Python – 11. Αντικειμενοστρεφής Προγραμματισμός

Αντικείμενα

Στον αντικειμενοστρεφή προγραμματισμό, τα χαρακτηριστικά (attributes) ενός αντικειμένου ονομάζονται και ***ιδιότητες (properties)***.

Ένα αντικείμενο μπορεί να κάνει διάφορες ενέργειες οι οποίες ονομάζονται ***μέθοδοι (methods)***. Οι μέθοδοι επιτρέπουν στα αντικείμενα να κάνουν διάφορες ενέργειες, μέσω των οποίων, μπορούμε να ελέγχουμε τις ιδιότητες του αντικειμένου.

Ένα αντικείμενο δημιουργείται από μια ειδική μέθοδο ονομάζεται ***κατασκευαστής (constructor)***.

Με την κατασκευή γίνεται απόδοση αρχικών τιμών σε ιδιότητες του αντικειμένου. Η διαδικασία αυτή στον προγραμματισμό ονομάζεται ***αρχικοποίηση (initialization)***.

Ορισμένες από τις ιδιότητες ενός αντικείμενου μπορούν να καθοριστούν αργότερα από τη βασική κατασκευή του.

Αφού υπάρχει μια ειδική μέθοδος η οποία κατασκευάζει αντικείμενα θα πρέπει να υπάρχει και μια μέθοδος η οποία να τα καταστρέφει και να απελευθερώνει την αντίστοιχη μνήμη. Η μέθοδος αυτή ονομάζεται ***αποδομητής (destructor)***.

Κλάσεις

Τα πρότυπα που συγκεντρώνουν τα κοινά στοιχεία αντικειμένων, ονομάζονται ***κλάσεις (classes)***.

Ποιά η διαφορά μεταξύ κλάσεων και αντικειμένων;

Η κλάση τάρτα είναι μια συνταγή για τάρτες. Αν την έχεις, μπορείς να φτιάξεις τάρτες, όμως, όσο έχεις μόνο την συνταγή, δεν έχεις τάρτα για να τη φας.

Το αντικείμενο τάρτα είναι μια «πραγματική» τάρτα, μια υλοποίηση της συνταγής με συγκεκριμένα χαρακτηριστικά, ενώ μπορούμε να έχουμε πολλά διαφορετικά αντικείμενα τάρτες κατασκευασμένα με την ίδια συνταγή.

Κληρονομικότητα

Οι κοινές ιδιότητες και μέθοδοι ορίζονται μόνο στην **κλάση γονέα**.

Οι ***υποκλάσεις (κλάσεις παιδιά)*** αυτόματα, έχουν τις ίδιες ιδιότητες και μεθόδους, χωρίς να είναι απαραίτητο να τις ορίσουμε ξανά στις υποκλάσεις.

Ορισμός κλάσης στην Python

Αν θέλουμε να ορίσουμε την κλάση όχημα με ορισμένες ιδιότητες και μεθόδους, από αυτές που αναφέραμε παραπάνω, η σύνταξη είναι:

```
class Vehicle:
  def __init__(self,color,price,wheels,speed):
    self.color=color
    self.price=price
    self.wheels=wheels
    self.speed=speed
  def accelerate(self, amount) :
    self.speed += amount
    return self.speed
  def decelerate(self, amount) :
    self.speed -= amount
    return self.speed
```

Στιγμιότυπα

Προηγουμένως ορίσαμε την κλάση Vehicle η οποία περιγράφει την έννοια του οχήματος. Η ύπαρξη της κλάσης δε σημαίνει και την αυτόματη ύπαρξη αντικειμένου αυτής της κλάσης. Σημαίνει μόνο ότι υπάρχει απλώς η περιγραφή με την οποία μπορούμε να δημιουργήσουμε τέτοια αντικείμενα.

Μέθοδος αρχικοποίησης τιμών

Αποτελεί τον κατασκευαστή των αντικειμένων. Δίνει τιμές στις ιδιότητες του αντικειμένου που ορίζονται στις παραμέτρους της **__init__**.

Με το όνομα **__init__**, η γλώσσα αντιλαμβάνεται ότι είναι μια μέθοδος κατασκευαστής και έτσι καλείται αυτόματα με κάθε νέα δημιουργία αντικειμένου, τύπου Vehicle. Αποτελεί τον **κατασκευαστή των αντικειμένων**.

Για να δημιουργήσουμε ένα αντικείμενο πρέπει να καλέσουμε την κλάση Vehicle, με ορίσματα ή τιμές για κάθε μία από τις ιδιότητες του αντικειμένου που αντιστοιχούν στις παραμέτρους της μεθόδου __init__.

Στο αντικείμενο που δημιουργείται δίνουμε ένα όνομα, όπως για παράδειγμα:
`mybeetle=Vehicle('yellow',20000.0,4,80)`

Με το `mybeetle=Vehicle('yellow',20000.0,4,80)` δημιουργούμε ένα όχημα *κίτρινο*, με τιμή *20.000 ευρώ*, *4 τροχούς* και *ταχύτητα 80 km/h*. Το όνομά του είναι mybeetle και είναι ένα αντικείμενο, ανήκει στην κλάση Vehicle και έχει συγκεκριμένες τιμές για τις ιδιότητες του οχήματος, αυτές που ορίσαμε με την κλήση της. Ένα τέτοιο αντικείμενο ονομάζεται στιγμιότυπο της κλάσης.

Η χρήση της δεσμευμένης λέξης self

Η δεσμευμένη λέξη **self** εμφανίζεται ως πρώτη παράμετρος σε όλες τις μεθόδους της κλάσης.

Αυτή η παράμετρος επιτρέπει στη μέθοδο να αναφέρεται στο ίδιο το αντικείμενο και όχι σε ολόκληρη την κλάση, πράγμα που τουλάχιστον ισχύει στη γενική περίπτωση.

Τοποθετώντας στη self το όνομα του αντικειμένου, διασφαλίζουμε ότι μια μέθοδος που θα κληθεί μέσω ενός αντικειμένου, θα επιδράσει μόνο σε αυτό.

Συγκεκριμένα, με την κλήση **mybeetle.accelerate(10)** ενεργοποιούμε τις οδηγίες της μεθόδου accelerate για το αντικείμενο mybeetle.

Σε αυτή την περίπτωση η def accelerate(self, amount) αντιλαμβάνεται ως self το mybeetle.

Άρα, όταν γράφουμε def accelerate(self, amount) και μετά δίνουμε οδηγίες, αυτό που εννοούμε είναι *"ορίστε πώς θα εφαρμόσεις την accelerate, σε όποιο αντικείμενο της κλάσης Vehicle την καλέσει"*.

Και υλοποιούμε με αυτόν τον τρόπο την αύξηση ταχύτητας στο συγκεκριμένο αντικείμενο με το mybeetle.accelerate(10).

Η ιδιαιτερότητα της μεθόδου __init__

Θέλουμε να είμαστε βέβαιοι ότι έχουμε δημιουργήσει ένα αντικείμενο στο οποίο θα εφαρμόσουμε μεθόδους της κλάσης πάνω στις ιδιότητές του.

Η μέθοδος-συνάρτηση αρχικοποίησης __init__ κατασκευάζει αντικείμενα, για αυτό ονομάζεται κατασκευαστής.

Θα δημιουργήσουμε μια κλάση με όνομα **dog** που θα έχει τις ιδιότητες:
➢ **breed** (ράτσα),
➢ **size** (μέγεθος),
➢ **color** (χρώμα)
και τις μεθόδους
➢ **eat** (τρώω) και
➢ **bark** (γαυγίζω).

Στη συνέχεια θα δημιουργήσουμε δύο στιγμιότυπα της κλάσης, αρχικοποιώντας τις ιδιότητές τους και θα καλέσουμε τις μεθόδους τους:

```
class Dog:
  def __init__(self, breed, size, color):
    self.breed=breed
    self.size = size
    self.color = color
    print 'breed:',self.breed
    print 'size:', self.size
    print 'color:',self.color
  def eat(self, food):
    print 'I am eating ', food
  def bark(self):
    print 'I am barking '

Max=Dog("terrier","medium","brown")
Rocky=Dog("labrador","large","light brown")
Max.eat("bone")
Rocky.eat("meat")
```

Δραστηριότητα εμπέδωσης

```
class Car:
  def __init__(self, make):
    self.make=make
    self.speed = 60
  def speed_up(self, speed):
    self.speed = speed
    print "I am driving at a speed",
self.speed, "km/h"
  def turn(self):
    print "I am turning … "
```

1. Ποιος είναι ο κατασκευαστής (constructor) της κλάσης;

```
def __init__(self, make):
  self.make=make
  self.speed = 60
```

2. Να καταγράψετε τις ιδιότητες της κλάσης, τις μεθόδους της, καθώς και τις ενέργειες που πραγματοποιούν.

Ιδιότητες:
```
make
speed
```

Μέθοδοι:
```
speed_up (επιτάχυνε)
turn (στρίψε)
```

3. Να προσθέσετε τις ιδιότητες color και year που αντιπροσωπεύουν το χρώμα και το έτος κυκλοφορίας του αυτοκινήτου αντίστοιχα και να αρχικοποιούνται στον κατασκευαστή.

```
def __init__(self, make,color,year):
  self.make=make
  self.speed = 60
  self.color = color
  self.year = year
```

4. Να αλλάξετε τη μέθοδο turn, έτσι ώστε να δέχεται ως παράμετρο μια συμβολοσειρά που ορίζει αν το αυτοκίνητο θα στρίψει αριστερά ή δεξιά.

```
def turn(self, direction):
  self.direction = direction
  print "I am turning", self.direction
```

5. Να δημιουργήσετε τα παρακάτω στιγμιότυπα της κλάσης:
- αντικείμενο με όνομα convertible και μάρκα "BMW", χρώμα "μαύρο" και έτος κυκλοφορίας "2013".
- αντικείμενο με όνομα sedan και μάρκα "TOYOTA", χρώμα "κόκκινο" και έτος κυκλοφορίας "2009".

```
>>> convertible = Car("BMW","μαύρο","2013")
>>> sedan = Car("TOYOTA","κόκκινο","2009")
```

6. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο convertible να στρίψει δεξιά.

```
>>> convertible.turn("right")
I am turning right
```

7. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο sedan να τρέχει με 90 χιλ/ώρα.

```
>>> sedan.speed_up(90)
I am driving at a speed 90 km/h
```

Ιδιότητες και Μέθοδοι

Ένας **σκύλος** έχει τις ιδιότητες *φύλο*, *ράτσα*, *χρώμα* κ.λπ. και μπορεί να εκτελεί διάφορες ενέργειες, όπως να *τρέξει*, να *γαυγίσει*, να *φάει*.

Ένα **αυτοκίνητο** έχει τις ιδιότητες *χρώμα*, *μάρκα*, *μοντέλο* κ.λπ. και μπορεί να εκτελεί διάφορες ενέργειες όπως να *τρέχει*, να *σταματάει*, να *στρίβει*.

Μέθοδοι είναι οι ενέργειες που μπορεί να κάνει ένα αντικείμενο και δεν είναι τίποτε άλλο παρά συναρτήσεις που ανήκουν στο αντικείμενο.

Ορισμός ιδιοτήτων

Μπορεί να δημιουργηθεί μέσα στην κλάση:

```
class C:
  attr_class = "τιμή1"
```

Μπορεί να δημιουργηθεί και από το στιγμιότυπο (αντικείμενο) της κλάσης:

```
>>> x = C()
>>> x.attr_inst = "τιμή2"
```

Ορισμός μεθόδων

Οι μέθοδοι είναι σημασιολογικά ίδιες με τις συναρτήσεις, αλλά υπάρχουν δύο συντακτικές διαφορές:

- Οι μέθοδοι ορίζονται μέσα στον ορισμό μιας κλάσης, προκειμένου να γίνει σαφής η σχέση μεταξύ της κλάσης και της μεθόδου.
- Η σύνταξη για την κλήση μιας μεθόδου είναι διαφορετική από τη σύνταξη για την κλήση μιας συνάρτησης.

Ως **υπογραφή** ορίζεται το όνομα, τα ορίσματα και η τιμή επιστροφής της μεθόδου.

Οι μέθοδοι ορίζονται μέσα στην κλάση, χρησιμοποιώντας την ίδια σύνταξη που χρησιμοποιούμε, για να ορίσουμε μια συνάρτηση.

Αυτό με τη διαφορά ότι πάντα η πρώτη παράμετρος πρέπει να είναι το στιγμιότυπο (αντικείμενο) της κλάσης του οποίου η μέθοδος καλείται. Επειδή το αντικείμενο που καλεί δεν είναι προκαταβολικά γνωστό, χρησιμοποιούμε στη θέση του μια παράμετρο την οποία ονομάζουμε πάντα με το όνομα self.

Με αυτόν τον τρόπο, η Python γνωρίζει σε ποιο αντικείμενο ανήκει η μέθοδος που καλείται και ποιου αντικειμένου τις ιδιότητες αλλάζει.

Μέθοδοι που δεν αφορούν στιγμιότυπα

Μια ιδιαίτερη περίπτωση μεθόδου είναι οι λεγόμενες στατικές μέθοδοι. Στο παράδειγμα της κλάσης σκύλος (Dog) είχαμε τη μέθοδο bark().

```
class Dog:
  ...
  def bark(self):
    print "I am barking"
```

Ανεξάρτητα με το ποιο στιγμιότυπο καλεί τη συνάρτηση, το αποτέλεσμα θα είναι το ίδιο έτσι όπως την έχουμε ορίσει, μια δήλωση του εκάστοτε σκύλου ότι γαβγίζει. Σε αυτή την περίπτωση η παράμετρος self είναι περιττή.

Για να το κάνουμε αυτό σαφές έχουμε ένα ειδικό διακριτικό, το **@staticmethod**, που προηγείται της συνάρτησης. Έτσι ο ορισμός γίνεται:

```
class Dog:
  ...
  @staticmethod
  def bark():
    print "I am barking"
```

Το διακριτικό @staticmethod

Μια μέθοδος με το διακριτικό **@staticmethod** είναι στην πράξη μια κλασική συνάρτηση που όμως έχουμε τοποθετήσει μέσα σε μια κλάση.

Σε περίπτωση μιας υποκλάσης που κληρονομεί την κλάση, ο ορισμός δεν επηρεάζεται και μένει ακριβώς ο ίδιος που ορίστηκε στην κλάση. Βέβαια, εφόσον χρησιμοποιήσουμε το διακριτικό @staticmethod, δεν χρησιμοποιούμε το αντικείμενο ως παράμετρο.

Η μέθοδος κλάσης (@classmethod)

Μια μέθοδος κλάσης είναι κληρονομήσιμη και άρα, αν κληθεί από μια υποκλάση, θα αφορά αυτή.

Για τον ορισμό μιας μεθόδου κλάσης περνάμε ως πρώτη παράμετρο την κλάση, αντί του αντικειμένου, με τη δεσμευμένη λέξη cls που εννοεί την κλάση που την καλεί.

```
class Dog:
  no_inst = 0
  def __init__(self, breed, size, color):
    ...
    Dog.no_inst = Dog.no_inst + 1
  ...
  @classmethod
  def get_no_of_dogs(cls):
    return cls.no_inst
```