Αντικειμενοστρεφής Προγραμματισμός

Εισαγωγή στις βασικές έννοιες της αντικειμενοστρεφούς Python.

Εισαγωγή

- Μέχρι τώρα έχουμε δει δύο παραδείγματα προγραμματισμού:
  - Προστακτικό (*imperative*).
  - Συναρτησιακός (*functional*).

- Ως τρίτη μορφή ορίζουμε τον **αντικειμενοστρεφή προγραμματισμό.**

- Αντικείμενα δημιουργούνται χρησιμοποιώντας κλάσεις, οι οποίες είναι το επίκεντρο του αντικειμενοστρεφούς προγραμματισμού.

- Κλάση αποτελεί μια δομή δεδομένων που περιβάλλει, ως αυτόνομη οντότητα με ταυτότητα και τα δικά της χαρακτηριστικά, τα διάφορα δεδομένα προς χειρισμό.

Κλάσεις (classes)

- Μια κλάση περιγράφει τί θα είναι το αντικείμενο, ενώ αποτελεί εντελώς ξεχωριστό κομμάτι του κώδικα.
- Ουσιαστικά, μια κλάση μπορεί να θεωρηθεί ως σκελετός ή σχέδιο για την δημιουργία ποικίλων, διαφορετικών, αντικειμένων.
- Για να δημιουργήσουμε μια κλάση χρησιμοποιούμε την ειδική λέξη κλειδί **class** και σε εσοχή τις ιδιότητες και τις διάφορες μεθόδους.
- Μεθόδοι αποτελούν συναρτήσεις, οι οποίες ανήκουν αποκλειστικά στην κλάση, στης οποία το block γράφηκαν.
  - *Μπορούμε να αποφανθούμε πως οι μεθόδοι μια κλάσης μπορούν να χρησιμοποιηθούν και από αντικείμενα της ίδιας κλάσης;*

Κλάσεις (classes) (συνέχεια)

- Τα αντικείμενα μπορούν να ονομαστούν και στιγμιότυπα κλάσεων (*class instances*).

- Μια κλάση ορίζει τις ιδιότητες ενός αντικειμένου, οι οποίες διαφέρουν ανάλογα το αντικείμενο. Δύο αντικείμενα μπορεί να έχουν τις ίδιες ιδιότητες ως είδος, αλλά διαφορετικές τιμές στην κάθε μια.

- **Παράδειγμα**: Έστω η κλάση *Άνθρωπος*. Ο καθένας από εμάς είναι ένα αντικείμενο, που ανήκει στην παραπάνω κλάση. Ο καθένας από εμάς έχει όνομα, επίθετο, φύλλο, ύψος, επάγγελμα και πολλές άλλες **ιδιότητες**. Επίσης ο καθένας από εμάς μπορεί να εκτελέσει βασικές λειτουργίες (κίνηση, γραφή, ανάγνωση) ή αλλιώς **μεθόδους**.

Δημιουργία κλάσεων

- Ως παράδειγμα θα δημιουργήσουμε μια κλάση Cat, η οποία έχει δύο ιδιότητες: color και legs. Στην συνέχεια η κλάση θα χρησιμοποιηθεί για την δημιουργία τριών διαφορετικών αντικειμένων.

```
1  class Cat:
2      def __init__(self, color, legs):
3          self.color = color
4          self.legs = legs
5
6  felix = Cat("ginger", 4)
7  rover = Cat("dog-colored", 4)
8  stumpy = Cat("brown", 3)
9
```

Μέθοδος __init__

- Η πιό σημαντική μέθοδος της κλάσης.
- Η __init__ καλείται όταν δημιουργείται ένα αντικείμενο, χρησιμοποιώντας το όνομα της μεθόδου ως συνάρτηση!
  - *Με βάση αυτήν την περιγραφή, περιγράψτε το προηγούμενο κομμάτι κώδικα.*
- Όλες οι μέθοδοι πρέπει να περιέχουν το **self** ως πρώτο όρισμα κατά την δήλωση τους. Η Python τοποθετεί την παράμετρο αυτόματα, οπότε δεν χρειάζεται να συμπεριληφθεί κατά την κλήση των μεθόδων.
- Το **self** αναφέρεται στο στιγμιότυπο που καλεί την μέθοδο.

Μέθοδος __init__ (συνέχεια)

- Τα στιγμιότυπα των κλάσεων περιέχουν, όπως προαναφέρθηκε, ιδιότητες (*attributes*), οι οποίες είναι δεδομένα, άμεσα σχετικά με αυτά.
  - *Στο παράδειγμα με τις γάτες, μπορείτε να διακρίνετε τις ιδιότητες;*

- Η πρόσβαση στις ιδιότητες γίνεται μέσω του τελεστή τελεία και από δίπλα το όνομα της ιδιότητας.
- **Παράδειγμα**:

```
1  class Cat:
2      def __init__(self, color, legs):
3          self.color = color
4          self.legs = legs
5
6  felix = Cat("ginger", 4)
7  print(felix.color)
8
```

Μέθοδος __init__ (συνέχεια)

- Στο προηγούμενο παράδειγμα, η __init__ πήρε δύο ορίσματα και τα ανέθεσε στις ιδιότητες του αντικειμένου.

- Η μέθοδος __init__ ονομάζεται κατασκευαστής κλάσης (*class constructor*).
- **Παράδειγμα #2:**

```
1  class Student:
2      def __init__(self, name):
3          self.name = name
4
5  student1 = Student("Nikos")
6  print(student1.name)
7
```

Μέθοδοι

- Οι κλάσεις μπορούν να περιέχουν διάφορες μεθόδους για την αύξηση της λειτουργικότητας τους.
- Πρέπει να θυμόμαστε πως όλες οι μέθοδοι πρέπει να έχουν ως πρώτο όρισμα το **self**!
- Η κλήση των μεθόδων γίνεται όπως ακριβώς η πρόσβαση σε ιδιότητες των αντικειμένων.
  - *Τι παραδείγματα κλάσεων και ιδιοτήτων μπορείτε να δώσετε;*

Μέθοδοι (συνέχεια)

- Παράδειγμα με την δημιουργία και την χρήση μεθόδων.

```
1  class Dog:
2      def __init__(self, name, color):
3          self.name = name
4          self.color = color
5
6      def bark(self):
7          print("Woof!")
8
9  azor = Dog("Azor", "brown")
10 print(azor.name)
11 print(azor.color)
12 azor.bark()
13
```

Μέθοδοι (συνέχεια)

- Οι κλάσεις μπορούν να έχουν και ιδιότητες κλάσης (*class attributes*), οι οποίες δημιουργούνται αναθέτοντας τιμές σε μεταβλητές μέσα στο σώμα της κλάσης. Αυτές οι ιδιότητες μπορούν να προσπελαστούν είτε από τα στιγμιότυπα, είτε από την ίδια την κλάση.

```
1  class Dog:
2      legs = 4
3
4      def __init__(self, name, color):
5          self.name = name
6          self.color = color
7
8      def bark(self):
9          print("Woof!")
10
11 rex = Dog("Rex", "Brown")
12 print(rex.legs)      # for Rex
13 print(Dog.legs)      # for any dog
14
```

Σφάλματα

- Η προσπάθεια προσπέλασης κάποιας ιδιότητας που δεν ανήκει στην κλάση προκαλείται **AttributeError**.

- Σφάλμα επίσης εμφανίζεται όταν καλούμε μια μέθοδο, η οποία δεν έχει οριστεί στην κλάση μας.

```
1  class square:
2      def __init__(self, width, height):
3          self.width = width
4          self.height = height
5
6  chess = square(50, 50)
7  print(chess.color)
8  chess.game()
9
```

Κληρονομικότητα

Βασικές έννοιες, κλάσεις-πρόγονοι και κλάσεις-απόγονοι

Εισαγωγή και χρήση

- Η κληρονομικότητα αποτελεί ένα τρόπο να μοιραζόμαστε ιδιότητες και μεθόδους μεταξύ κλάσεων.

- Ας φανταστούμε, τα σκυλιά, οι γάτες και τα κουνέλια έχουν τέσσερα πόδια και χρώμα. Και τα τρία είναι ζώα, οπότε μπορώ να πω πως είναι εξειδικεύσεις της ανώτερης κλάσης (*superclass*) ζώα.

- Για να ορίσουμε την μια κλάση ως απόγονο μιας κλάσης, γράφουμε στην παρένθεση δίπλα από το όνομα του απογόνου το όνομα του προγόνου.

Παράδειγμα

```
1  class Animal:
2      def __init__(self, color, legs):
3          self.color = color
4          self.legs = legs
5
6  class Cat(Animal):
7      def purr(self):
8          print("Purr..")
9
10 class Dog(Animal):
11     def bark(self):
12         print("Woof!")
13
14 fido = Dog("Fido", "brown")
15 fido.bark()
```

Εισαγωγή και χρήση (συνέχεια)

- Η κλάση η οποία κληρονομεί κάποια άλλη ονομάζεται υποκλάση (*subclass*) ή απόγονος (*descendant*).
- Η κλάση η οποία κληρονομείται ονομάζεται ανώτερη κλάση ή υπερκλάση (*superclass*) ή πρόγονος (*ancestor).*
- Αν μια κλάση που κληρονομεί περιέχει κοινές ιδιότητες και μεθόδους, επικρατούν τα στοιχεία του απογόνου.

Κληρονομικότητα (συνέχεια)

```
1  class Wolf:
2      def __init__(self, name, color):
3          self.name = name
4          self.color = color
5
6      def bark(self):
7          print("Grr...")
8
9  class Dog(Wolf):
10     def bark(self):
11         print("Woof!")
12
13 husky = Dog("Max", "grey")
14 husky.bark()
15
```

Πηγές

Το υλικό των διαφάνειων βασίστηκε στις σειρές διαδικτυακών μαθημάτων των ιστοσελίδων:
www.sololearn.com & www.tutorialspoint.com.

Επίσης υλικό για τους ορισμούς αντλίθηκε από την ιστοσελίδα:
www.el.wikipedia.org