Κλάσεις και αντικείμενα

Προβλήματα του Διαδικασιακού Προγραμματισμού

- Υπάρχει διαχωρισμός των δεδομένων από τις διαδικασίες χειρισμού τους.
  - Τα δεδομένα έρχονται στο προσκήνιο όταν έχει αποφασιστεί ποιες θα είναι οι λειτουργίες που θα εφαρμοστούν σε αυτά.
- Λόγω αυτού του διαχωρισμού η προσθήκη μιας νέας λειτουργίας ή η ανάγκη άλλων τροποποιήσεων μπορεί να χρειαστούν μικρές έως και εκτεταμένες αλλαγές στη δομή του προγράμματος.
- Ελλοχεύει ο κίνδυνος να χρησιμοποιηθούν λάθος διαδικασίες σε λάθος δεδομένα, ιδιαίτερα αν αυτά είναι global.
- Οι παραπάνω δυσκολίες γίνονται μεγαλύτερες όσο μεγαλώνει το μέγεθος του προγράμματος.

Αντικειμενοστρεφής Προγραμματισμός

- Δημιουργούνται τύποι δεδομένων που συμπεριλαμβάνουν τα δεδομένα και τις ενέργειες πάνω σε αυτά σε μία οντότητα. Οι τύποι δεδομένων αυτοί περιγράφουν το υπό επίλυση πρόβλημα.
- Με αυτό τον τρόπο λύνονται τα προβλήματα του διαδικασιακού προγραμματισμού, ιδιαίτερα για μεγάλα προγράμματα όπου συμμετέχουν πολλοί προγραμματιστές.
- Τα αντικείμενα ταξινομούνται, ανάλογα με τα κοινά τους χαρακτηριστικά, σε κατηγορίες – κλάσεις.

Παράδειγμα Κλάσης

- Αυτοκίνητα
- Τα αυτοκίνητα έχουν βασικά χαρακτηριστικά και τρόπους χειρισμού.
- Χαρακτηριστικά: χρώμα, κατασκευαστής, μοντέλο, μέγιστη ταχύτητα, πλήθος θυρών, κλπ.
- Τρόποι χειρισμού: τρόπος εκκίνησης, τρόπος φρεναρίσματος, τρόπος αλλαγής πορείας, επιτάχυνση, κλπ.

```
class <Όνομα_κλάσης>:
        <Εντολές>
```

```
>>> class Car:
...         pass #εντολή χωρίς αποτέλεσμα
>>> acar=Car()
>>> another_car=Car()  #Δημιουργία αντικειμένων – στιγμιοτύπων της κλάσης Car
>>> type(acar)  #class '__main__.Car'
>>> acar # __main__.Car object at 0x……
```

Τι είναι ένα αντικείμενο;

- Είναι ένα κομμάτι λογισμικού που περιέχει μεταβλητές και μεθόδους.
- Ο αντικειμενοστρεφής προγραμματισμός επικεντρώνεται στα εξής:
  - Αφαίρεση (abstraction): δημιουργείται ένα απλοποιημένο μοντέλο για μια πολύπλοκη οντότητα του φυσικού κόσμου, που ονομάζεται κλάση, με τα απαραίτητα στοιχεία της οντότητας (**χαρακτηριστικά -μεταβλητές**) και τους τρόπους χειρισμού αυτών (**μέθοδοι**).
    - Με εξαίρεση τον τρόπο κλήσης, ότι ισχύει για τις συναρτήσεις ισχύει και για τις μεθόδους.
  - Ενθυλάκωση: Διαίρεση του κώδικα σε μία δημόσια διεπαφή και μια ιδιωτική υλοποίηση για αυτή.
  - Πολυμορφισμός: Η δυνατότητα της υπερφόρτωσης πρότυπων τελεστών έτσι ώστε να έχουν κατάλληλη συμπεριφορά με βάση το πλαίσιο λειτουργίας τους.
  - Κληρονομικότητα: Η δυνατότητα δημιουργίας υποκλάσεων που περιέχουν εξειδικεύσεις των γονέων τους.

Η μέθοδος __init__()

- Είναι μέθοδος με ειδική σημασία στην Python.
- Ανήκει στην κατηγορία των μεθόδων που ονομάζονται κατασκευαστές (constructors) και αναλαμβάνει τη δημιουργία ενός νέου αντικειμένου.
- Η Python την εκτελεί αυτόματα όποτε δημιουργηθεί ένα αντικείμενο με βάση την αντίστοιχη κλάση.
- Πριν και μετά το όνομα της μεθόδου τοποθετούνται δύο ενωτικά (_), τα οποία δείχνουν την ειδική χρήση της από την Python.
- Στην πραγματικότητα η δημιουργία ενός αντικειμένου, όπως με την εντολή acar=Car(), καλεί αυτόματα πρώτα την μέθοδο __new()__ για δημιουργία του ακατέργαστου αντικειμένου και μετά την __init__() για την αρχικοποίησή του.
- Η __init__() μπορεί να πάρει παραμέτρους.
- Δεν υπάρχουν destructors για απελευθέρωση μνήμης. Η Python απελευθερώνει τη μνήμη αυτόματα.

Παράδειγμα 2ο

```
>>> class Car: # ορισμός κλάσης με μία μέθοδο και ένα χαρακτηριστικό
...         def __init__(self,brand):
...                 self.brand=brand
>>> acar=Car("BMW")
>>> another_car=Car("Opel")
>>> acar.brand #BMW
>>> another_car.brand #Opel
```

Το προσδιοριστικό self

- Είναι ο δείκτης αναφοράς σε αυτό καθαυτό το αντικείμενο.
- Αυτό συμβαίνει, επειδή κατά τον ορισμό της κλάσης, δεν μπορεί να είναι γνωστό το όνομα που θα επιλεγεί για το αντικείμενο που θα προκύψει από την κλάση.
- Μπαίνει πάντα ως πρώτο όρισμα στην __init__(), αλλά και σε κάθε άλλη μέθοδο που ορίζεται εντός μιας κλάσης.
- Χρησιμοποιείται επίσης και για τον προσδιορισμό των χαρακτηριστικών – μεταβλητών που ορίζονται σε μια κλάση.

Χαρακτηριστικά

- Χαρακτηριστικά Δεδομένων: μεταβλητές οι οποίες ανήκουν στο αντικείμενο – στιγμιότυπο της κλάσης.
  - Για την προσπέλασή τους χρησιμοποιείται αποκλειστικά η self
  - Κάθε αντικείμενο που δημιουργείται από την κλάση έχειι το δικό του αντίγραφο των χαρακτηριστικών αυτής.
  - Τα χαρακτηριστικά δεδομένων μπορούν να αρχικοποιηθούν στην __init__() και να μεταβάλλονται μέσω μεθόδων.
  - Τα χαρακτηριστικά των δεδομένων μπορούν να αποδοθούν και δυναμικά, δηλαδή, αφού πρώτα δημιουργηθεί το αντικείμενο – στιγμιότυπο της κλάσης και ενώ δεν έχουν συμπεριληφθεί στη δήλωση της κλάσης.
- Χαρακτηριστικά Κλάσης: ορίζονται σε επίπεδο κλάσης και είναι κοινά για όλα τα αντικείμενα – στιγμιότυπα.

Χαρακτηριστικά Δεδομένων

```
>>> class Car:
...         def __init__(self):
...             self.speed=0
...         def incr_speed(self, increment)
...             self.speed+=increment
>>> acar=Car()
>>> acar.speed
0
>>> acar.incr_speed(10)
10
```

Χαρακτηριστικά κλάσης

```
>>> class Car:
...         speed=0
...         def incr_speed(self, increment)
...                 self.__class__.speed+=increment
>>> acar=Car()
>>> acar.__class__.speed
 0
>>> acar.incr_speed(10)
>>> acar.__class__.speed
10
```

Ενσωματωμένα Χαρακτηριστικά

```
>>> class Car:
        '''Απλή κλάση αυτοκινήτου'''
...     def __init__(self,brand):
...         self.brand=brand
>>> acar=Car("BMW")
```

```
>>> acar.__doc__ #συμβολοσειρά τεκμηρίωσης
'Απλή κλάση αυτοκινήτου'
>>> acar.__module__ #δομοστοιχείο ορισμού κλάσης
'__main__'
>>> print(acar.__dict__) #το λεξικό του ονοματοχώρου της κλάσης
{'brand':'BMW'}
>>> print(acar.__class__)
<class '__main__.Car'>
```

Ιδιωτικά Χαρακτηριστικά

- Αν το όνομα ενός χαρακτηριστικού αρχίζει με __ (δύο ενωτικά) και δεν τελειώνει με __, το χαρακτηριστικό θεωρείται ιδιωτικό (private).
- Αυτό σημαίνει ότι το χαρακτηριστικό δεν μπορεί να χρησιμοποιηθεί εξωτερικά του αντικειμένου με τη σύνταξη αντικείμενο.χαρακτηριστικό.

Μέθοδοι

- Είναι συναρτήσεις που χρησιμοποιούνται για ενέργειες πάνω στα χαρακτηριστικά ενός αντικειμένου.
- Ορίζονται εντός του σώματος μιας κλάσης.
- Το πρώτο όρισμα είναι υποχρεωτικά το self.
- Κατά την κλήση μιας μεθόδου δεν περιλαμβάνεται το self.
- Οι τοπικές μεταβλητές των μεθόδων είναι ορατές μόνο εντός του σώματος της μεθόδου που ορίζονται.
- Αν το όνομα μιας μεθόδου αρχίζει με __ (δύο ενωτικά) και δεν τελειώνει με __, η μέθοδος θεωρείται ιδιωτική (private).
  - Αυτό σημαίνει ότι η μέθοδος δεν μπορεί να χρησιμοποιηθεί εξωτερικά του αντικειμένου με τη σύνταξη αντικείμενο.μέθοδος.

Κληρονομικότητα

- Χρησιμοποιείται για την υλοποίηση ιεραρχικών δομών.
- Είναι η δημιουργία νέων κλάσεων με βάση κλάσεις που έχουν ήδη δημιουργηθεί.
- Η νέα κλάση ονομάζεται παράγωγη, ενώ η αρχική ονομάζεται βασική ή γονική κλάση.
- Η παράγωγη κλάση είναι εξειδίκευση της βασικής.
- Η παράγωγη κλάση κληρονομεί τα χαρακτηριστικά και τις μεθόδους της βασικής.
- Η παράγωγη κλάση μπορεί να επεκτείνει υφιστάμενες μεθόδους της βασικής ή να τροποποιήσει τη συμπεριφορά μιας βασικής κλάσης.
- Η παράγωγη κλάση μπορεί να επεκτείνει τη λειτουργικότητα της βασικής κλάσης με προσθήκη επιπλέον χαρακτηριστικών ή/και μεθόδων.

Παράδειγμα 3ο

```
>>> class Animal:
        def __init___(self):
            print("Δημιουργία ζώου")
        def who_am_i(self):
            print("Ζώο")
        def eat(self):
            print("Τρέφομαι")

>>> class Cat(Animal):
            def __init__(self):
                Animal.__init__(self)
                print("Δημιουργία γάτας")
            def who_am_i(self):
                print("Γάτα")
            def talk(self):
                print("Μιάου")
```

Πολυμορφισμός

- Είναι η χρήση ενός τελεστή ή μιας μεθόδου με διαφορετικούς τρόπους για διαφορετικού τύπου δεδομένα.
- Αν και δεν είναι απαραίτητο, χρησιμοποιείται σε συνδυασμό με την κληρονομικότητα.

```
>>> class Animal:
        def talk(self):
            return self.say()
        def say(self):
            return

>>> class Cat(Animal):
            def say(self):
                return 'Μιάου'
>>> class Dog(Animal):
            def say(self):
                return 'Γαβ Γαβ'
>>> acat=Cat()
>>> adog=Dog()
>>> acat.talk() #'Μιάου'
>>> adog.talk() #'Γαβ Γαβ'
```

Παράδειγμα 4ο

```
class atom(object):
   def __init___(self,atno,x,y,z):
       self.atno = atno
       self.position = (x,y,z)
   def symbol(self):    # a class method
       return Atno_to_Symbol[atno]
   def___repr__(self): # overloads printing
        return '%d %10.4f %10.4f %10.4f' %  (self.atno,
        self.position[0], self.position[1], self.position[2])
```

```
>>> at = atom(6,0.0,1.0,2.0)
>>> print at
6  0.0000  1.0000 2.0000
>>> at.symbol()
'C'
```

Χρησιμοποίηση της atom για τη δημιουργία της κλάσης Molecule

```
class molecule:
  def __init__(self,name='Generic'):
        self.name = name
        self.atomlist = []
  def addatom(self,atom):
        self.atomlist.append(atom)
  def___repr__(self):
        str = 'This is a molecule named %s\n' % self.name
        str = str+'It has %d atoms\n' % len(self.atomlist)
        for atom in self.atomlist:
              str = str + `atom` + '\n'
        return str
```

Χρήση της κλάσης Molecule

```
>>> mol = molecule('Water')
>>> at = atom(8,0.,0.,0.)
>>> mol.addatom(at)
>>> mol.addatom(atom(1,0.,0.,1.))
>>> mol.addatom(atom(1,0.,1.,0.))
>>> print mol
This is a molecule named Water
It has 3 atoms
8   0.000 0.000 0.000
1   0.000 0.000 1.000
1   0.000 1.000 0.000
```

- Επαναχρησιμοποίηση κώδικα: πληκτρολογείται απλά ο κώδικας που τυπώνει ένα atom μία φορά. Αυτό σημαίνει ότι αν αλλάξει η προδιαγραφή της κλάσης atom τότε απαιτείται ενημέρωση μιας θέσης μόνο.

Κληρονομικότητα

```
class qm_molecule(molecule):
   def addbasis(self):
     self.basis = []
     for atom in self.atomlist:
       self.basis = add_bf(atom,self.basis)
```

- __init__, __repr__, και __addatom__ ανήκουν στη γονική κλάση (molecule).
- Δημιουργείται μια νέα συνάρτηση η __addbasis__ για την προσθήκη ενός basis συνόλου.
- Επαναχρησιμοποίηση κώδικα
  - Οι βασικές συναρτήσεις δεν απαιτείται να επαναπληκτρολογηθούν, παρά μόνο να κληρονομηθούν.
  - Όταν υπάρχει αλλαγή προδιαγραφής αντικειμένου είναι ελάχιστη η αλλαγή που γίνεται.

Υπερφόρτωση τελεστή

```
class qm_molecule(molecule):
   def__repr__(self):
      str = 'QM Rules!\n'
      for atom in self.atomlist:
         str = str + `atom` + '\n'
      return str
```

- Κληρονομούνται τώρα μόνο οι __init__ και __addatom__ από τη γονική κλάση.
- Ορίζουμε μια νέα έκδοση της __repr__ μόνο για την qm_molecule.

Προσθήκες σε γονικές συναρτήσεις

- Κάποιες φορές απαιτείται να επεκταθούν, αντί να αντικατασταθούν οι γονικές συναρτήσεις

```
class qm_molecule(molecule):
  def___init__(self,name="Generic",basis="6-31G**"):
    self.basis = basis
    super(qm_molecule, self).__init__(name)
```