Ελληνικό Μεσογειακό Πανεπιστήμιο
Σχολή Μηχανικών / Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

HMY01K06
Επιστημονικός Προγραμματισμός με Python

6+1 ερωτήσεις επανάληψης για τη Διάλεξη 12

Αντικειμενοστρεφής Προγραμματισμός

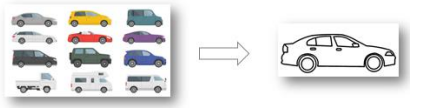
Κλάσεις , Αντικείμενα, Κληρονομικότητα, Πολυμορφισμός

Φθινόπωρο 2025

Ερώτηση 1/7

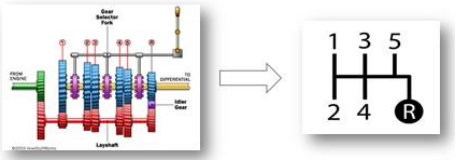
Ποια είναι τα τέσσερα ειδικά γνωρίσματα της αντικειμενοστρέφειας;

Αφαίρεση (abstraction): Μια κλάση αποτελεί ένα *απλοποιημένο μοντέλο-άποψη* μιας *πολύπλοκης οντότητας* του φυσικού κόσμου, που διατηρεί μόνο τα *στοιχεία που κρίνονται σημαντικά* για την συγκεκριμένη μοντελοποίηση

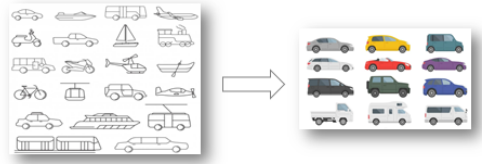


Ενθυλάκωση (encapsulation). Οι *λεπτομέρειες υλοποίησης* μιας κλάσης *αποκρύπτονται* από το περιβάλλον χρήσης της

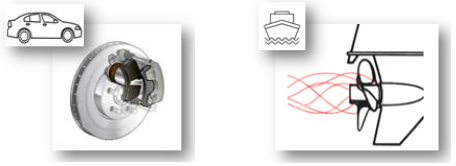
Η κλάση έχει μια *δημόσια διεπαφή* η οποία περιγράφει τις *ιδιότητες και μεθόδους χρήσης της* ενώ η *εσωτερική δομή* της παραμένει *ιδιωτική*. Έτσι προστατεύονται τα εσωτερικά δεδομένα από ανεπιθύμητη προσπέλαση / μεταβολή



Κληρονομικότητα και σύνθεση (inheritance and composition). Όταν δημιουργούνται *νέες κλάσεις* που βασίζονται και εξειδικεύουν προϋπάρχουσες, τότε αυτές *κληρονομούν δομικά στοιχεία (ιδιότητες και μεθόδους)* των αρχικών κλάσεων



Πολυμορφισμός (polymorphism) - Προκύπτει όταν *διαφορετικές κλάσεις* που έχουν δημιουργηθεί κληρονομώντας στοιχεία από την *ίδια βασική κλάση* έχουν κάποιες μεθόδους με *ίδιο όνομα* και *ίδια λειτουργικότητα* υλοποιημένες με *διαφορετικό τρόπο* ανάλογα με την περίσταση



Ερώτηση 2/7

Ποιά είναι τα δυο βασικά δομικά στοιχεία των κλάσεων;

Ιδιότητες ή χαρακτηριστικά / (attributes): Χρησιμοποιούνται για την περιγραφή ιδιοτήτων και καταστάσεων μιας κλάσης

```
>>> f.name
'C:/Users/MK/my_friends.txt'
>>> f.mode
'w'
>>> f.closed
True
```

Τα παραδείγματα αναφέρονται στην κλάση της Python `_io.TextIOWrapper` η οποία περιλαμβάνει αρχεία κειμένου

Μέθοδοι χειρισμού (methods). *Συναρτήσεις* που ορίζονται μέσα στην κλάση και περιγράφουν συγκεκριμένες *ενέργειες* που ορίζονται *πάνω στα χαρακτηριστικά της*. Αποτελούν ουσιαστικό στοιχείο της ιδέας της *ενθυλάκωσης* (*encapsulation*) στο αντικειμενοστρεφές υπόδειγμα προγραμματισμού. Ξεχωρίζουν από τις ιδιότητες επειδή ακολουθούνται πάντα από παρενθέσεις

```
>>> f.write(names)
>>> f.close()
```

Οι *λεπτομερείς ενέργειες* που πραγματοποιούνται για την αποθήκευση δεδομένων στο αρχείο, είναι *κωδικοποιημένες στην συνάρτηση* (μέθοδο) *write*. Ο χρήστης εντούτοις δεν ενδιαφέρεται για αυτές τις λεπτομέρειες. Απλώς καλεί την *write*, η οποία και *αναλαμβάνει την αποθήκευση των δεδομένων* όπως εκείνη κρίνει καλύτερα (encapsulation of complexity)

Ερώτηση 3/7

Δώστε ένα παράδειγμα δημιουργίας και αρχικοποίησης μιας απλής κλάσης

```
>>> class Cat:
        gene = 'Felis_catus'
        def __init__(self, p1, p2, p3):
            self.name = p1
            self.color = p2
            self.age = p3
>>> my_cat = Cat('Molly','white',2)
>>> your_cat = Cat('Jimmy','red',3)
```

Γενικές ιδιότητες κλάσης ***κοινές για όλα τα στιγμιότυπα***

Ειδικές ιδιότητες δεδομένων ***διαφορετικές για κάθε στιγμιότυπο***
Ορίζονται μέσα στην μέθοδο ***__init__***

Ερώτηση 4/7

Σε τι διαφέρουν οι *γενικές* από τις *ειδικές* ιδιότητες μιας κλάσης;

- *Γενικές ιδιότητες της κλάσης* (class attributes): Ιδιότητες οι οποίες ανήκουν *στην κλάση* και όχι στο κάθε αντικείμενο-στιγμιότυπο της κλάσης που δημιουργείται
  - Ορίζονται στον κορμό της κλάσης ***χωρίς*** το πρόθεμα `self.` ***έξω*** από την συνάρτηση `__init__` και ***έξω*** από οποιαδήποτε συνάρτηση-μέθοδο της κλάσης
  - Θεωρούνται *κοινές ιδιότητες* για *όλα τα στιγμιότυπα* της κλάσης
  - Αν αλλάξουν τιμή *στην ίδια την κλάση*, τότε η αλλαγή αυτή επηρεάζει *όλα τα αντικείμενα -στιγμιότυπα της κλάσης*, ενώ αν αλλάξουν τιμή *μόνο σε ένα αντικείμενο-στιγμιότυπο*, η αλλαγή αυτή *δεν επηρεάζει τα υπόλοιπα στιγμιότυπα*
- *Ειδικές ιδιότητες των στιγμιότυπων της κλάσης* (data attributes): Ιδιότητες οι οποίες ανήκουν μόνο στο αντικείμενο-στιγμιότυπο της κλάσης που θα δημιουργηθεί
  - *Ορίζονται* και *αρχικοποιούνται* μέσα στην ειδική συνάρτηση `__init__(self,..)` της κλάσης
  - Αν αλλάξουν τιμή σε ένα αντικείμενο-στιγμιότυπο, η αλλαγή *δεν επηρεάζει τα υπόλοιπα στιγμιότυπα*
  - *Μια καλή πρακτική* είναι οι ειδικές ιδιότητες των στιγμιότυπων να μεταβάλλονται *μόνο μέσω ειδικών μεθόδων (συναρτήσεων)* που ορίζονται μέσα στην κλάση. *(Η Python συνιστά μεν, αλλά δεν επιβάλλει αυτή την πρακτική σε αντίθεση με άλλες γλώσσες όπως Java και C++)*

Ερώτηση 5/7

Ποιές είναι οι τρεις κατηγορίες μεθόδων/συναρτήσεων που μπορούμε να συναντήσουμε σε μια κλάση;

1. **Κανονικές** □□□□

   Όλες οι συναρτήσεις που ορίζονται μέσα σε μια κλάση, *εφόσον το όνομά τους δεν ξεκινάει με διπλό underscore (__)*. Οι συναρτήσεις αυτές αποτελούν τις *μεθόδους της κλάσης* και χρησιμοποιούνται από τα στιγμιότυπα της κλάσης για να μεταβάλλουν τα χαρακτηριστικά τους

2. **Ιδιωτικές** __□□□□

   Συναρτήσεις που ορίζονται μέσα σε μια κλάση, των οποίων το όνομα *ξεκινάει με διπλό underscore (__)* αλλά *δεν τερματίζει με διπλό underscore (__)* ονομάζονται ιδιωτικές και μπορούν να καλούνται ***μόνο από άλλες συναρτήσεις στο εσωτερικό της κλάσης*** *(δεν αναγνωρίζονται έξω από την κλάση - στην ουσία πρόκειται για τοπικές συναρτήσεις)*

3. **Ειδικές** __□□□□__

   Συναρτήσεις που ορίζονται μέσα σε μια κλάση και το όνομά τους ταυτίζεται με κάποιο από τα ειδικά χαρακτηριστικά, εκείνα δηλαδή που *ξεκινάνε και τερματίζουν με διπλό underscore (__)* Οι συναρτήσεις αυτές αποτελούν *τις ειδικές μεθόδους της κλάσης* και χρησιμοποιούνται για να *επανακαθορίσουν / εξειδικεύσουν* την τιμή κάποιων *ειδικών χαρακτηριστικών* σύμφωνα με τις απαιτήσεις του προγραμματιστή της κλάσης (πχ `__init__`)

Ερώτηση 6/7

Δώστε ένα παράδειγμα πολυμορφισμού θυγατρικών κλάσεων
που προκύπτουν από την ίδια γονεϊκή κλάση

```
class Animal:
    # Είδος
    # Γένος
    # ...

    def speak():
        print("I don't know what to say...")
```

```
class Cat(Animal): #Θυγατρική κλάσης της Animal
    def speak(): #Νέα μέθοδος speak. αντικαθιστά την γονεϊκή
        print("Meow!")

class Dog(Animal):#Θυγατρική κλάσης της Animal
    def speak(): #Νέα μέθοδος speak. αντικαθιστά την γονεϊκή
        print("Woof!")
```

```
>>> just_an_animal = Animal()
>>> my_dog = Dog()
>>> my_cat = Cat()

>>> my_dog.speak()
Woof!

>>> my_cat.speak()
Meow!

>>> just_an_animal.speak()
I don't know what to say...
```

Bonus Ερώτηση 7/7

Τι ακριβώς είναι οι μετα-κλάσεις (*metaclasses*) στην Python;

Είναι *κλάσεις κλάσεων*, δηλαδή κλάσεις των οποίων τα *στιγμιότυπα* είναι με τη σειρά τους *άλλες κλάσεις.* Το απλούστερο παράδειγμα μετα-κλάσης στην Python είναι η κλάση `type` με όλες τις υπόλοιπες κλάσεις της γλώσσας *(συμπεριλαμβανομένης και της ίδιας της type!)* να αποτελούν στιγμιότυπά της

```
F = 3.25
print(F)
print(F.__class__) #ισοδύναμο του type(F)
print(F.__class__.__class__) # type(type(F))
```

```
3.25
<class 'float'>
<class 'type'>
```

η κλάση float είναι στιγμιότυπο της κλάσης type

```
s = "Hello"
print(s)
print(s.__class__) #ισοδύναμο του type(s)
print(s.__class__.__class__)
```

```
Hello
<class 'str'>
<class 'type'>
```

η κλάση string είναι στιγμιότυπο της κλάσης type

```
L = [1,2,3]
print(L)
print(L.__class__) #ισοδύναμο του type(L)
print(L.__class__.__class__)
```

```
[1, 2, 3]
<class 'list'>
<class 'type'>
```

η κλάση list είναι στιγμιότυπο της κλάσης type

```
print(F.__class__.__class__.__class__)
```

```
<class 'type'>
```