Αντικείμενα Python

Κεφάλαιο 14

Python για Όλους
www.py4e.com

Προειδοποίηση

- Αυτή η διάλεξη αφορά κυρίως τους ορισμούς και τη μηχανική για τα αντικείμενα
- Αυτή η διάλεξη είναι πολύ περισσότερο για το «πώς λειτουργεί» και λιγότερο για το «πώς το χρησιμοποιείτε»
- Δεν θα έχετε ολόκληρη την εικόνα παρά μόνο όταν όλο αυτό εξεταστεί στο πλαίσιο ενός πραγματικού προβλήματος
- Επομένως, σας παρακαλώ να αναστείλετε την δυσπιστία σας και να μάθετε τις τεχνικές για τις επόμενες 40 περίπου διαφάνειες …

5. Δομές Δεδομένων

Αυτό το κεφάλαιο περιγράφει μερικά πράγματα για τα οποία έχετε μάθει ήδη με περισσότερες λεπτομέρειες και προσθέτει επίσης μερικά νέα πράγματα.

5.1 Περισσότερα για Λίστες

Ο τύπος δεδομένων λίστας έχει μερικές ακόμη μεθόδους. Ακολουθούν όλες οι μέθοδοι αντικειμένων λίστας:

list.**append** (x)
Προσθέτει ένα στοιχείο στο τέλος της λίστας. Ισοδύναμο με a[len (a):] = [x].

list.**extend** (L)
Επεκτείνει τη λίστα προσθέτοντας όλα τα στοιχεία τις δοθείσας λίστας. Ισοδύναμο με a[len (a):] = L.

list.**insert** (i, x)
Τοποθετεί ένα στοιχείο σε μια δεδομένη θέση. Το πρώτο όρισμα είναι ο δείκτης του στοιχείου πριν από το οποίο πρέπει να εισαχθεί, οπότε το a.insert (0, x) εισάγει στην αρχή της λίστας και το a.insert (len (a), x) ισοδυναμεί με a.append (x).

list.**remove** (x)
Αφαιρεί το πρώτο στοιχείο από τη λίστα, του οποίου η τιμή είναι ίση με x. Προκαλεί σφάλμα εάν δεν υπάρχει τέτοιο στοιχείο.

list.**pop** ([i])
Αφαιρεί το στοιχείο στη δεδομένη θέση της λίστας και το επιστρέφει. Εάν δεν έχει καθοριστεί δείκτης, το a.pop () αφαιρεί και επιστρέφει το τελευταίο στοιχείο στη λίστα. (Οι αγκύλες γύρω από το i στην σύνταξη της μεθόδου υποδηλώνουν ότι η παράμετρος είναι προαιρετική και όχι ότι πρέπει να πληκτρολογείτε αγκύλες σε αυτήν τη θέση. Αυτή τη σημείωση θα τη βλέπετε συχνά στην αναφορά βιβλιοθηκών Python.)

https://docs.python.org/3/tutorial/datastructures.html

sqlite3 – Διασύνδεση DB-API 2.0 για βάσεις δεδομένων SQLite

Πηγαίος κώδικας: Lib/sqlite3/

Η SQLite είναι μια βιβλιοθήκη C που παρέχει μια ελαφριά βάση δεδομένων που βασίζεται σε δίσκο και δεν απαιτεί ξεχωριστή διαδικασία διακομιστή και επιτρέπει την πρόσβαση στη βάση δεδομένων χρησιμοποιώντας μια μη τυποποιημένη παραλλαγή της γλώσσας ερωτημάτων SQL. Ορισμένες εφαρμογές μπορούν να χρησιμοποιήσουν την SQLite για εσωτερική αποθήκευση δεδομένων. Είναι επίσης δυνατό στο πρωτότυπο μιας εφαρμογής να χρησιμοποιήσετε την SQLite και στη συνέχεια να μεταφέρετε τον κώδικα σε μια μεγαλύτερη βάση δεδομένων όπως η PostgreSQL ή η Oracle.

Η ενότητα sqlite3 γράφτηκε από τον Gerhard Häring. Παρέχει μια διεπαφή SQL συμβατή με τις προδιαγραφές DB-API 2.0 που περιγράφονται από το **PEP 249**.

Για να χρησιμοποιήσετε τη μονάδα, πρέπει πρώτα να δημιουργήσετε ένα αντικείμενο σύνδεσης που αντιπροσωπεύει τη βάση δεδομένων. Εδώ τα δεδομένα θα αποθηκευτούν στο αρχείο example.db:

```
import sqlite3
con = sqlite3.connect('example.db')
```

Μπορείτε επίσης να δώσετε το ειδικό όνομα :memory: για να δημιουργήσετε μια βάση δεδομένων στη μνήμη RAM.

Μόλις έχετε μια σύνδεση, μπορείτε να δημιουργήσετε ένα αντικείμενο δρομέα και να καλέσετε τη μέθοδο execute() για την εκτέλεση εντολών SQL:

```
cur = con.cursor()

# Create table
cur.execute('''CREATE TABLE stocks
               (date text, trans text, symbol text, qty real, price real)''')
```

https://docs.python.org/3/library/sqlite3.html

Ας Ξεκινήσουμε με τα Προγράμματα

```
inp = input('Ευρωπαϊκός όροφος;')
usf = int(inp) + 1
print('Όροφος στις ΗΠΑ', usf)
```

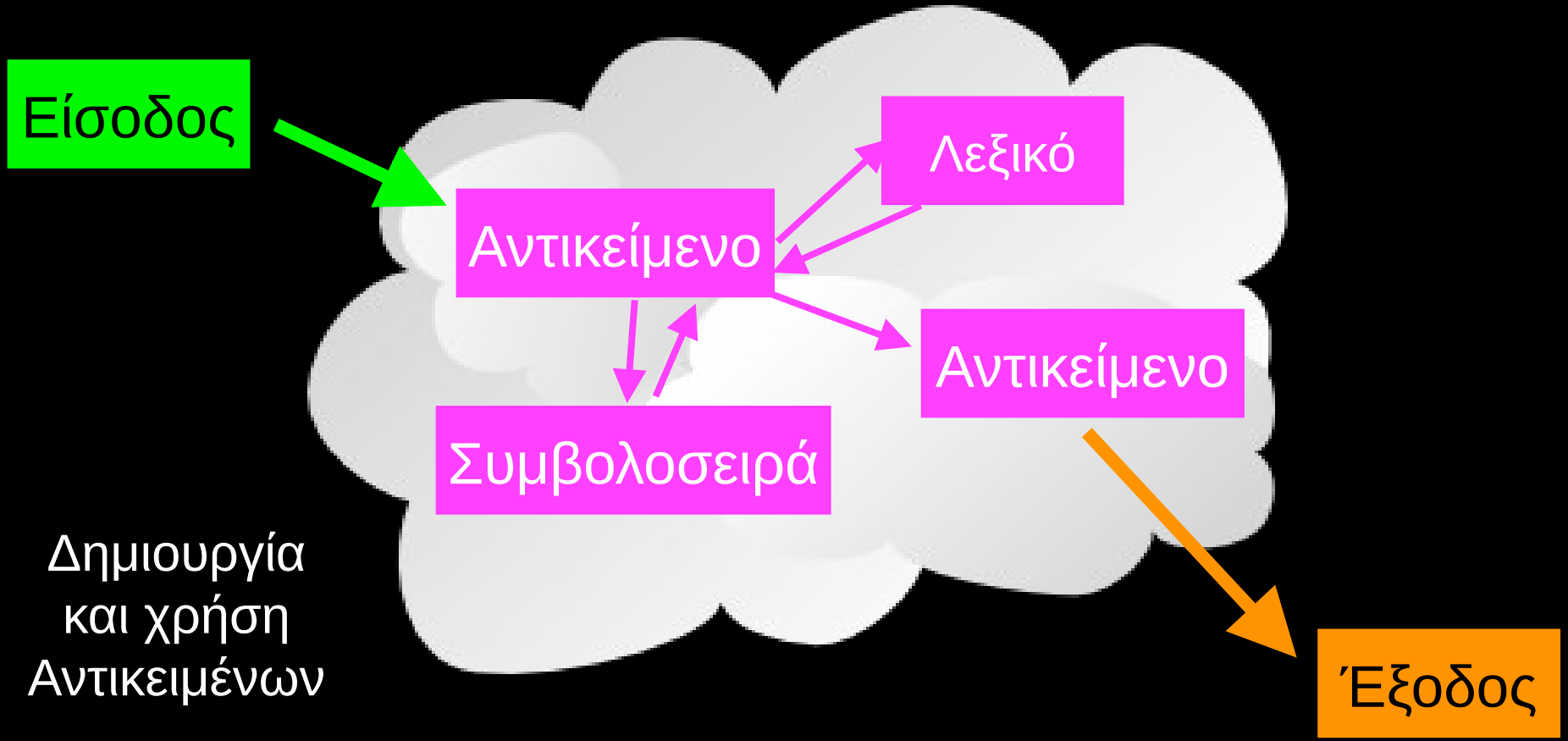
Ευρωπαϊκός όροφος; 0
Όροφος στις ΗΠΑ 1

Αντικειμενοστραφής

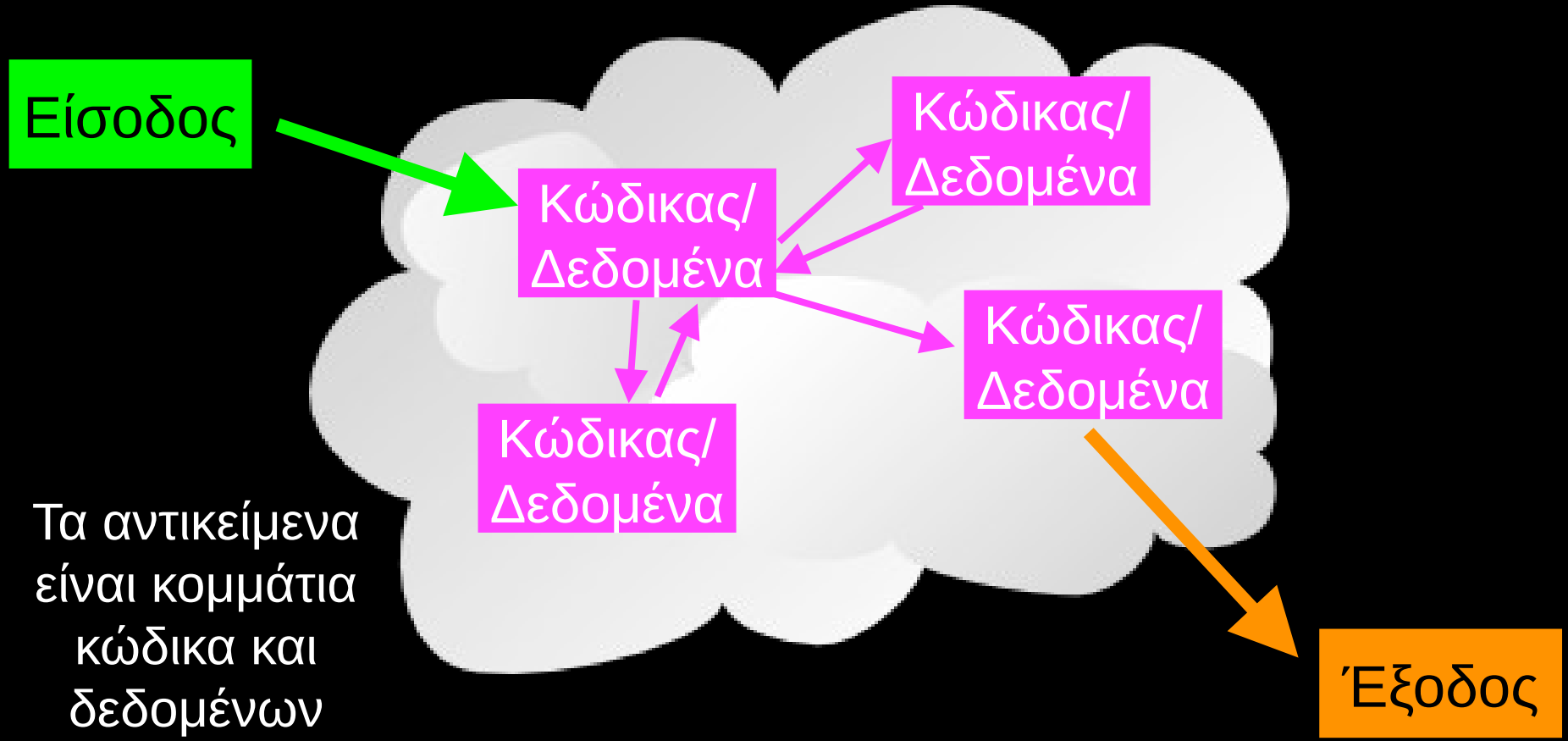
- Ένα πρόγραμμα αποτελείται από πολλά συνεργαζόμενα αντικείμενα
- Αντί να είναι το «ολόκληρο πρόγραμμα» - κάθε αντικείμενο είναι ένα μικρό «νησί» μέσα στο πρόγραμμα και συνεργάζεται με άλλα αντικείμενα
- Ένα πρόγραμμα αποτελείται από ένα ή περισσότερα αντικείμενα που συνεργάζονται - τα αντικείμενα χρησιμοποιούν τις δυνατότητες των άλλων αντικειμένων

Αντικείμενο

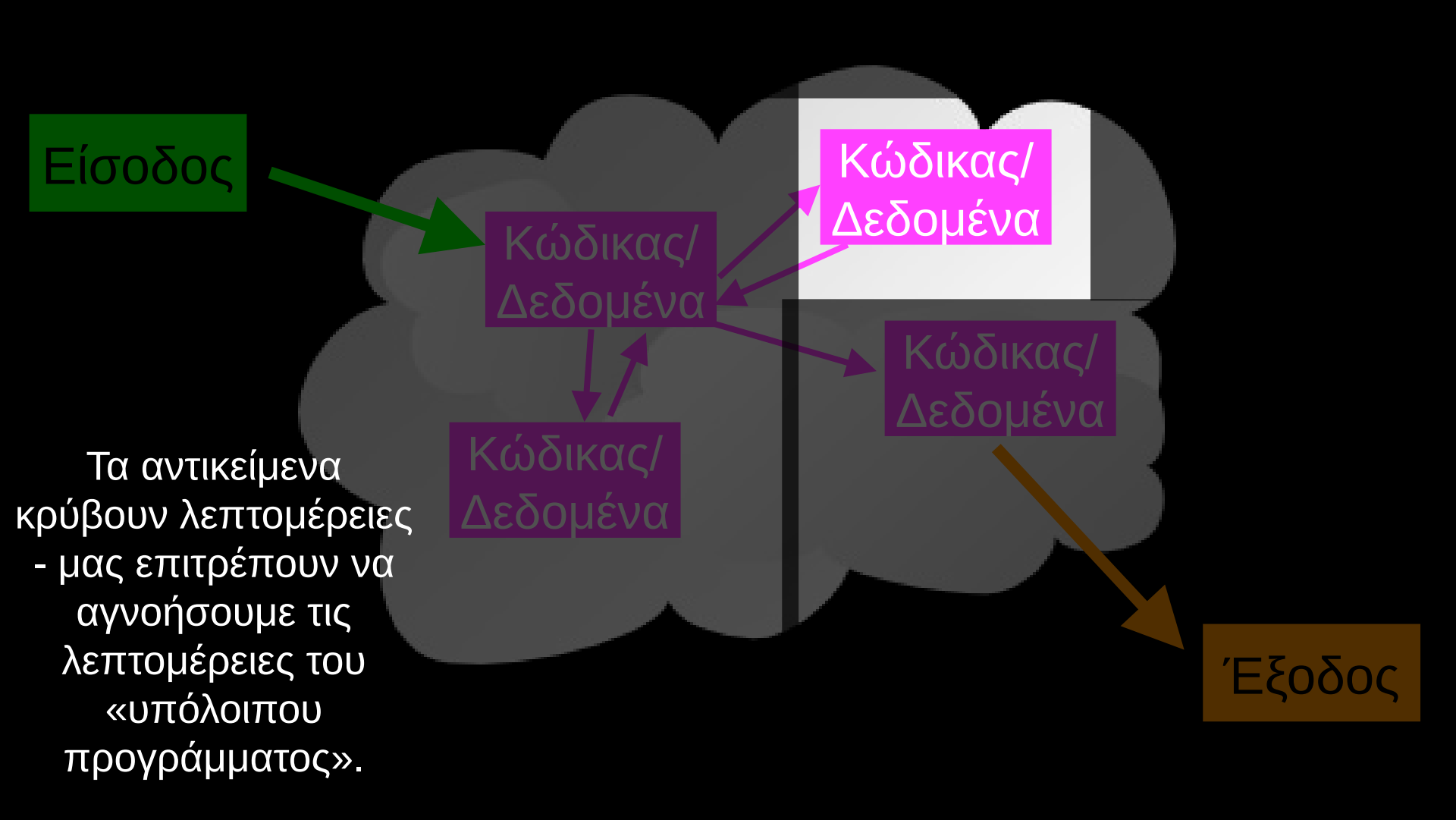
- Ένα αντικείμενο είναι ένα κομμάτι αυτοτελούς κώδικα και δεδομένων
- Μια βασική πτυχή της προσέγγισης αντικείμενο είναι η διάσπαση του προβλήματος σε μικρότερα κατανοητά μέρη (διαίρει και βασίλευε)
- Τα αντικείμενα έχουν όρια που μας επιτρέπουν να αγνοούμε τις περιττές λεπτομέρειες
- Χρησιμοποιούσαμε αντικείμενα εξ αρχής: Αντικείμενα Συμβολοσειράς, Αντικείμενα Ακεραίου, Αντικείμενα Λεξικού, Αντικείμενα Λίστας...



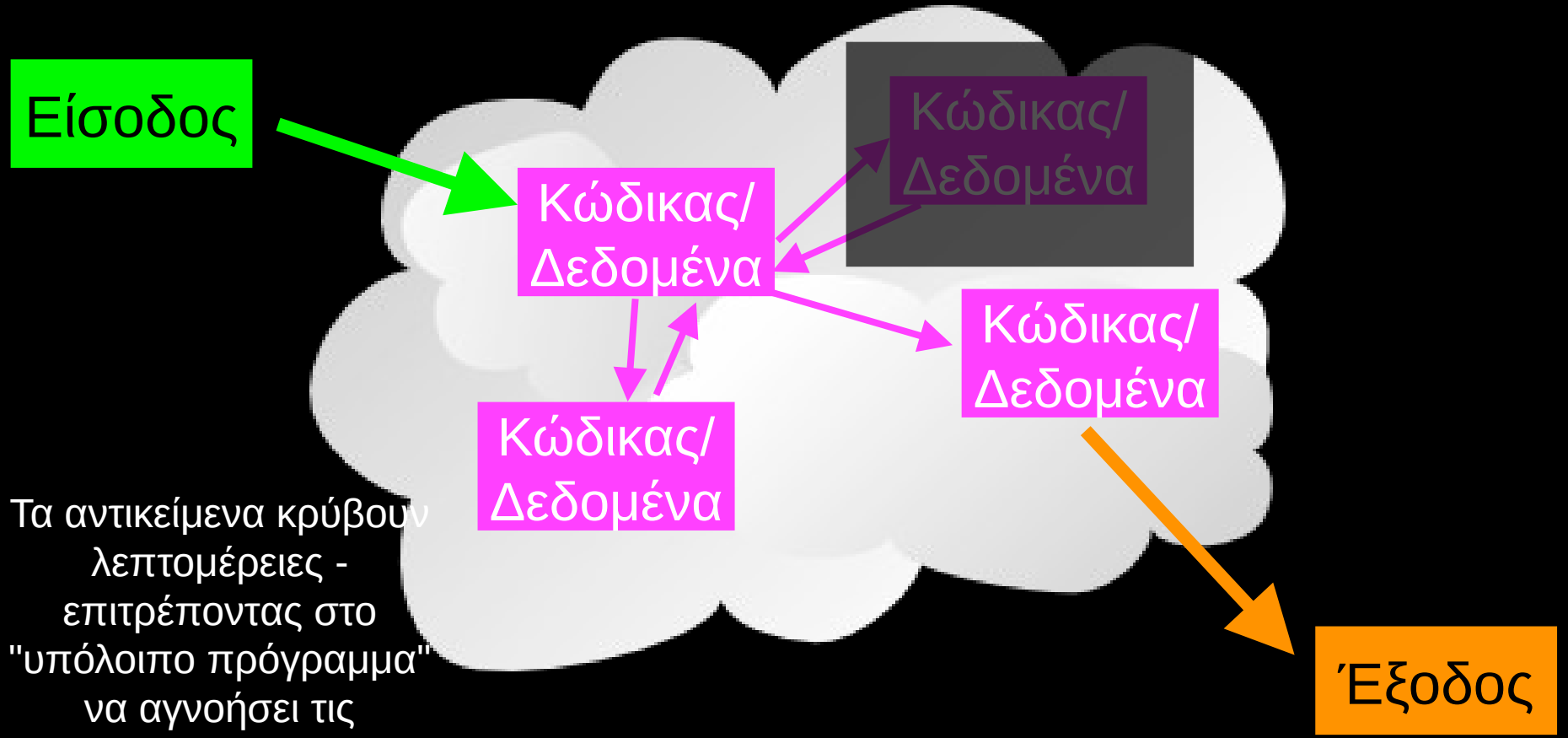
Δημιουργία
και χρήση
Αντικειμένων



Τα αντικείμενα είναι κομμάτια κώδικα και δεδομένων



Τα αντικείμενα κρύβουν λεπτομέρειες - μας επιτρέπουν να αγνοήσουμε τις λεπτομέρειες του «υπόλοιπου προγράμματος».



Τα αντικείμενα κρύβουν λεπτομέρειες - επιτρέποντας στο "υπόλοιπο πρόγραμμα" να αγνοήσει τις λεπτομέρειες σχετικά με το "εμείς".

Ορισμοί

- Κλάση - ένα πρότυπο
- Μέθοδος ή Μήνυμα - Μια καθορισμένη δυνατότητα μιας κλάσης
- Πεδίο ή Χαρακτηριστικό - Ένα κομμάτι δεδομένων σε μια κλάση
- Αντικείμενο ή Στιγμιότυπο - Μια συγκεκριμένη παρουσία μιας κλάσης

Ορολογία: Κλάση

Ορίζει τα αφηρημένα χαρακτηριστικά ενός αντικειμένου, συμπεριλαμβανομένων των χαρακτηριστικών του αντικειμένου (τα πεδία ή τις ιδιότητες του) και τη συμπεριφορά του αντικειμένου (τα πράγματα που μπορεί να κάνει, ή μεθόδους, λειτουργίες ή χαρακτηριστικά). Θα μπορούσε κάποιος να πει ότι μια κλάση είναι ένα σχέδιο ή ένα εργοστάσιο που περιγράφει τη φύση «του κάτι». Για παράδειγμα, η κλάση Σκύλος θα αποτελείται από χαρακτηριστικά που μοιράζονται όλα τα σκυλιά, όπως η φυλή και το χρώμα της γούνας (χαρακτηριστικά) και η ικανότητα να γαβγίζουν και να κάθονται (συμπεριφορές).

http://en.wikipedia.org/wiki/Object-oriented_programming

Ορολογία: Στιγμιότυπο

Κάποιος μπορεί να έχει ένα στιγμιότυπο μιας κλάσης ή ένα συγκεκριμένο αντικείμενο. Το στιγμιότυπο είναι το πραγματικό αντικείμενο που δημιουργήθηκε κατά την εκτέλεση. Στην προγραμματιστική ορολογία, το αντικείμενο Λάση είναι ένα στιγμιότυπο της κλάσης Σκύλος. Το σύνολο τιμών των χαρακτηριστικών ενός συγκεκριμένου αντικειμένου ονομάζεται κατάστασή του. Το αντικείμενο αποτελείται από την κατάσταση και τη συμπεριφορά που ορίζεται στην κλάση του αντικειμένου.

Το αντικείμενο και το στιγμιότυπο χρησιμοποιούνται συχνά ως ισοδύναμα.

http://en.wikipedia.org/wiki/Object-oriented_programming

Ορολογία: Μέθοδος

Οι ικανότητες ενός αντικειμένου. Στη γλώσσα, οι μέθοδοι είναι ρήματα. Η Λάσι, όντας Σκύλος, έχει την ικανότητα να γαβγίζει. Έτσι το γαυγίζω() είναι μία από τις μεθόδους της Λάσι. Μπορεί επίσης να έχει κι άλλες μεθόδους, για παράδειγμα κάτσε() ή φάε() ή περπάτα() ή σώσε_timmy(). Μέσα στο πρόγραμμα, η χρήση μιας μεθόδου επηρεάζει συνήθως μόνο ένα συγκεκριμένο αντικείμενο. όλα τα σκυλιά μπορούν να γαβγίσουν, αλλά χρειάζεστε μόνο ένα συγκεκριμένο σκυλί για να κάνει το γάβγισμα

Η Μέθοδος και το Μήνυμα χρησιμοποιούνται συχνά ως ισοδύναμα.

http://en.wikipedia.org/wiki/Object-oriented_programming

Μερικά Αντικείμενα της Python

```
>>> x = 'abc'
>>> type(x)
<class 'str'>
>>> type(2.5)
<class 'float'>
>>> type(2)
<class 'int'>
>>> y = list()
>>> type(y)
<class 'list'>
>>> z = dict()
>>> type(z)
<class 'dict'>
```

```
>>> dir(x)
[ … 'capitalize', 'casefold', 'center', 'count',
'encode', 'endswith', 'expandtabs', 'find',
'format', … 'lower', 'lstrip', 'maketrans',
'partition', 'replace', 'rfind', 'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
>>> dir(y)
[… 'append', 'clear', 'copy', 'count', 'extend',
'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
>>> dir(z)
[…, 'clear', 'copy', 'fromkeys', 'get', 'items',
'keys', 'pop', 'popitem', 'setdefault', 'update',
'values']
```

Ένα Δείγμα Κλάσης

class είναι μια δεσμευμένη λέξη

Κάθε αντικείμενο PartyAnimal έχει λίγο κώδικα

Λέει στο αντικείμενο an να εκτελέσει τον κώδικα party() μέσα σε αυτό

```
class PartyAnimal:
   x = 0

   def party(self) :
     self.x = self.x + 1
     print("Μέχρι στιγμής",self.x)

an = PartyAnimal()

an.party()
an.party()
an.party()
```

Αυτό είναι το πρότυπο για την κατασκευή του αντικειμένου PartyAnimal

Κάθε αντικείμενο PartyAnimal έχει λίγα δεδομένα

Κατασκευάζει ένα αντικείμενο PartyAnimal και το αποθηκεύει στο an

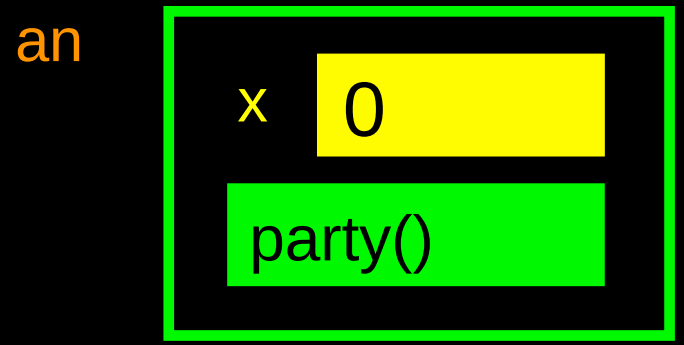
PartyAnimal.party(an)

```
class PartyAnimal:
  x = 0

  def party(self) :
    self.x = self.x + 1
    print("Μέχρι στιγμής",self.x)

an = PartyAnimal()

an.party()
an.party()
an.party()
```

```
$ python party1.py
```

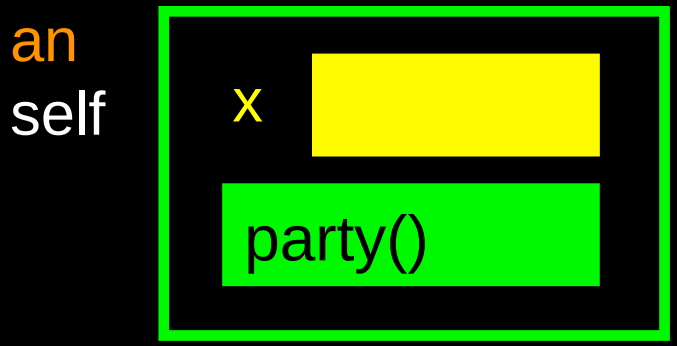
```
class PartyAnimal:
  x = 0

  def party(self) :
    self.x = self.x + 1
    print("Μέχρι στιγμής",self.x)

an = PartyAnimal()

an.party()
an.party()
an.party()
```

$ python party1.py



```
class PartyAnimal:
  x = 0

  def party(self) :
    self.x = self.x + 1
    print("Μέχρι στιγμής",self.x)

an = PartyAnimal()

an.party()
an.party()
an.party()
```

```
$ python party1.py
Μέχρι στιγμής 1
Μέχρι στιγμής 2
Μέχρι στιγμής 3
```



PartyAnimal.party(an)

Παίζοντας με τα dir() και type()

Ένας Σπαστικός Τρόπος για να Βρείτε Δυνατότητες

- Η εντολή dir() απαριθμεί τις δυνατότητες
- Αγνοήστε αυτά με κάτω παύλα - αυτά χρησιμοποιούνται από την ίδια την Python
- Οι υπόλοιπες είναι πραγματικές λειτουργίες που μπορεί να εκτελέσει το αντικείμενο
- Είναι σαν τον type() - μας λέει κάτι * για * μια μεταβλητή

```
>>> y = list()
>>> type(y)
<class 'list'>
>>> dir(y)
['__add__', '__class__',
'__contains__', '__delattr__',
'__delitem__', '__delslice__',
'__doc__', … '__setitem__',
'__setslice__', '__str__',
'append', 'clear', 'copy',
'count', 'extend', 'index',
'insert', 'pop', 'remove',
'reverse', 'sort']
>>>
```

```
class PartyAnimal:
    x = 0

    def party(self) :
      self.x = self.x + 1
      print("Μέχρι στιγμής",self.x)

an = PartyAnimal()

print("Type", type(an))
print("Dir ", dir(an))
```

Μπορούμε να χρησιμοποιήσουμε το dir() για να βρούμε τις «δυνατότητες» της νεοδημιουργημένης κλάσης μας.

```
$ python party3.py
Type <class '__main__.PartyAnimal'>
Dir  ['__class__', ... 'party', 'x']
```

Δοκιμάστε το dir() με μια Συμβολοσειρά

```
>>> x = 'Γειά σας'
>>> dir(x)
['__add__', '__class__', '__contains__', '__delattr__',
'__doc__', '__eq__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__le__', '__len__', '__lt__',
'__repr__', '__rmod__', '__rmul__', '__setattr__', '__str__',
'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs', 'find', 'index', 'isalnum', 'isalpha', 'isdigit',
'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust',
'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

Κύκλος Ζωής Αντικειμένων

http://en.wikipedia.org/wiki/Constructor_(computer_science)

Κύκλος Ζωής Αντικειμένων

- Τα αντικείμενα δημιουργούνται, χρησιμοποιούνται και απορρίπτονται
- Έχουμε ειδικά μπλοκ κώδικα (μεθόδους) που καλούνται
  - Τη στιγμή της δημιουργίας (κατασκευαστής)
  - Τη στιγμή της καταστροφής (καταστροφέας)
- Οι κατασκευαστές χρησιμοποιούνται πολύ
- Οι καταστροφείς χρησιμοποιούνται σπάνια

Κατασκευαστής

Ο πρωταρχικός σκοπός του κατασκευαστή είναι να δημιουργήσει ορισμένες μεταβλητές στιγμιότυπου ώστε να έχουν τις κατάλληλες αρχικές τιμές όταν δημιουργείται το αντικείμενο

```
class PartyAnimal:
   x = 0

   def __init__(self):
     print('Κατασκευάστηκα')

   def party(self) :
     self.x = self.x + 1
     print('Μέχρι στιγμής',self.x)

   def __del__(self):
     print('Καταστράφηκα', self.x)

an = PartyAnimal()
an.party()
an.party()
an = 42
print('το an περιέχει το',an)
```

```
$ python party4.py
Κατασκευάστηκα
Μέχρι στιγμής 1
Μέχρι στιγμής 2
Καταστράφηκα 2
το an περιέχει το 42
```

Ο κατασκευαστής και ο καταστροφέας είναι προαιρετικοί. Ο κατασκευαστής χρησιμοποιείται συνήθως για τη ρύθμιση μεταβλητών. Ο καταστροφέας χρησιμοποιείται σπάνια.

Κατασκευαστής

Στον αντικειμενοστραφή προγραμματισμό, ένας κατασκευαστής σε μια κλάση είναι ένα ειδικό μπλοκ δηλώσεων που καλείται όταν δημιουργείται ένα αντικείμενο

http://en.wikipedia.org/wiki/Constructor_(computer_science)

Πολλά Στιγμιότυπα

- Μπορούμε να δημιουργήσουμε πολλά αντικείμενα - η κλάση είναι το πρότυπο για το αντικείμενο
- Μπορούμε να αποθηκεύσουμε κάθε ξεχωριστό αντικείμενο στη δική του μεταβλητή
- Αυτό το ονομάζουμε πολλαπλά στιγμιότυπα της ίδιας κλάσης
- Κάθε στιγμιότυπο έχει το δικό της αντίγραφο των μεταβλητών στιγμιότυπου

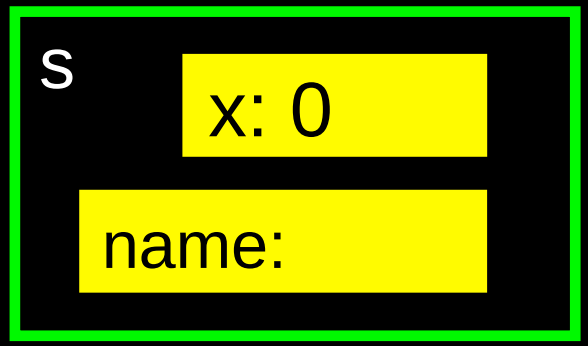
```
class PartyAnimal:
    x = 0
    name = ""
    def __init__(self, z):
      self.name = z
      print(self.name,"Κατασκευάστηκε")

    def party(self) :
      self.x = self.x + 1
      print(self.name,"πλήθος party",self.x)

s = PartyAnimal("Sally")
j = PartyAnimal("Jim")

s.party()
j.party()
s.party()
```

Οι κατασκευαστές μπορούν να έχουν πρόσθετες παραμέτρους. Αυτές μπορούν να χρησιμοποιηθούν για τη ρύθμιση μεταβλητών στιγμιότυπου για το συγκεκριμένο στιγμιότυπο της κλάσης (δηλαδή, για το συγκεκριμένο αντικείμενο).

party5.py

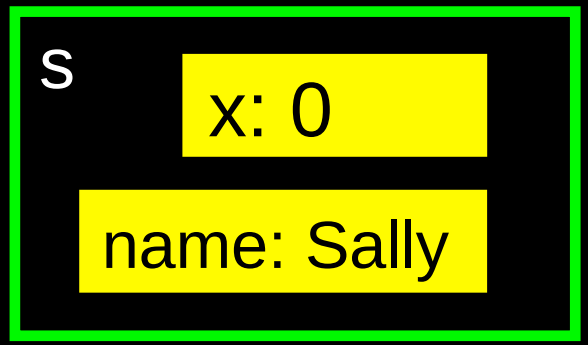
```
class PartyAnimal:
    x = 0
    name = ""
    def __init__(self, z):
      self.name = z
      print(self.name,"Κατασκευάστηκε")

    def party(self) :
      self.x = self.x + 1
      print(self.name,"πλήθος party",self.x)

s = PartyAnimal("Sally")
j = PartyAnimal("Jim")

s.party()
j.party()
s.party()
```

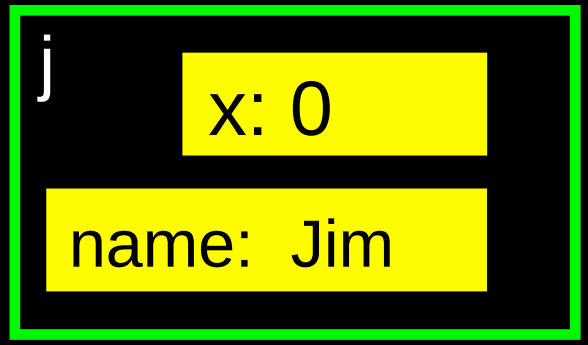
```
class PartyAnimal:
    x = 0
    name = ""
    def __init__(self, z):
      self.name = z
      print(self.name,"Κατασκευάστηκε")

    def party(self) :
      self.x = self.x + 1
      print(self.name,"πλήθος party",self.x)

s = PartyAnimal("Sally")
j = PartyAnimal("Jim")

s.party()
j.party()
s.party()
```



```
class PartyAnimal:
    x = 0
    name = ""
    def __init__(self, z):
      self.name = z
      print(self.name,"Κατασκευάστηκε")

    def party(self) :
      self.x = self.x + 1
      print(self.name,"πλήθος party",self.x)

s = PartyAnimal("Sally")
j = PartyAnimal("Jim")

s.party()
j.party()
s.party()
```

Έχουμε δύο ανεξάρτητα στιγμιότυπα





```
class PartyAnimal:
    x = 0
    name = ""
    def __init__(self, z):
      self.name = z
      print(self.name,"Κατασκευάστηκε")

    def party(self) :
      self.x = self.x + 1
      print(self.name,"πλήθος party",self.x)

s = PartyAnimal("Sally")
j = PartyAnimal("Jim")

s.party()
j.party()
s.party()
```

```
Sally Κατασκευάστηκε
Jim Κατασκευάστηκε
Sally πλήθος party 1
Jim πλήθος party 1
Sally πλήθος party 2
```

Κληρονομικότητα

http://www.ibiblio.org/g2swap/byteofpython/read/inheritance.html

Κληρονομικότητα

- Όταν δημιουργούμε μια νέα κλάση - μπορούμε να επαναχρησιμοποιήσουμε μια υπάρχουσα κλάση και να κληρονομήσει όλες τις δυνατότητες της υπάρχουσας κλάσης και στη συνέχεια να προσθέσουμε το δικό μας κομμάτι για να κάνουμε τη νέα μας κλάση
- Μια άλλη μορφή αποθήκευσης και επαναχρησιμοποίησης
- Γράφω μία φορά - επαναχρησιμοποιώ πολλές φορές
- Η νέα κλάση (παιδί) έχει όλες τις δυνατότητες της παλιάς κλάσης (γονέα) - και μερικές ακόμη

Ορολογία: Κληρονομικότητα

Οι «υποκλάσεις» είναι πιο εξειδικευμένες εκδόσεις μιας κλάσης, οι οποίες κληρονομούν χαρακτηριστικά και συμπεριφορές από τις γονικές τους κλάσεις και μπορούν να εισαγάγουν και επιπλέον δικές τους.

http://en.wikipedia.org/wiki/Object-oriented_programming

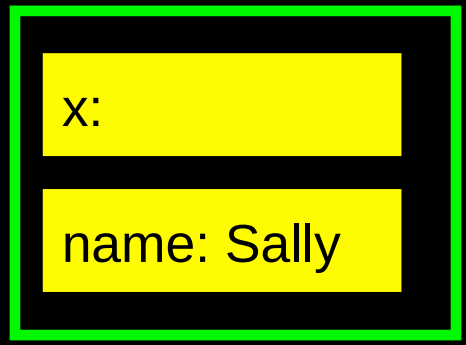
```
class PartyAnimal:
   x = 0
   name = ""
   def __init__(self, nam):
     self.name = nam
     print(self.name, "κατασκευάστηκε")

   def party(self) :
     self.x = self.x + 1
     print(self.name,"πλήθος party",self.x)

class FootballFan(PartyAnimal):
   points = 0
   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name, "πόντοι",self.points)
```

```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```

Το FootballFan είναι μια κλάση που επεκτείνει το PartyAnimal. Έχει όλες τις δυνατότητες του PartyAnimal και όχι μόνο.

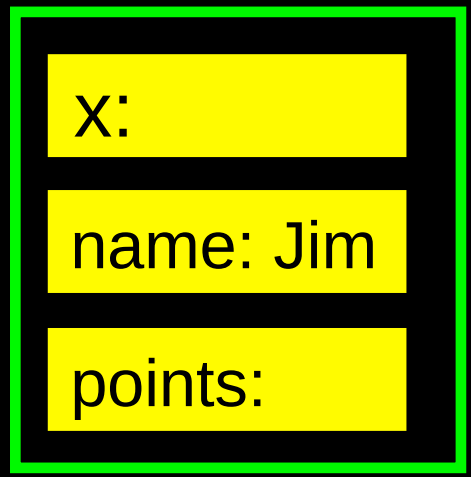
```
class PartyAnimal:
   x = 0
   name = ""
   def __init__(self, nam):
     self.name = nam
     print(self.name,"κατασκευάστηκε")

   def party(self) :
     self.x = self.x + 1
     print(self.name ,"πλήθος party", self.x)

class FootballFan(PartyAnimal):
   points = 0
   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"πόντοι",self.points)
```

```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```



```
class PartyAnimal:
   x = 0
   name = ""
   def __init__(self, nam):
     self.name = nam
     print(self.name,"κατασκευάστηκε")

   def party(self) :
     self.x = self.x + 1
     print(self.name ,"πλήθος party", self.x)

class FootballFan(PartyAnimal):
   points = 0
   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"πόντοι",self.points)
```

```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```



Ορισμοί

- Κλάση – Ένα πρότυπο
- Ιδιότητα – Μια μεταβλητή της κλάσης
- Μέθοδος - Μια λειτουργία μέσα σε μια κλάση
- Αντικείμενο – Ένα συγκεκριμένο στιγμιότυπο μιας κλάσης
- Κατασκευαστής – Κώδικας που εκτελείτε όταν ένα αντικείμενο κατασκευάζεται
- Κληρονομικότητα - Η δυνατότητα να επεκτείνετε μια κλάση για να δημιουργήσετε μια νέα κλάση.

Σύνοψη

- Ο αντικειμενοστραφής προγραμματισμός είναι μια πολύ δομημένη προσέγγιση στην επαναχρησιμοποίηση κώδικα
- Μπορούμε να ομαδοποιήσουμε δεδομένα και λειτουργίες μαζί και να δημιουργήσουμε πολλά ανεξάρτητα στιγμιότηπα μιας κλάσης

Ευχαριστίες / Συνεισφορές

Αρχική ανάπτυξη : Charles Severance, University of Michigan School of Information

Απόδοση στα Ελληνικά: Κιουρτίδου Δ. Κωνσταντία

… Εισαγάγετε νέους Μεταφραστές και άτομα που έχουν συνεισφέρει εδώ

Συνέχεια…

Πρόσθετες Πηγές Πληροφοριών

- “Snowman Cookie Cutter" by Didriks is licensed under CC https://www.flickr.com/photos/dinnerseries/23570475099

- Photo from the television program *Lassie*. Lassie watches as Jeff (Tommy Rettig) works on his bike is

  https://en.wikipedia.org/wiki/Lassie#/media/File:Lassie_and_Tommy_Rettig_1956.JPG