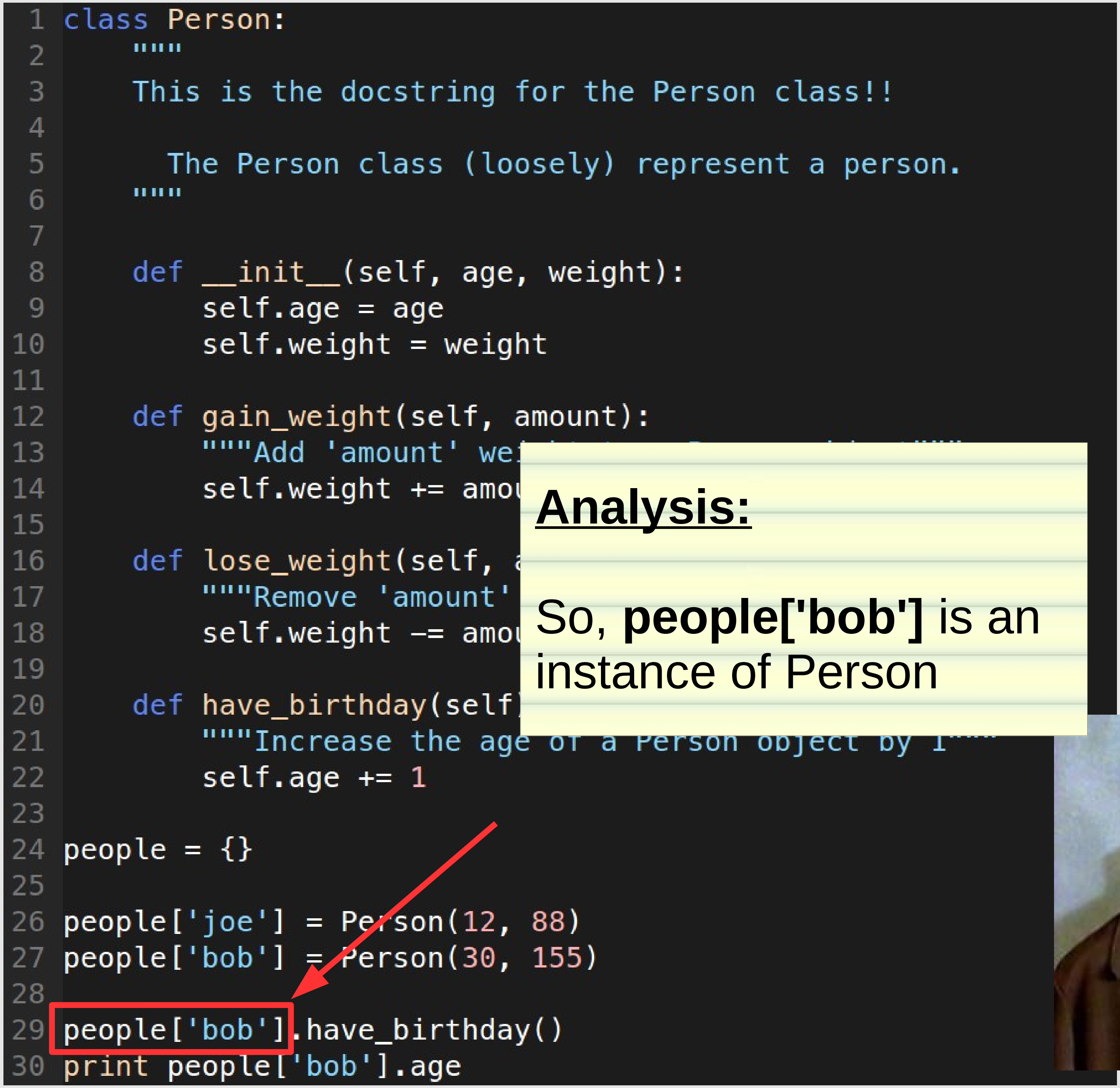


```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' wei
        self.weight += amou

    def lose_weight(self, a
        """Remove 'amount'                          ""
        self.weight -= amou

    def have_birthday(self
        """Increase the age of a Person object by 1"""
        self.age += 1

people = {}

people['joe'] = Person(12, 88)
people['bob'] = Person(30, 155)

people['bob'].have_birthday()
print people['bob'].age
```

Solution:

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

Using Dictionaries of Instances

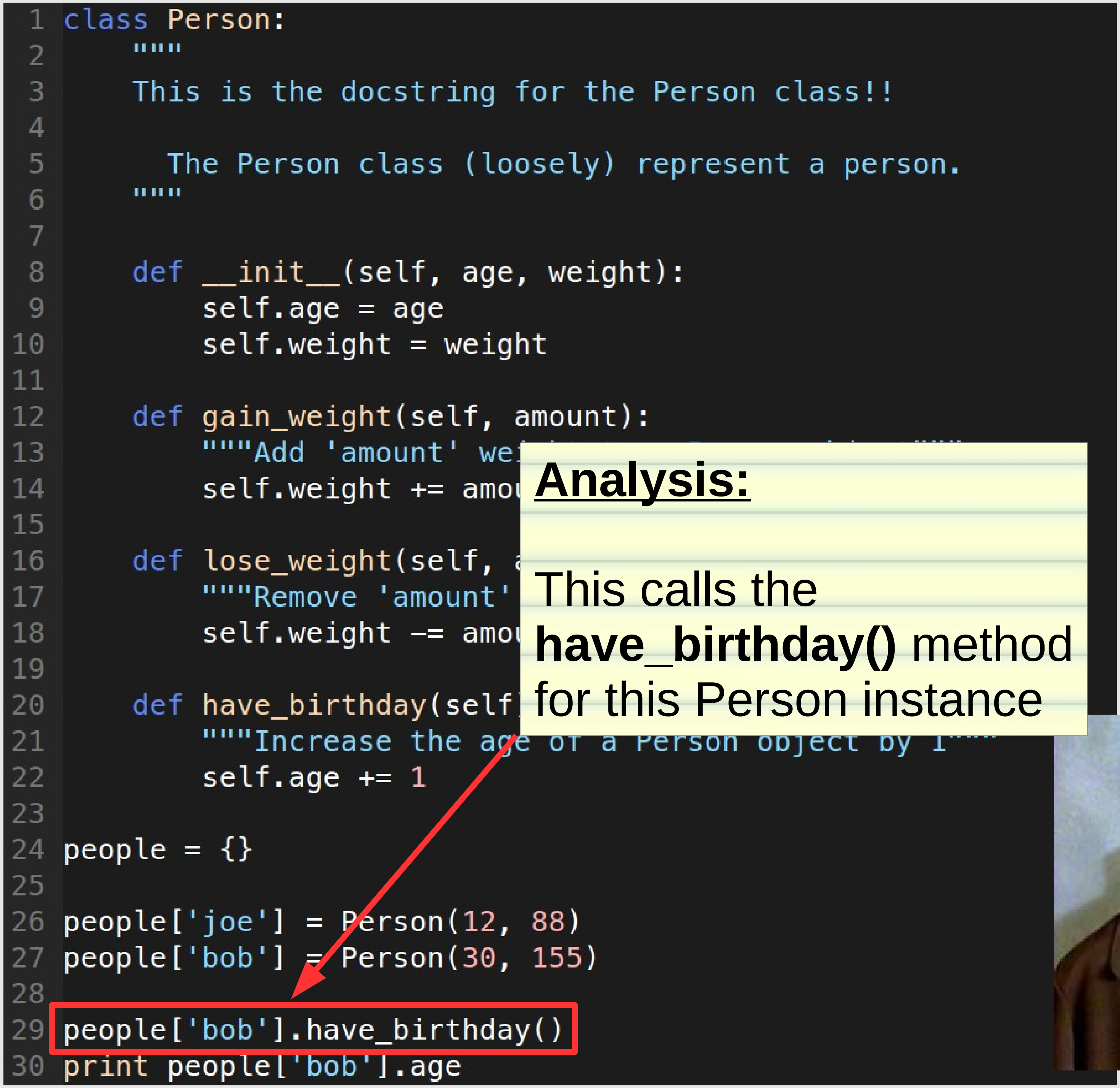


```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' wei
        self.weight += amou

    def lose_weight(self, a
        """Remove 'amount'
        self.weight -= amou

    def have_birthday(self)
        """Increase the age of a Person object by 1"""
        self.age += 1

people = {}

people['joe'] = Person(12, 88)
people['bob'] = Person(30, 155)

people['bob'].have_birthday()
print people['bob'].age
```

Analysis:

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

Using Dictionaries of Instances



```
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' wei
        self.weight += amou

    def lose_weight(self, a
        """Remove 'amount'
        self.weight -= amou

    def have_birthday(self)
        """Increase the age of a Person object by 1"""
        self.age += 1

people = {}

people['joe'] = Person(12, 88)
people['bob'] = Person(30, 155)

people['bob'].have_birthday()
print people['bob'].age
```

Analysis:

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

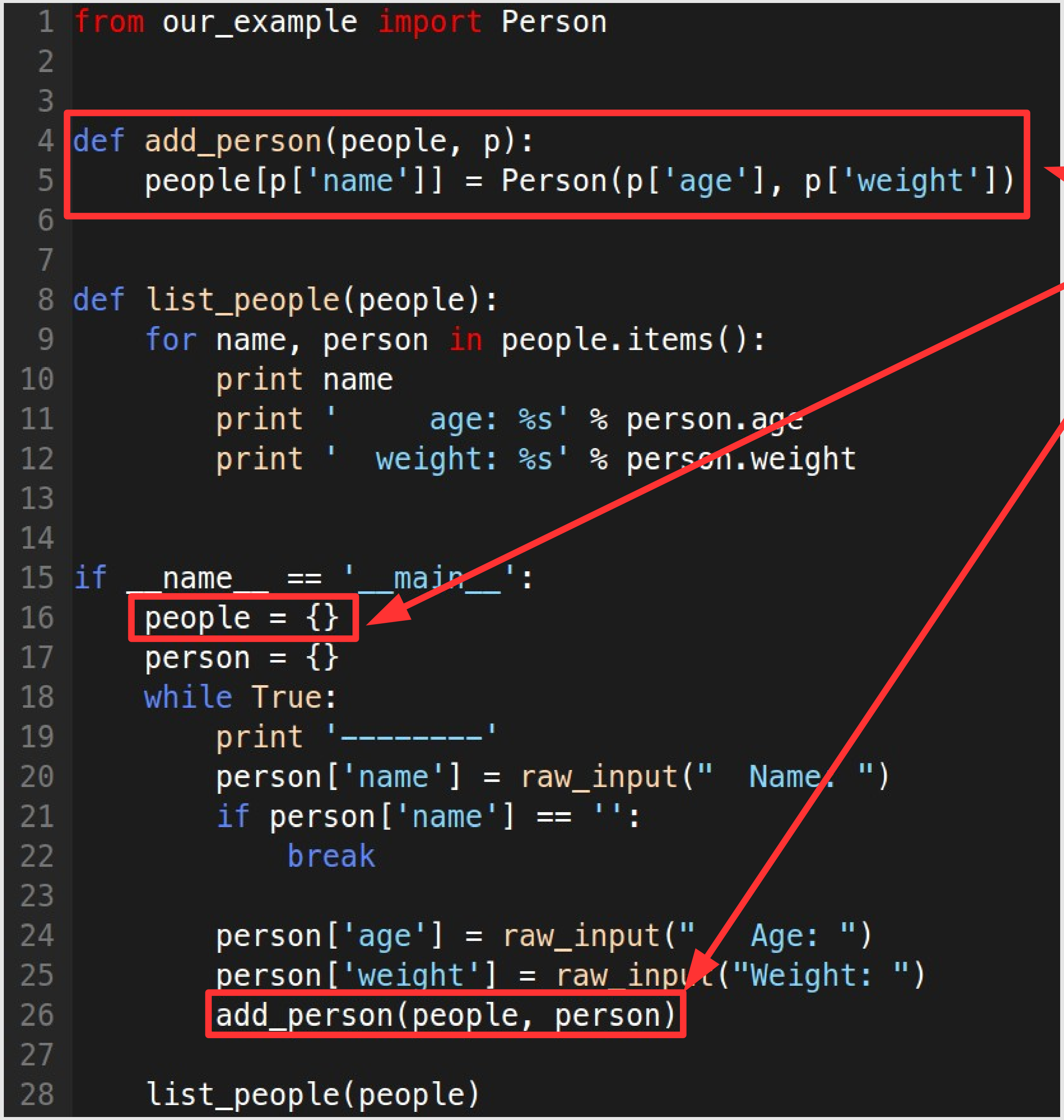
Next Step

<u>**Now we just need a function that can:**</u>

1) create a Person instance

2) add the instance to a dictionary

Building Dictionaries of Instances from User Input

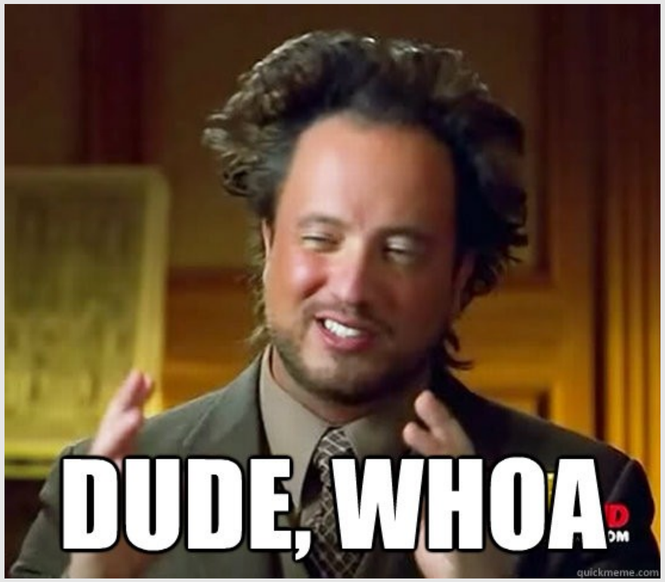


```
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

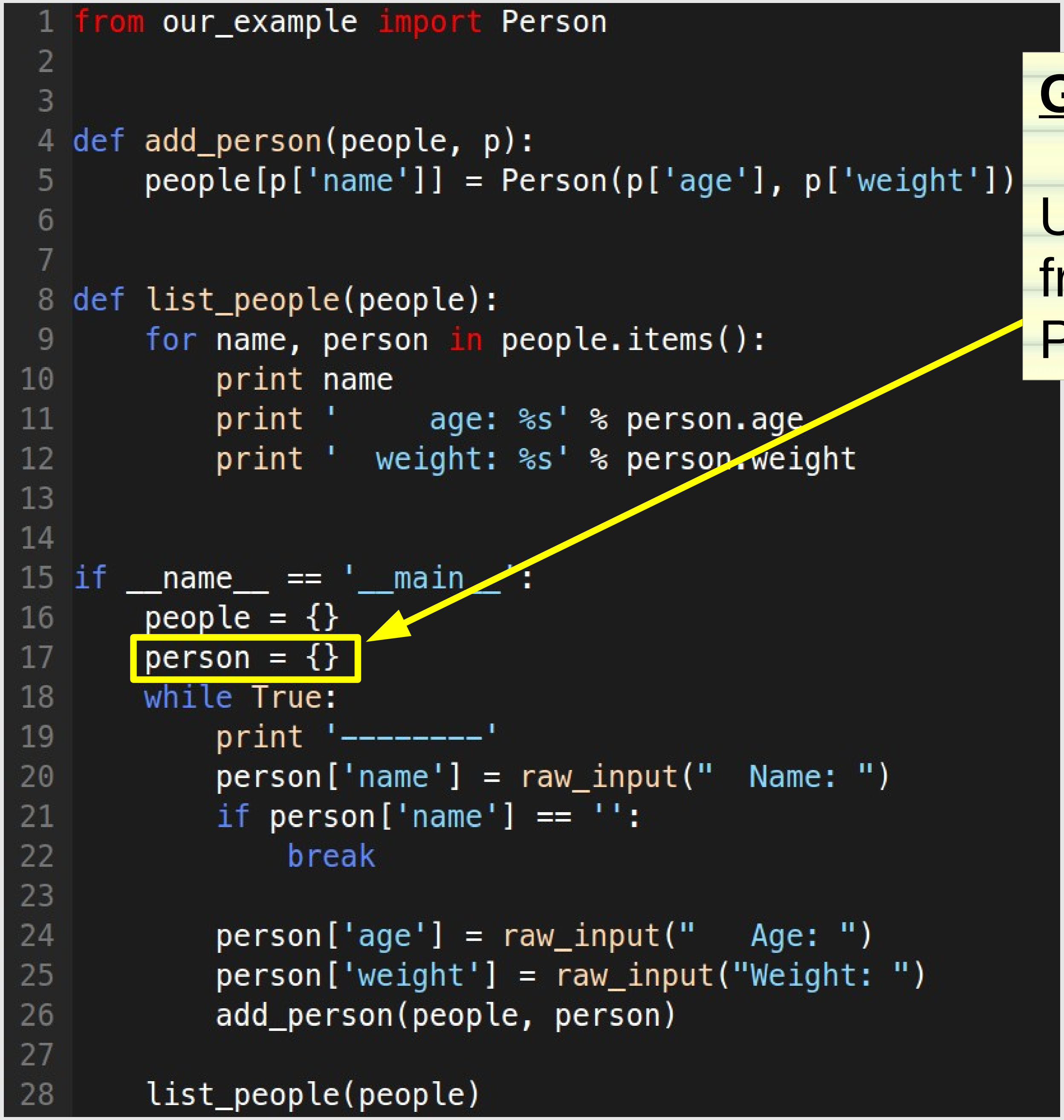
Solution:

This implements both requirements

what's all that other stuff?



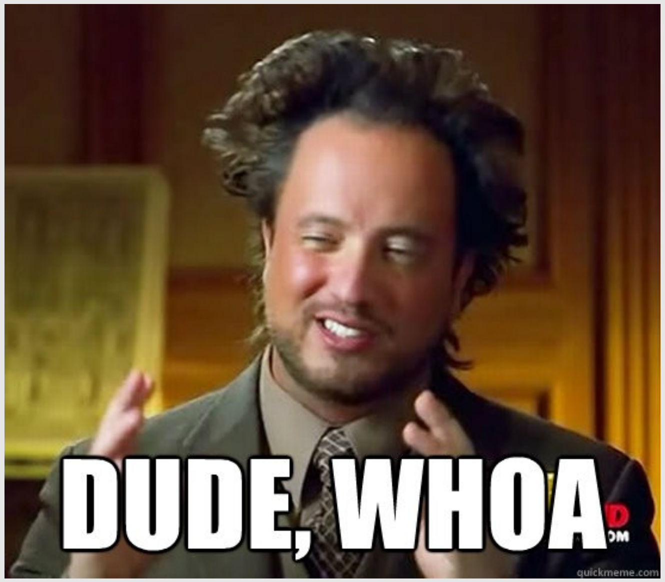
Building Dictionaries of Instances from User Input



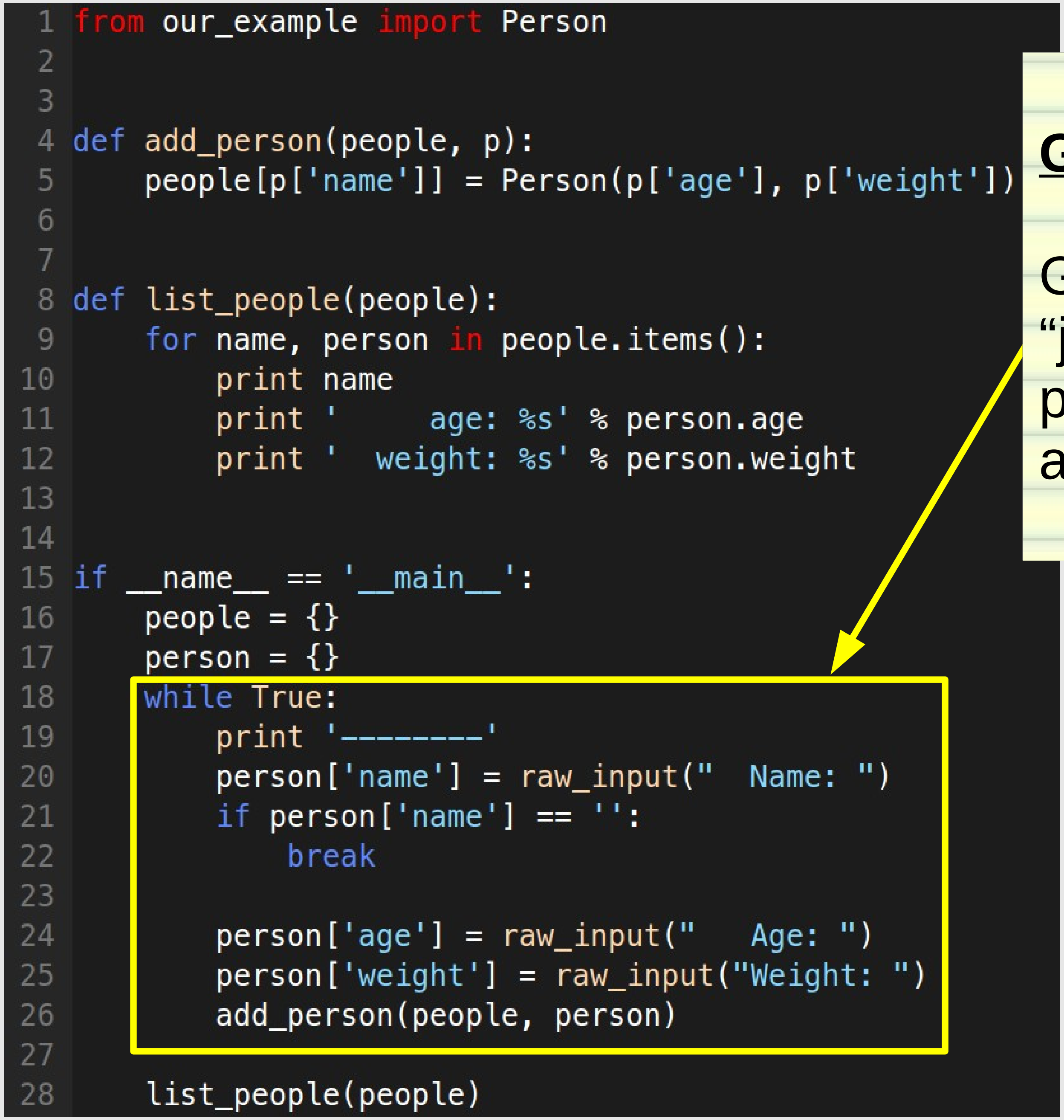
```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

<u>Getting User Input:</u>

Used to store input from user about a Person



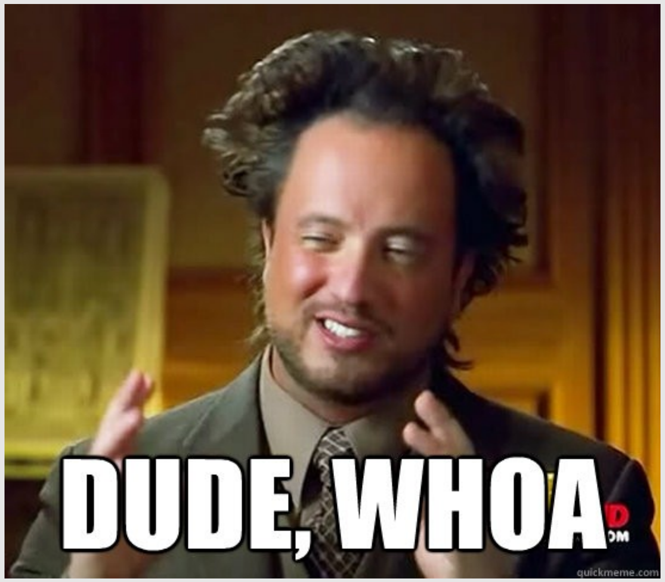
Building Dictionaries of Instances from User Input



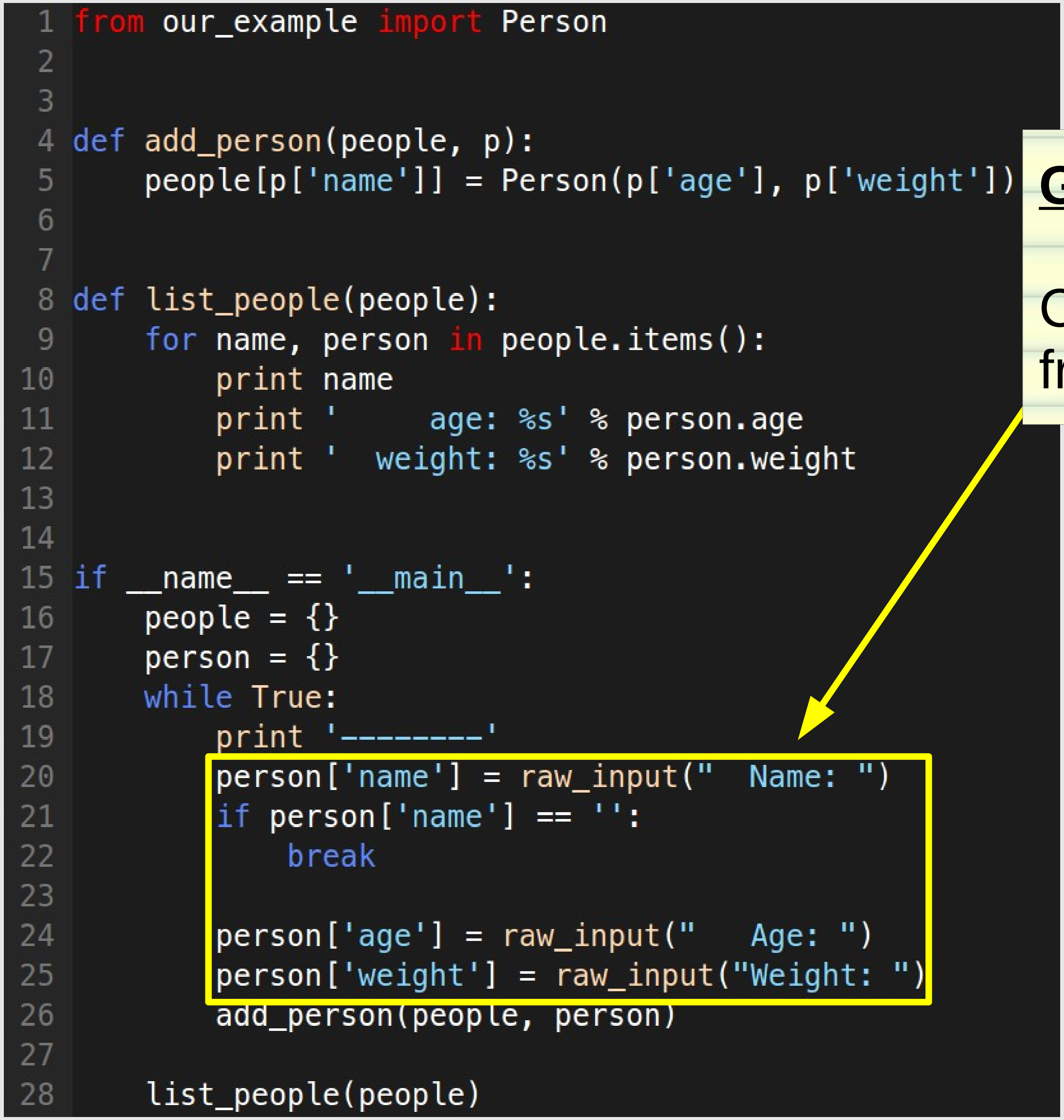
```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

Getting User Input:

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



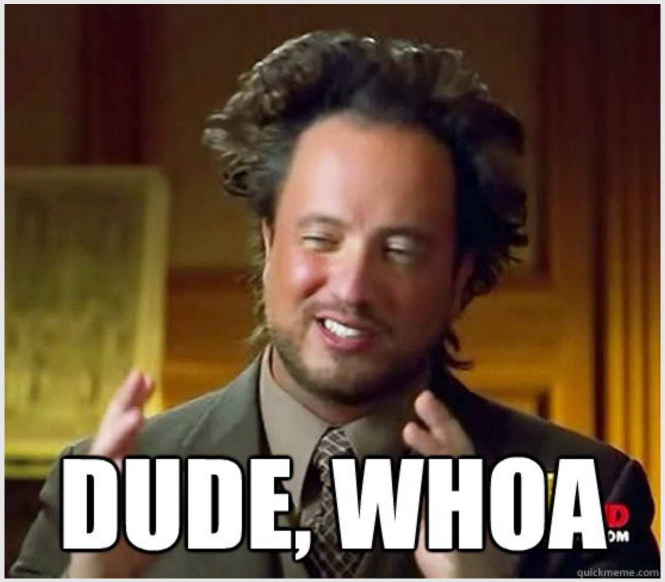
Building Dictionaries of Instances from User Input



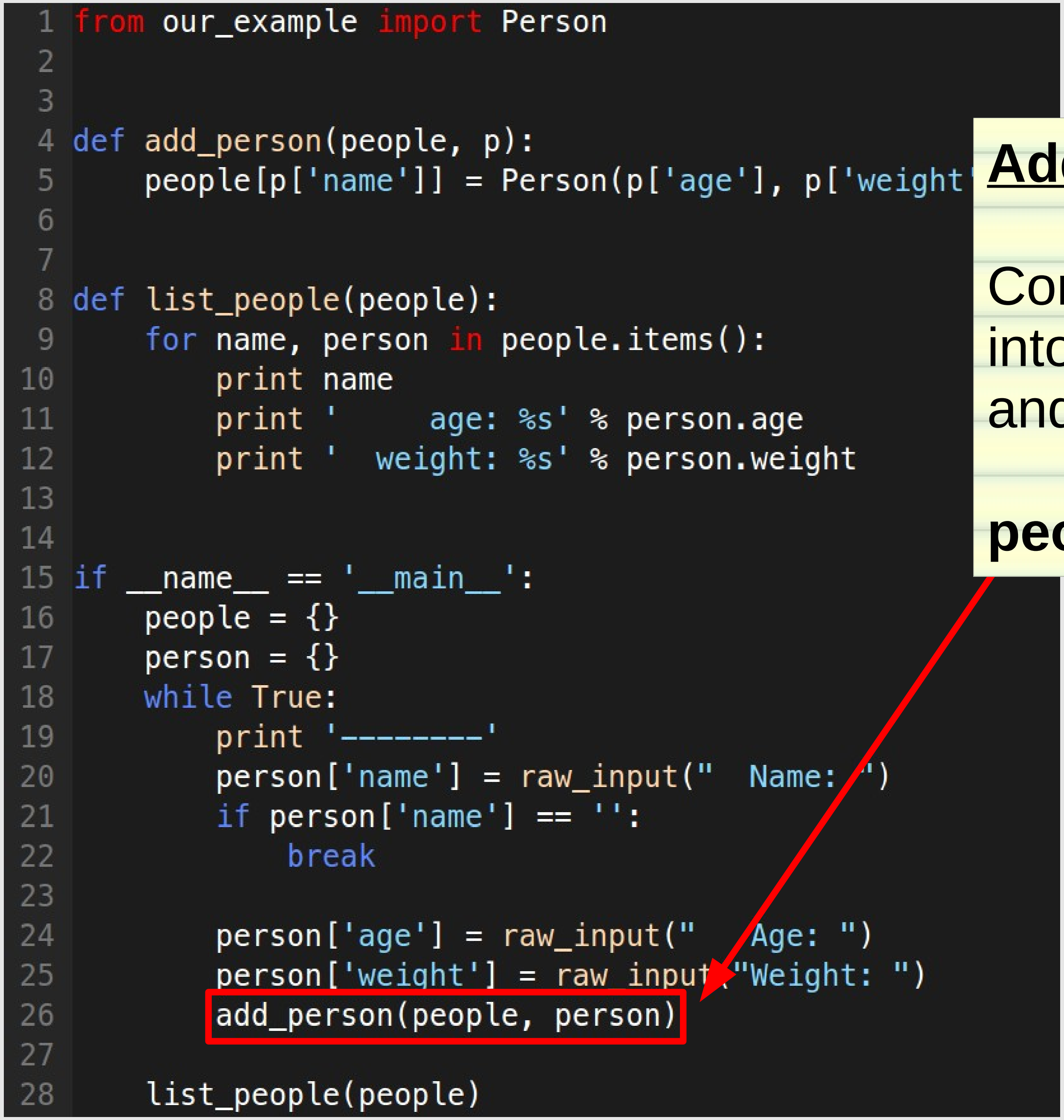
```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

Getting User Input:

Collect person attributes from user via keyboard



Building Dictionaries of Instances from User Input

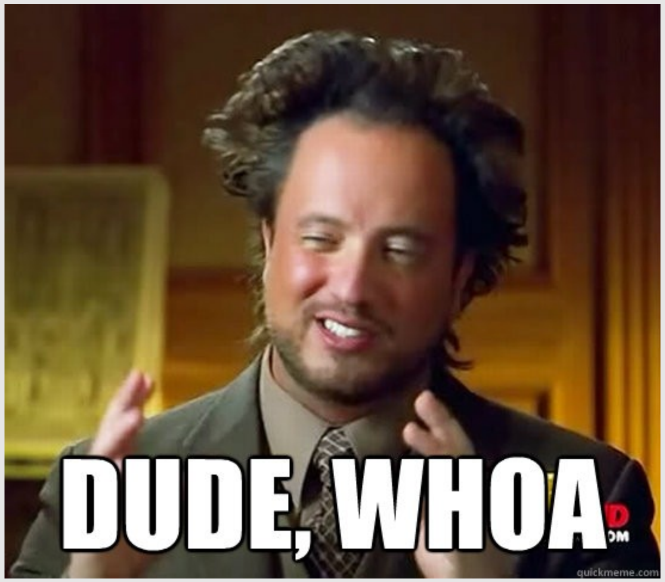


```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

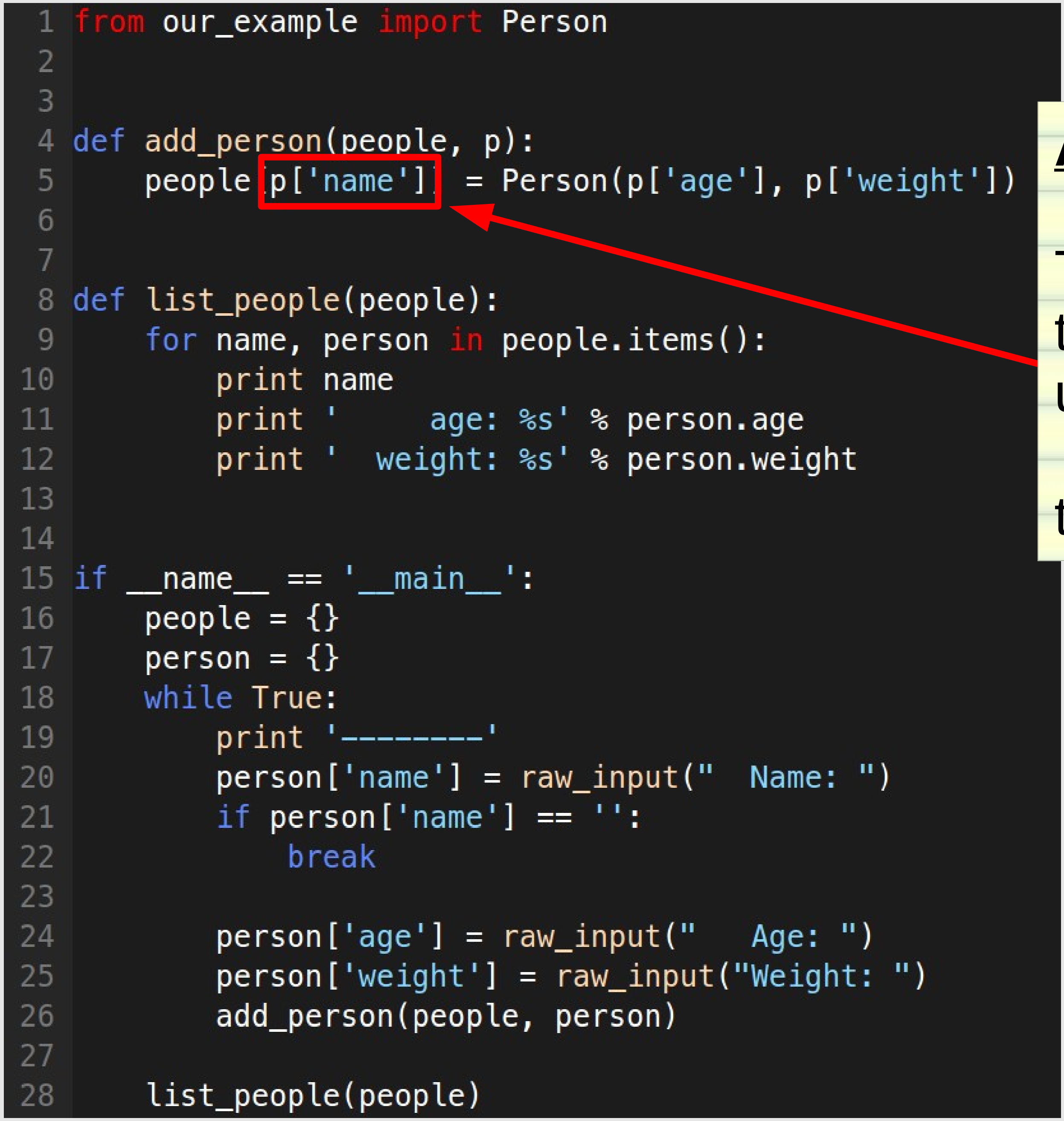
<u>Adding the Person:</u>

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

people dictionary !



Building Dictionaries of Instances from User Input

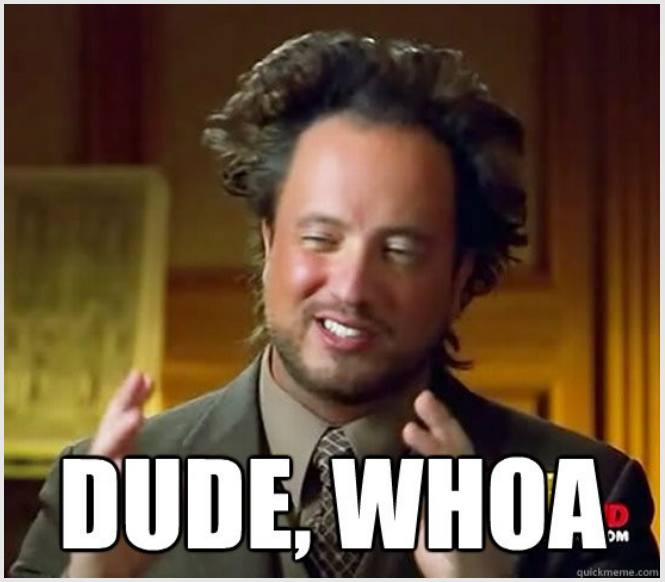


```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

Adding the Person:

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

this string will be a **key**



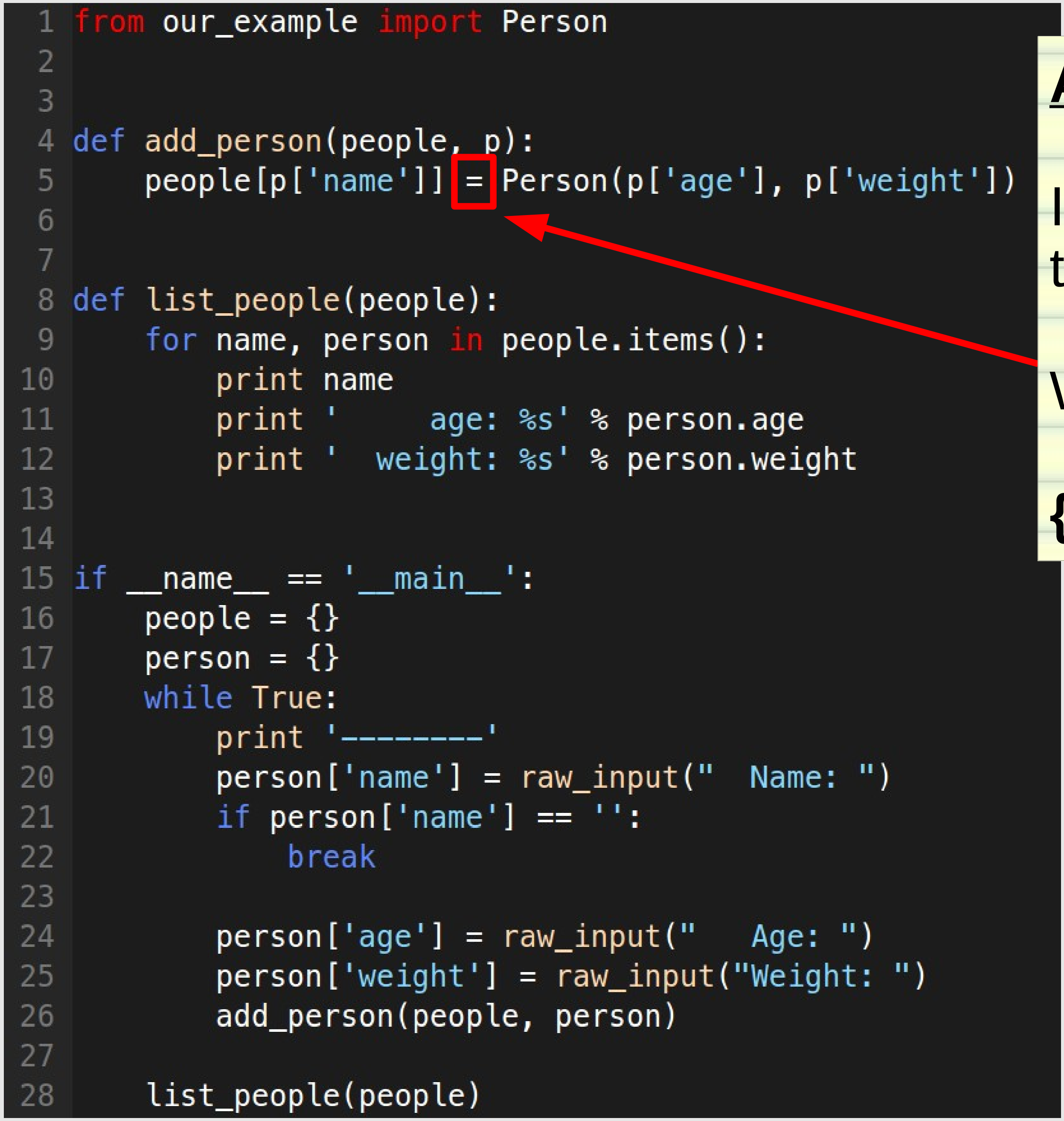
Building Dictionaries of Instances from User Input

```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    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



```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

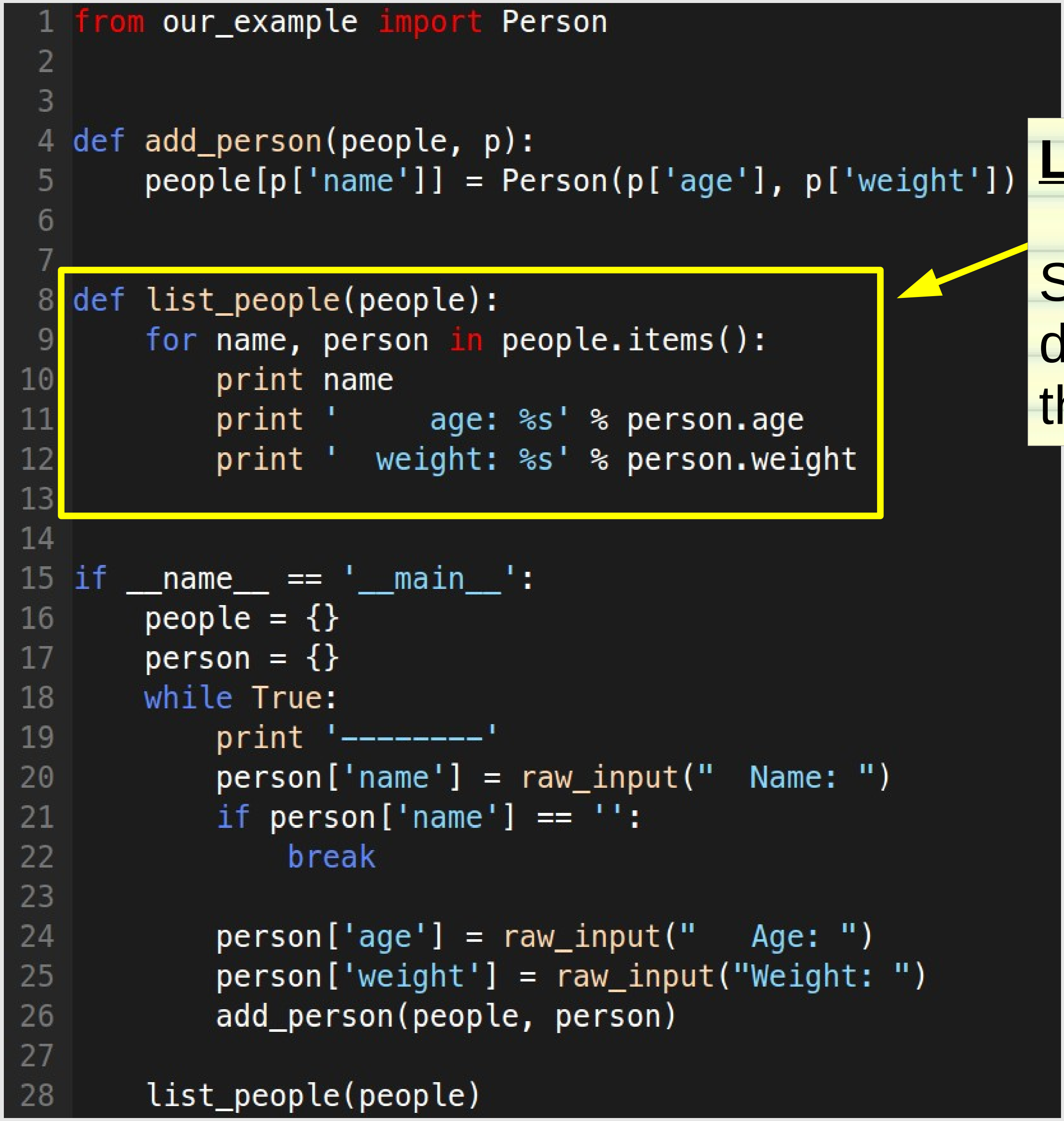
Adding the Person:

Insert the instance into the dictionary.

We have something like:

{'bob': <Person instance>}

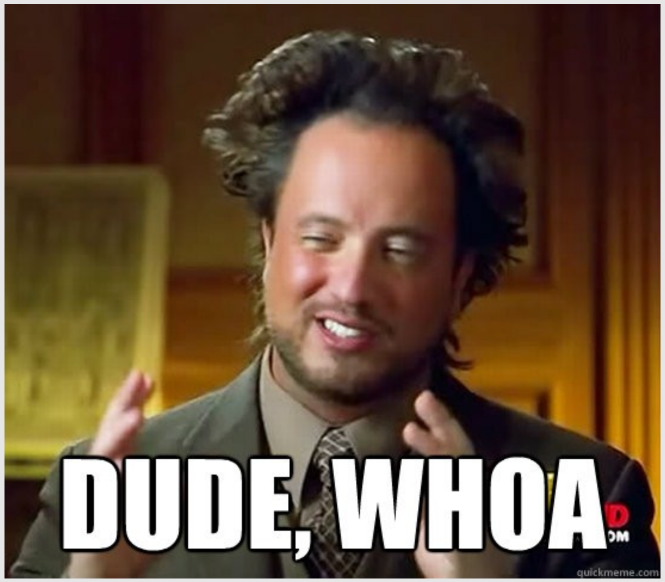
Building Dictionaries of Instances from User Input



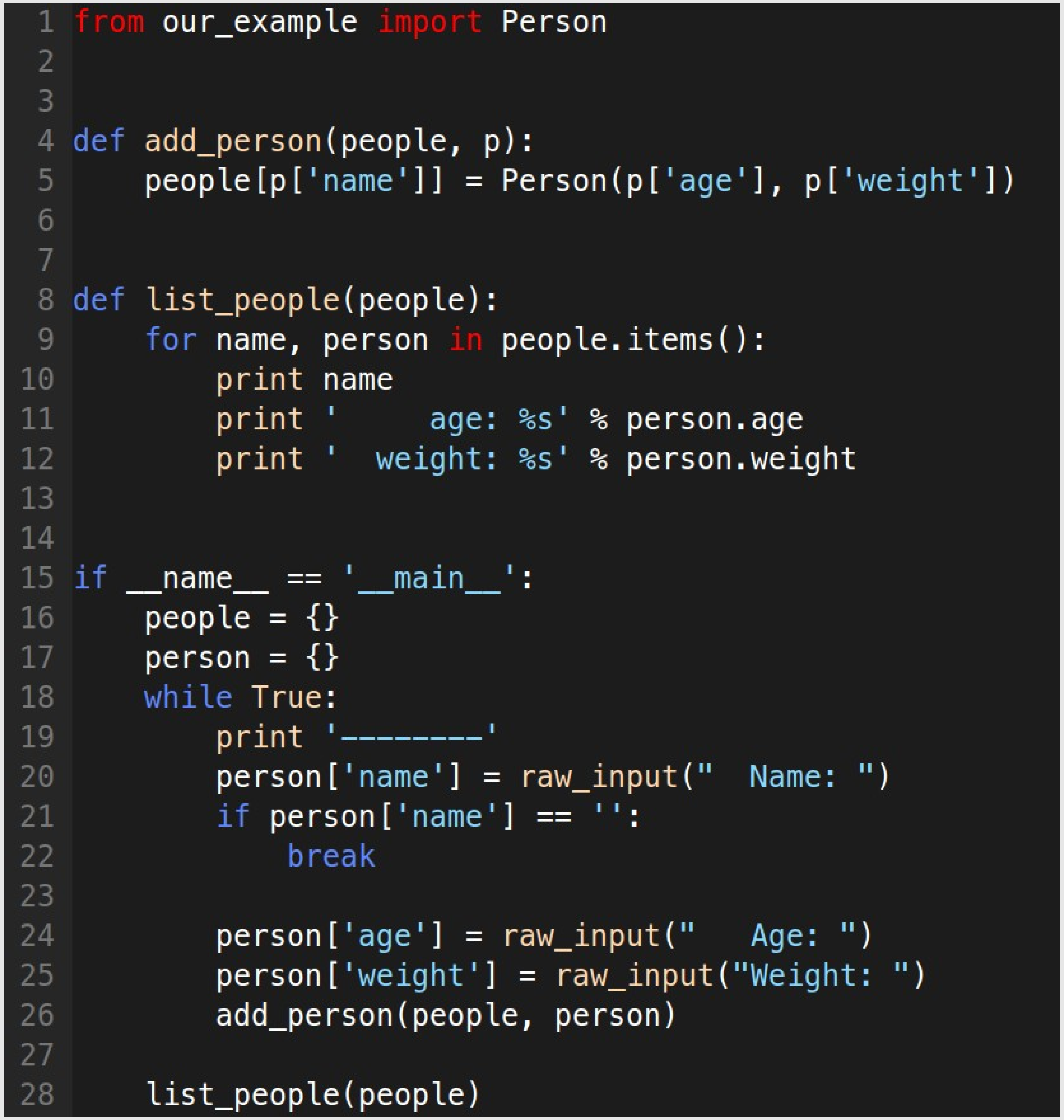
```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    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

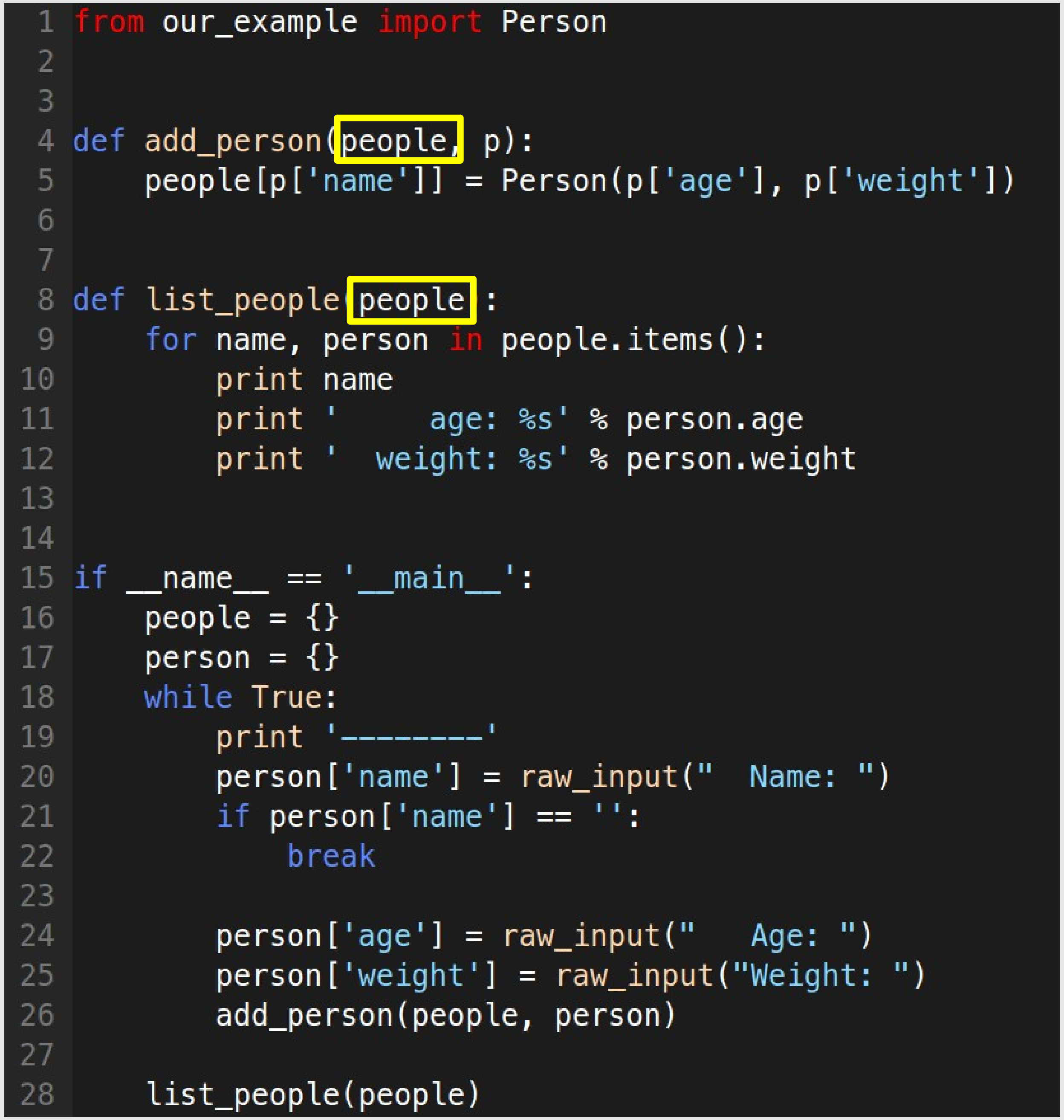


```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

Wait.

There is a pattern here.

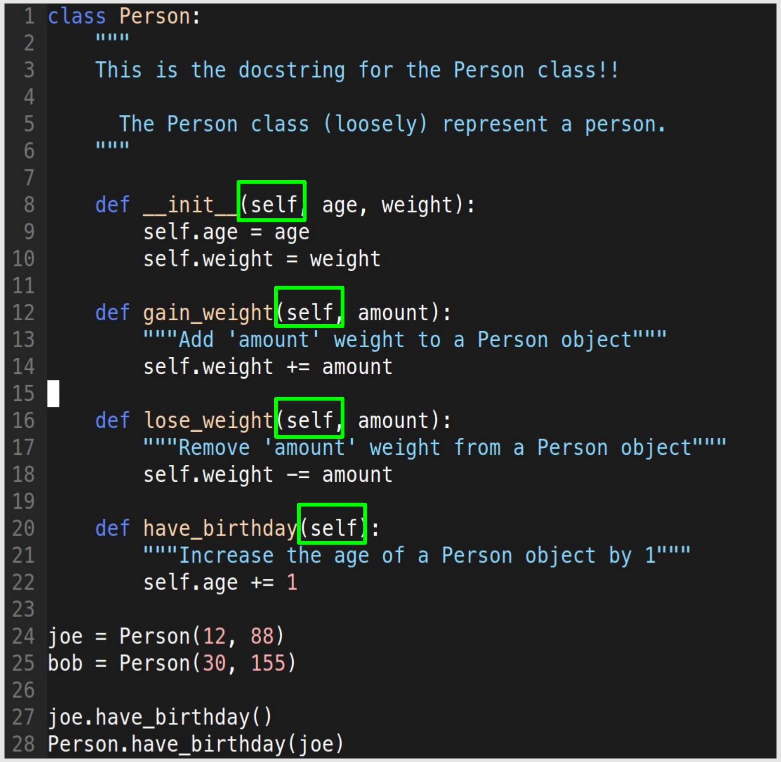
Building Dictionaries of Instances from User Input



```python
from our_example import Person


def add_person(people, p):
    people[p['name']] = Person(p['age'], p['weight'])


def list_people(people):
    for name, person in people.items():
        print name
        print '     age: %s' % person.age
        print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = {}
    person = {}
    while True:
        print '--------'
        person['name'] = raw_input("  Name: ")
        if person['name'] == '':
            break

        person['age'] = raw_input("   Age: ")
        person['weight'] = raw_input("Weight: ")
        add_person(people, person)

    list_people(people)
```

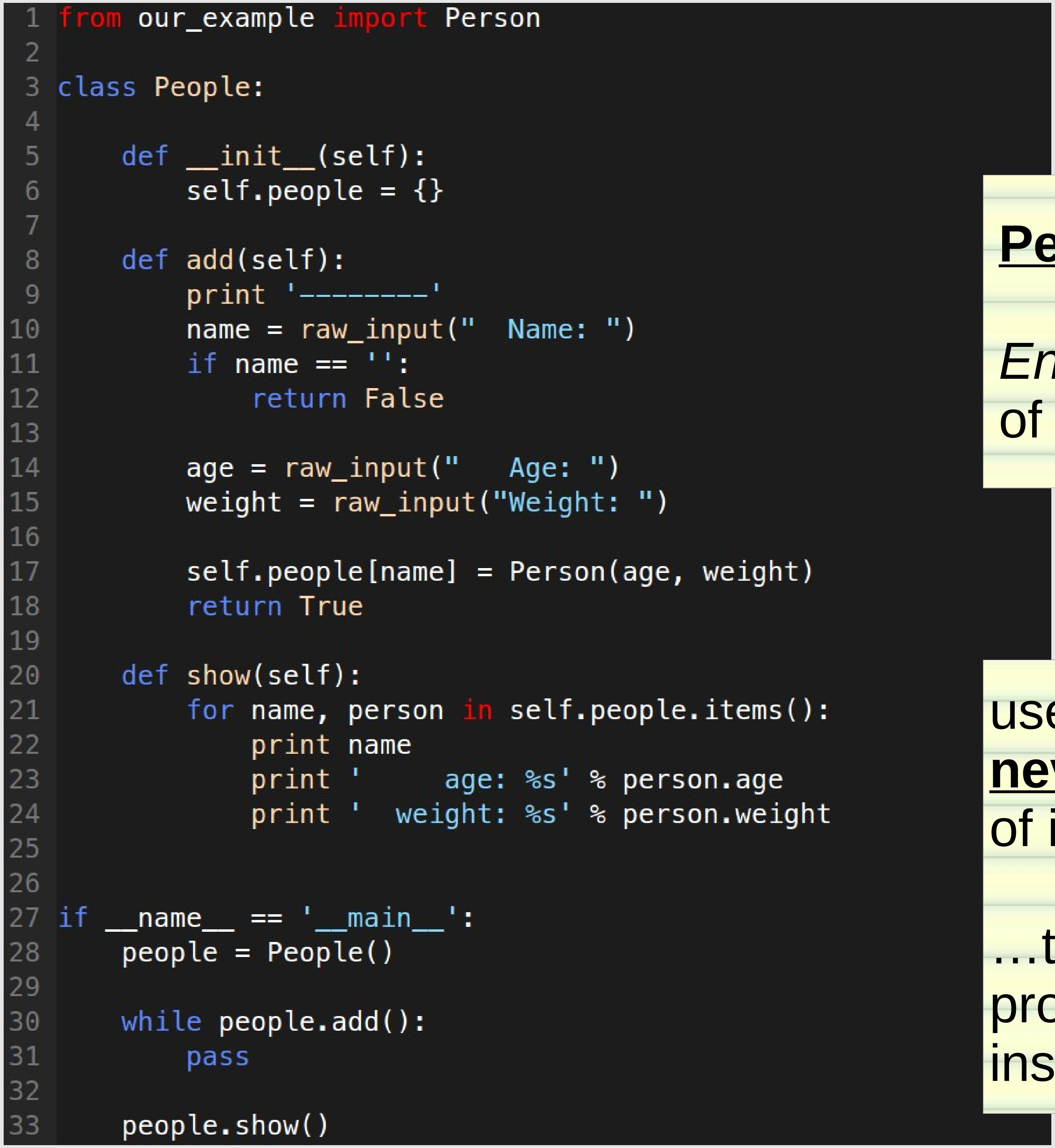
These functions all operate on people

maybe People should be an Object



```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

An example of encapsulation



```python
from our_example import Person

class People:

    def __init__(self):
        self.people = {}

    def add(self):
        print '--------'
        name = raw_input("  Name: ")
        if name == '':
            return False

        age = raw_input("   Age: ")
        weight = raw_input("Weight: ")

        self.people[name] = Person(age, weight)
        return True

    def show(self):
        for name, person in self.people.items():
            print name
            print '     age: %s' % person.age
            print '  weight: %s' % person.weight


if __name__ == '__main__':
    people = People()

    while people.add():
        pass

    people.show()
```

<u>People as a Class:</u>

Encapsulates a dictionary of Person instances

users of the People class **<u>never</u>** access the dictionary of instances directly

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