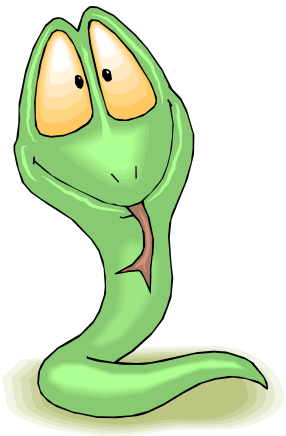
CSE-391: Artificial Intelligence
University of Pennsylvania
Matt Huenerfauth

Εισαγωγή στα Αντικείμενα

Δεκέμβριος 2018



Όλα είναι αντικείμενα…

- Στην Python ότι χρησιμοποιούμε είναι αντικείμενο:

```
'hello'.upper()
list3.append('a')
dict2.keys()
```

  - Οι κλήσεις αυτές μοιάζουν με την κλήση μεθόδων στην Java ή την C++.
  - Εκτός από τις έτοιμες κλάσεις της γλώσσας, μπορείτε να δημιουργήσετε δικά σας αντικείμενα…

Ορίζοντας μία Κλάση

- Η **κλάση** είναι ένας ειδικός τύπος δεδομένων που ορίζει τον τρόπο με τον οποίο μπορούμε να κατασκευάσουμε ένα αντικείμενο.
  - Η κλάση αποθηκεύει δεδομένα και συμπεριφορές που μοιράζονται όλα τα αντικείμενα της κλάσης αυτής.
  - 'Instances' είναι αντικείμενα που δημιουργήθηκαν και ακολουθούν τον ορισμό που δόθηκε από την κλάση.

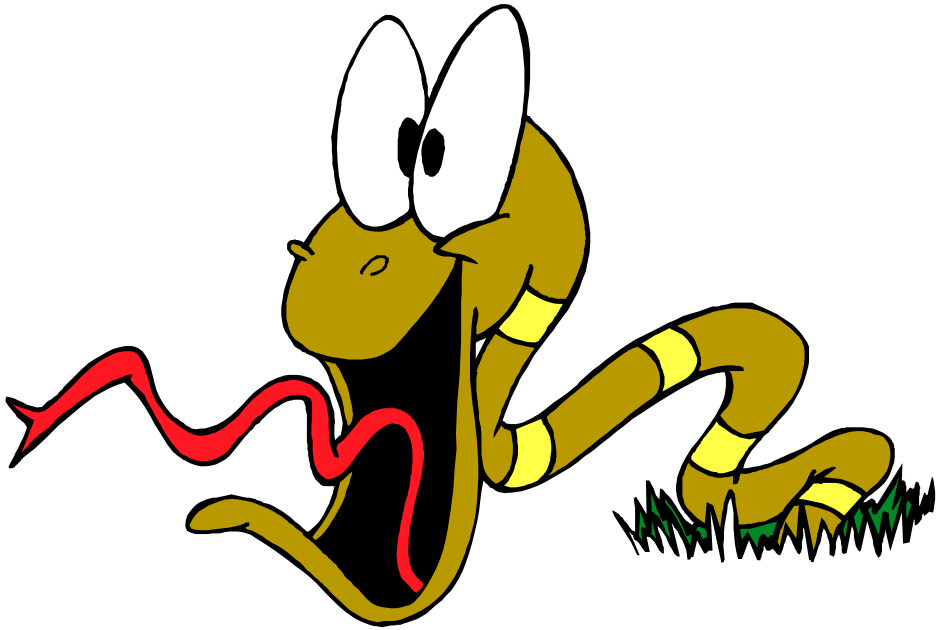
Methods σε κλάσεις

- Μπορείτε να ορίσετε μία μέθοδο σε μία κλάση ορίζοντας την σαν συνάρτηση μέσα στην κλάση.
  - Υπάρχει μία παράμετρος **`self`** σε όλες τις μεθόδους.
  - Συνήθως υπάρχει μία ειδική μέθοδος που λέγεται **`__init__`**
  - Θα μιλήσουμε για αυτά σε λίγο...

Ορισμός της κλάσης student

```
class Student:
  '''Κλάση που αντιπροσωπεύει
  έναν φοιτητή.'''
  def __init__(self,n,a):
      self.full_name = n
      self.age = a
  def get_age(self):
      return self.age
```

Δημιουργώντας και καταργώντας αντικείμενα (Instances)



Instantiating Objects

- Για να κατασκευάσουμε ένα αντικείμενο δεν χρησιμοποιούμε “new” όπως στην Java.
- Απλά χρησιμοποιούμε το όνομα της κλάσης με () και καταχωρούμε το αποτέλεσμα σε μία μεταβλητή.

```
b = Student('Bob Smith', 21)
```

- Τα arguments που περνάμε καταλήγουν στην μέθοδο **`__init__()`**.

Constructor: __init__

- **`__init__`** λειτουργεί σαν κατασκευαστής (constructor) για την κλάση.
  - Όταν δημιουργούμε ένα νέο αντικείμενο της κλάσης, η μέθοδος αυτή καλείται. Συνήθως εδώ γίνεται "αρχικοποίηση" των δεδομένων του αντικειμένου.
  - Περνάμε τις αρχικές τιμές στο αντικείμενο της κλάσης μέσω της μεθόδου **`__init__`** .

```
b = Student('Bob', 21)
```

Εδώ η **__init__** δέχεται "Bob" and 21.

Instantiating Objects

- Για να κατασκευάσουμε ένα αντικείμενο δεν χρησιμοποιούμε “new” όπως στην Java.
- Απλά χρησιμοποιούμε το όνομα της κλάσης με () και καταχωρούμε το αποτέλεσμα σε μία μεταβλητή.

```
b = Student(“Bob Smith”, 21)
```

- Τα arguments που περνάμε πάνε στην μέθοδο **`__init__()`**.

Constructor: __init__

- Η μέθοδος **`__init__`** μπορεί να πάρει όποιον αριθμό παραμέτρων χρειάζεται.
  - Όπως σε οποιαδήποτε άλλη συνάρτηση, οι παράμετροι μπορούν να ορίζονται με αρχικές τιμές και έτσι δεν είναι απαραίτητες για το καλών πρόγραμμα.

- Η πρώτη παράμετρος όμως, **`self`** στον ορισμό της __init__ είναι ειδική…

Self

- Το πρώτο argument κάθε μεθόδου είναι η αναφορά στο τρέχων αντικείμενο της κλάσης.
  - Εξ ορισμού, ονομάζουμε την παράμετρο αυτή **`self`**.

- Στην **`__init__`**, η self αναφέρεται στο αντικείμενο το οποίο δημιουργείται, έτσι σε όλες τις άλλες μεθόδους της κλάσης αναφέρεται στο αντικείμενο του οποίου η μέθοδος καλέστηκε.
  - Το self είναι σαν το keyword ‘this’ της Java ή C++.
  - Η Python χρησιμοποιεί το ‘self’ συχνότερα από ότι η Java χρησιμοποιεί ‘this.’

Self

- Παρά το ότι ορίζουμε τη **`self`**, όταν ορίζουμε την μέθοδο, δεν την περιλαμβάνουμε όταν την καλούμε.
- Η Python την περνάει αυτόματα.

Defining a method:

(this code inside a class definition.)

```
def set_age(self, num):
    self.age = num
```

Calling a method:

```
>>> x.set_age(23)
```

Δεν χρειάζεται “free”

- Όταν δεν χρειάζεται πλέον το αντικείμενο δεν είμαστε υποχρεωμένοι να το καταστρέψουμε.
  - Η Python έχει αυτόματο garbage collection.
  - Η Python βρίσκει ποιες αναφορές σε αντικείμενα δεν χρησιμοποιούνται και αυτόματα ελευθερώνει την μνήμη που δεσμεύουν.
  - Αυτό γενικά δουλεύει αρκετά καλά.
  - Δεν υπάρχει “destructor” method για τις κλάσεις.

Πρόσβαση στα attributes και μέθοδοι

Ορισμός κλάσης Student

```
class Student:
  '''κλάση που αναπαριστά ένα
  φοιτητή'''
  def __init__(self,n,a):
      self.full_name = n
      self.age = a
  def get_age(self):
      return self.age
```

Σύνταξη για πρόσβαση

```
>>> f = Student ("Bob Smith", 23)

>>> f.full_name      # Access an attribute.
"Bob Smith"

>>> f.get_age()      # Access a method.
23
```

Πρόσβαση σε άγνωστα members

- Τι συμβαίνει εάν δεν γνωρίζουμε το όνομα ενός attribute ή μίας μεθόδου μίας κλάσεις που θέλουμε να χρησιμοποιήσουμε έως το run time;

- Υπάρχει τρόπος να πάρουμε ένα string που να περιέχει το όνομα ενός attribute ή μεθόδου μίας κλάσης και μία αναφορά σε αυτό για να το χρησιμοποιήσουμε;

getattr(object_instance, string)

```
>>> f = Student('Bob Smith', 23)

>>> getattr(f, 'full_name')
"Bob Smith"

>>> getattr(f, 'get_age')
 <method get_age of class studentClass at 010B3C2>

>>> getattr(f, 'get_age')()   # We can call this.
23

>>> getattr(f, 'get_birthday')
      # Raises AttributeError - No method exists.
```

hasattr(object_instance,string)

```
>>> f = Student('Bob Smith', 23)

>>> hasattr(f, 'full_name')
True

>>> hasattr(f, 'get_age')
True

>>> hasattr(f, 'get_birthday')
False
```