

week 2

κλάσεις και κληρονομικότητα

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Την εβδομάδα αυτή θα δούμε έννοιες του αντικειμενοστραφούς προγραμματισμού, όπως η κληρονομικότητα και θα αναπτύξουμε μια εφαρμογή με κλάσεις

L2.1

Γνωρίσματα κλάσεων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην διάλεξη L2.1 θα εξετάσουμε τα γνωρίσματα των κλάσεων και τρόπους να αναπαραστήσουμε αντικείμενα

week 2

κλάσεις και κληρονομικότητα

week 2

L2.1 Γνωρίσματα κλάσεων


L2.2 Παράδειγμα: οι κλάσεις Card-Deck

L2.3 Κληρονομικότητα κλάσεων

L2.3 Παράδειγμα: το παιχνίδι 31

Μάθημα L2.1

Ειδικές μέθοδοι

Προγραμματίζω
με την python 

- V2.1.1 Γνωρίσματα κλάσεων
- V2.1.2 Παράδειγμα: η κλάση Point
- V2.1.3 Οι μέθοδοι `__str__` και `__repr__`
- V2.1.4 Διαγραφή αντικειμένων και γνωρισμάτων

V2.1.1

γνωρίσματα κλάσεων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Πώς θα επιτρέπαμε στην κλάση Employee να γνωρίζει τη λίστα των εργαζόμενων;

Θα μεταφέρουμε τη λίστα των εργαζομένων `the_employees` μέσα στην κλάση, ως **γνώρισμα κλάσης (class attribute)**. Τα γνωρίσματα αυτά ορίζονται στο επίπεδο της κλάσης και είναι γνωστά σε όλα τα αντικείμενα. Για παράδειγμα η λίστα αυτή θα πρέπει να ενημερώνεται από το δημιουργό αντικειμένων για κάθε νέο αντικείμενο.

Εν γένει γίνεται αναφορά στα γνωρίσματα κλάσης με σημειογραφία **<ΌνομαΚλάσης>.<ΌνομαΓνωρίσματος>**

νέα έκδοση της κλάσης Employee

Προγραμματίζω
με την python



employee example

class Employee():

''' Ο εργαζόμενος σε μια επιχείρηση '''

the_employees = []

def __init__(self, name, salary):

self.name = name

self.salary = salary

Employee.the_employees.append(self)

V2.1.2

παράδειγμα: η κλάση **Point**

Νίκος Αβούρης, Πανεπιστήμιο Πατρών


Παράδειγμα 2. η κλάση **Point**

Να ορίσετε μια κλάση **Point** που περιγράφει σημεία (x,y) στο καρτεσιανό επίπεδο. Ο δημιουργός αντικειμένων της κλάσης δέχεται ως όρισμα τη θέση του σημείου (x,y) , όπου x,y ακέραιοι.

Τα αντικείμενα της κλάσης θα πρέπει να έχουν μια μέθοδο **distance(p)** που λαμβάνει ως όρισμα ένα άλλο σημείο p και υπολογίζει την απόσταση του σημείου από το p .

Η κλάση **Point** περιλαμβάνει ως γνώρισμα κλάσης μια λίστα που περιέχει τα σημεία που έχουν δημιουργηθεί.

Παράδειγμα 2. η εφαρμογή **Point**


Προγραμματίζω
με την python 

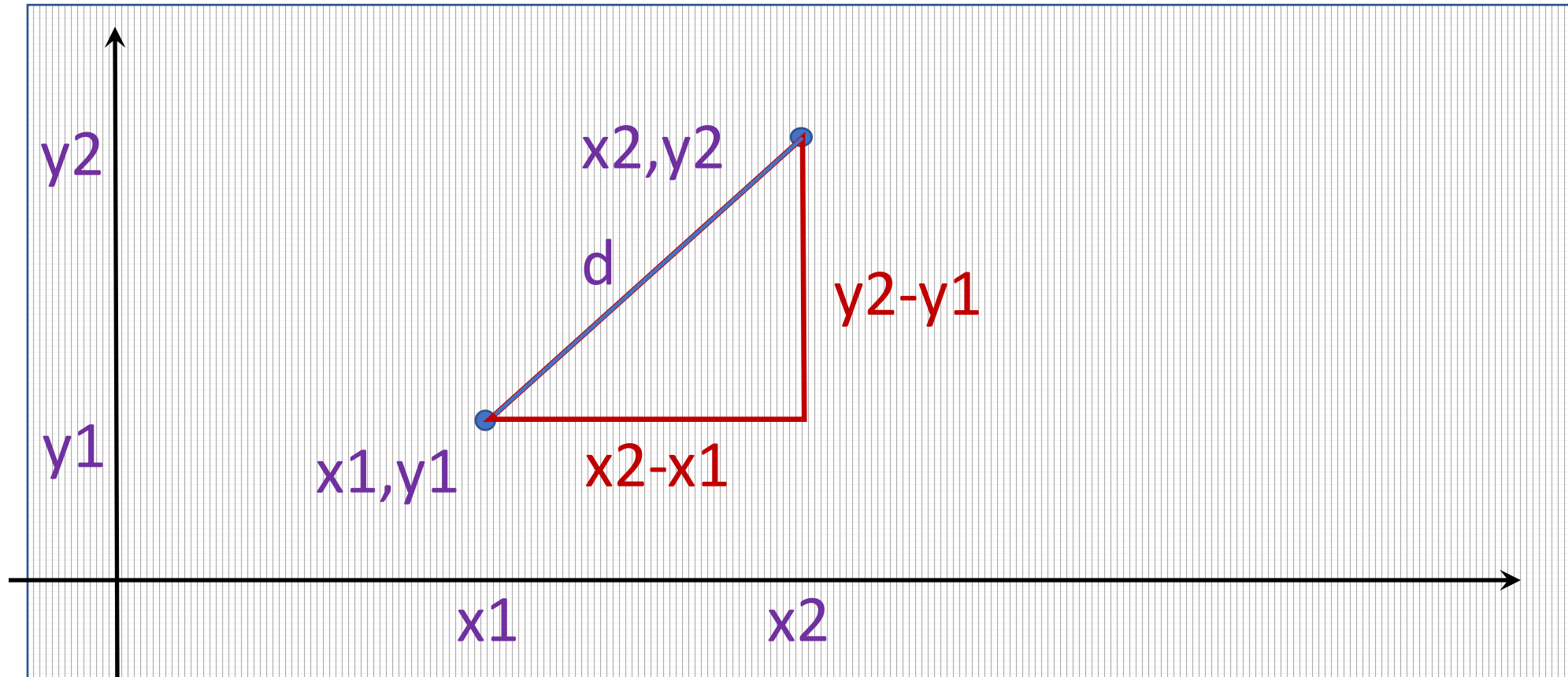
Να κατασκευάσετε πρόγραμμα που επιτρέπει στον χρήστη να ορίσει διαδοχικά σημεία. Για κάθε νέο σημείο που εισάγεται να εμφανίζει τις αποστάσεις των ήδη υφιστάμενων σημείων από το νέο σημείο.

Με <enter> τερματίζει το πρόγραμμα.


Σημείωση: οι συντεταγμένες να δίνονται ως 2 ακέραιοι χωρισμένοι με κόμμα: 100,50 (δεν απαιτείται αμυντικός προγραμματισμός, για έλεγχο της εισόδου του χρήστη).

σημεία στο πεδίο, λίγη γεωμετρία

Προγραμματίζω
με την python 



Η κλάση Point

Προγραμματίζω
με την python 

```
class Point():  
    ''' ένα σημείο στο καρτεσιανό επίπεδο '''  
    the_points = []  
    def __init__(self, x=0, y=0):  
        self.x = int(x)  
        self.y = int(y)  
        Point.the_points.append(self)  
  
    def distance(self, p):  
        return ((self.x - p.x)**2 + (self.y - p.y)**2 )**0.5
```


V2.1.3

Οι μέθοδοι

`__str__` και `__repr__`

Νίκος Αβούρης, Πανεπιστήμιο Πατρών


τρόποι αναπαράστασης αντικειμένου

Προγραμματίζω
με την python 

```
>>>s1
>>>
>>>print(s1)
<__main__.Student object at 0x105625908>>>>
```

- Το αντικείμενο `s1` δεν έχει αναπαράσταση στο περιβάλλον του διερμηνευτή,
- Η `print(s1)` επιστρέφει μόνο πληροφορία για την κλάση και τη διεύθυνση μνήμης

Οι μέθοδοι `__str__` και `__repr__` μας επιτρέπουν να ορίσουμε τι επιστρέφουν οι συναρτήσεις `repr()` και `print()`


Προγραμματίζω
με την python 

```
# Η __str__ αφορά την print(object) - χρήσιμη για το χρήστη
def __str__(self):
    place = self.origin[:-1] if self.origin[-1] in "ςs" else self.origin
    return self.name+', από ' + place + ' , ηλικία: {}'.format(self.age)

# Η __repr__ αφορά την repr(object) - χρήσιμη για debugging
def __repr__(self):
    return '['+','.join([self.name, self.origin, str(self.age)])+']'
```

Αν ορίσουμε την `__repr__` αυτή χρησιμοποιείται από την `repr()` και από την `print()` όχι όμως το αντίθετο

το αποτέλεσμα:

Προγραμματίζω
με την python 


```
>>>print(s1)
```

```
Ορέστης, από Βόλο, ηλικία: 22
```

```
>>>print(repr(s1))
```


```
[Ορέστης, Βόλος, 22]
```


Άσκηση: να τροποποιήσετε το παράδειγμα 2 ώστε η κλάση Point να τυπώνει τα αντικείμενά της

Προγραμματίζω
με την python 


```
class Point():  
    ''' ένα σημείο στο καρτεσιανό επίπεδο '''  
    the_points = []  
    def __init__(self, x=0, y=0):  
        self.x = int(x)  
        self.y = int(y)  
        Point.the_points.append(self)  
    def distance(self, p):  
        return ((self.x - p.x)**2 + (self.y - p.y)**2 )**0.5  
    def __str__(self):  
        return '('+str(self.x)+','+str(self.y)+')'
```

ειδικές μέθοδοι

Προγραμματίζω
με την python 

Ως τώρα έχουμε δει την `__init__()`
`__str__()` και την `__repr__()` ως
μεθόδους της κλάσης `object` που
κληρονομούν οι κλάσεις, και έχουν
ειδική συμπεριφορά, είναι οι λεγόμενες
ειδικές μέθοδοι.
υπάρχουν και άλλες που υλοποιούν
ειδικές χρήσεις, τελεστές.

ειδικές μέθοδοι

Προγραμματίζω
με την python 

`__init__` Δημιουργός αντικειμένων: `X = Class(args)`
`__del__` διαγραφή αντικειμένου `X`
`__add__` υλοποίηση του τελεστή `+` (πχ `X + Y`, `X += Y`)
`__or__` υλοποίηση του τελεστή `OR` | (bitwise OR) `X`
`__repr__`, **`__str__`** αναπαράσταση αντικειμένου `print(X)`, `repr(X)`, `str(X)`
`__call__` Κλήση συνάρτησης `X(*args, **kwargs)`
`__getattr__` Εύρεση γνωρίσματος `X.undefined`
`__setattr__` Τιμή σε γνώρισμα `X.any = value`
`__delattr__` Διαγραφή γνωρίσματος `del X.any`
`__len__` Μήκος `len(X)`
`__lt__`, **`__gt__`**, **`__le__`**, **`__ge__`**, **`__eq__`**, **`__ne__`** Τελεστής σύγκρισης `X < Y`, `X > Y`,
`X <= Y`, `X >= Y`, `X == Y`, `X != Y`
`__iter__`, **`__next__`** μέθοδοι για υλοποίηση επαναληπτικών δομών και συνοπτικών
λυστών
`__contains__` Έλεγχος το ανήκειν, `item in X`
`__enter__`, **`__exit__`** Χειριστής context

V2.1.4

διαγραφή αντικειμένων και γνωρισμάτων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών


del obj

διαγραφή αντικειμένου obj

```
>>>p1 =Point(100,100)
>>>p1
<Point object at 0x1037e1e48>
>>>del p1
>>>p1
```


```
Traceback (most recent call last):  File
"<input>", line 1, in <module>NameError:
name 'p1' is not defined
```

`delattr(obj, 'attr_name')` διαγράφη γνωρίσματος `attr_name`

Προγραμματίζω
με την python 

```
>>>p2 =Point(50,50)
>>>print(p2)
<Point object at 0x1037b5e10>
>>>delattr(p2, 'x')
>>>print(p2.x)
Traceback (most recent call last):  File
"<input>", line 1, in
<module>AttributeError: 'Point' object
has no attribute 'x'
```

Άσκηση: να τροποποιήσετε τον κώδικα της κλάσης Point ώστε ο χρήστης να μπορεί να διαγράψει ένα σημείο με βάση τις συντεταγμένες του

Προγραμματίζω
με την python 

Ο χρήστης θα πρέπει να δίνει ένα νέο σημείο με την εντολή `insert x,y`
και να διαγράφει ένα σημείο με την εντολή `delete x,y`

main program

while True:

 command = input('Εντολή (insert x,y ή delete x,y) :')

if command == '': **break**

if len(command.split())<2: **continue**

 coords = command.split()[1]

if coords.count(',') != 1: **continue**

 x, y = coords.split(',')

if x.isdigit() **and** y.isdigit():

if command.split()[0] == 'insert':

 new_point = Point(x,y)

 print('Υπάρχουν συνολικά {} σημεία'.format(len(Point.the_points)))

for p **in** Point.the_points:

if p != new_point:

 print('Το σημείο {} είναι σε απόσταση {:.2f} από το σημείο'.format(p, p.distance(new_point)))

elif command.split()[0] == 'delete':

 deleted = **False**

 new_points = []

for p **in** Point.the_points:

if p.x == int(x) **and** p.y == int(y):

del p

 deleted = **True**

else: new_points.append(p)

 Point.the_points = new_points

if deleted:

 print('Τα σημεία μετά τη διαγραφή είναι:')

for p **in** Point.the_points: print(p)


else: print('δεν βρέθηκε το σημείο')

Προγραμματίζω
με την python



point_v3

ενδοσκόπηση

Προγραμματίζω
με την python 

```
>>> class Tiny():
    def __init__(self, name):
        self.name = name
    def __str__(self):
        return 'my name is ..'+self.name+'
and i am an object of '+self.__class__.__name__
```

```
>>> s = Tiny('nikos')
>>> print(s)
my name is ..nikos and i am an object of Tiny
```

Τα ειδικά γνωρίσματα, όπως το `__class__`, επιτρέπουν σε αντικείμενα να γνωρίζουν το πλαίσιο στο οποίο υπάρχουν.

Άσκηση: να δημιουργήσετε κλάση, τα αντικείμενα της οποίας ξέρουν σε ποια κλάση ανήκουν (introspection - ενδοσκόπηση)

άσκηση

- Πώς θα τυπώσουμε όλα τα στιγμιότυπα μιας κλάσης;
- Χρησιμοποιήστε τη βιβλιοθήκη gc

```
import gc # garbage collector
for obj in gc.get_objects():
    if isinstance(obj, Tiny):
        print(obj)
```

L2.2

Παράδειγμα: οι κλάσεις Card-Deck

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην διάλεξη L2.2 θα εστιάσουμε στο πρώτο μας παράδειγμα ανάπτυξης κλάσεων, τις κλάσεις Card και Deck

week 2

κλάσεις και κληρονομικότητα

L2.1 Γνωρίσματα κλάσεων


L2.2 Παράδειγμα: Οι κλάσεις Card και Deck

L2.2 Κληρονομικότητα κλάσεων

L2.3 Παράδειγμα: το παιχνίδι 31

Μάθημα L2.2

Παράδειγμα: Οι κλάσεις Card-Deck

Προγραμματίζω
με την python 


- V2.2.1 Παράδειγμα: Η κλάση Card
- V2.2.2 Παράδειγμα: Η κλάση Deck
- V2.2.3 Διαγράμματα κλάσεων με UML

V2.2.1

παράδειγμα: η κλάση **Card**

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Η κλάση Card που αφορά τα φύλλα μιας τράπουλας

Προγραμματίζω
με την python 


Ένα αντικείμενο της κλάσης ορίζεται από δύο παραμέτρους: η μια αφορά την αξία και παίρνει ως τιμή έναν από τους εξής χαρακτήρες: **'A123456789TJQK'** ενώ η άλλη αφορά το σύμβολο του φύλλου και παίρνει τιμή έναν από τους εξής χαρακτήρες: **'cdhs'**, δηλαδή: clubs (♣), diamonds (♦), hearts (♥) and spades (♠) που αντιστοιχούν στα : σπαθί, καρό, κούπα, μπαστούνι

Js



βαλές μπαστούνι

Η κλάση Card – μέθοδοι

Προγραμματίζω
με την python 

Να ορίσετε τη μέθοδο που τυπώνει συνοπτικά το φύλλο και μια μέθοδο `detailed_info` που τυπώνει το Ελληνικό όνομά του (οι φιγούρες ονομάζονται Κ=ρήγας, Q=ντάμα, J=βαλές).

Να ορίσετε μια μεταβλητή κλάσης τύπου λεξικό στο οποίο φυλάσσονται τα ελληνικά ονόματα. Επίσης μια λίστα με τα φύλλα που έχουν δημιουργηθεί.

Js



βαλές μπαστούνι

Να επεκτείνετε την κλάση Card με τις εξής μεθόδους:


- `is_figure()` Επιστρέφει True αν ένα φύλλο είναι φιγούρα αλλιώς False
- `color()` Επιστρέφει την τιμή 'black' αν είναι ένα από τα φύλλα με μαύρο χρώμα ή 'red' αν είναι κόκκινο

V2.2.2

παράδειγμα: η κλάση **Deck**

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Να ορισθεί μια κλάση **Deck** που περιγράφει τράπουλες

Προγραμματίζω
με την python 

Ένα αντικείμενο της κλάσης περιέχει αρχικά 52 φύλλα. Θα πρέπει το αντικείμενο αυτό να μπορεί να ανακατέψει τα φύλλα του (μέθοδος **shuffle**), να μας επιτρέπει να τραβήξουμε φύλλο (μέθοδος **draw**), ενώ θα πρέπει ακόμη να μπορεί να μαζέψει τα φύλλα που έχει μοιράσει (μέθοδος **collect**).

Υλοποιείται με δύο λίστες, τη λίστα **content** που περιέχει τα φύλλα που η τράπουλα περιέχει και την **pile** που περιέχει τα φύλλα που έχουν μοιραστεί. Μπορείτε να χρησιμοποιήσετε την κλάση **Card**.



Deck
+ content: List + pile: List
+ shuffle(): None + draw(): Card + collect(): None

Να επεκτείνετε την κλάση Deck με τη μέθοδο `pile_details()` που παράγει στην Ελληνική γλώσσα μια πλήρη περιγραφή των φύλλων που έχουν τραβηχτεί (είναι στο τραπέζι). πχ

Τα φύλλα στο τραπέζι είναι:

Τέσσερα κούπα


Ντάμα σπαθί

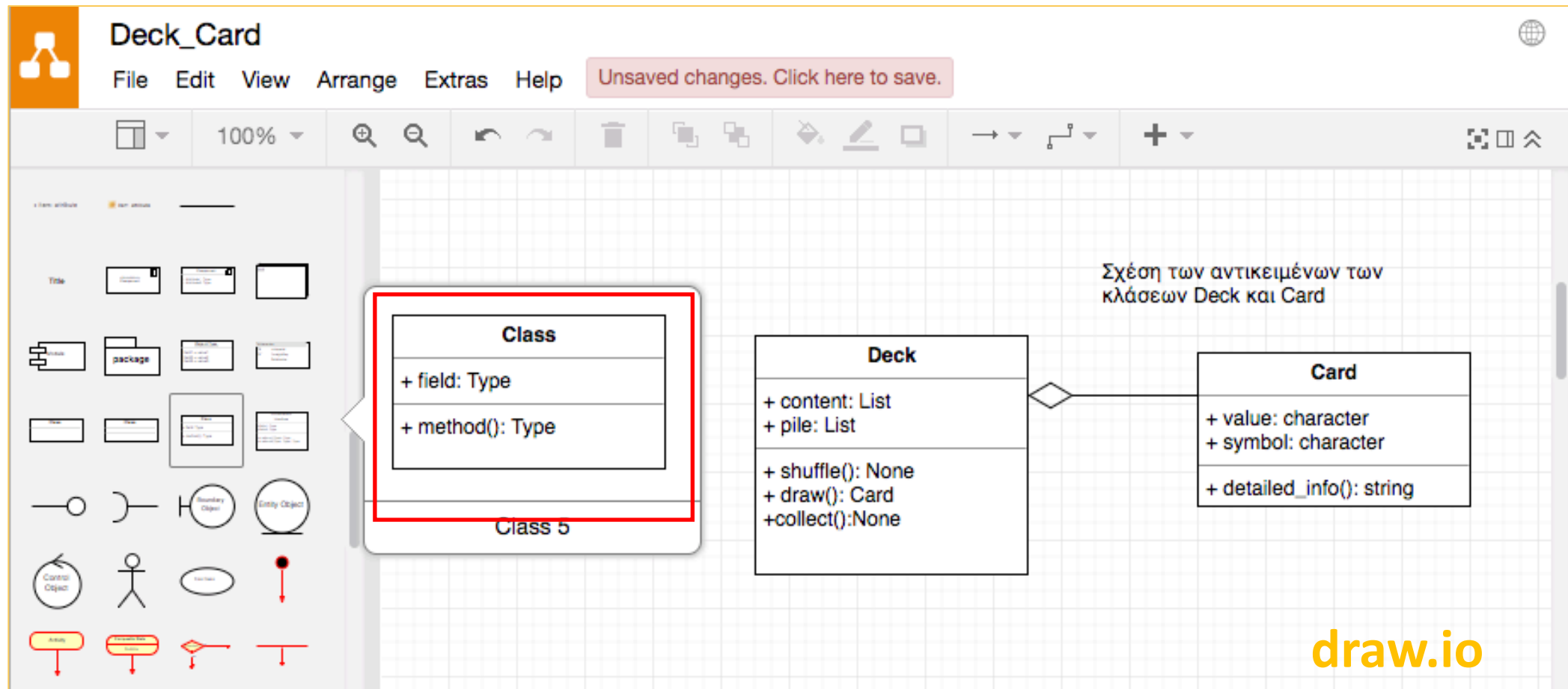
V2.2.3

διαγράμματα κλάσεων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

αποτύπωση σχέσης κλάσεων σε διάγραμμα UML

Προγραμματίζω
με την python 



L2.3

κληρονομικότητα κλάσεων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην διάλεξη L2.3 θα δούμε μηχανισμούς για δημιουργία υπο-κλάσεων, κληρονομικότητα καθώς

week 2

κλάσεις και κληρονομικότητα

L2.1 Γνωρίσματα κλάσεων


L2.2 Παράδειγμα: Οι κλάσεις Card και Deck

L2.3 Κληρονομικότητα κλάσεων

L2.4 Παράδειγμα: το παιχνίδι 31

Μάθημα L2.3

Κληρονομικότητα κλάσεων

Προγραμματίζω
με την python 

- V2.3.1 Κληρονομικότητα κλάσεων
- V2.3.2 Παράδειγμα: η μισθοδοσία
- V2.3.3 Μέθοδοι κλάσεων


V2.3.1

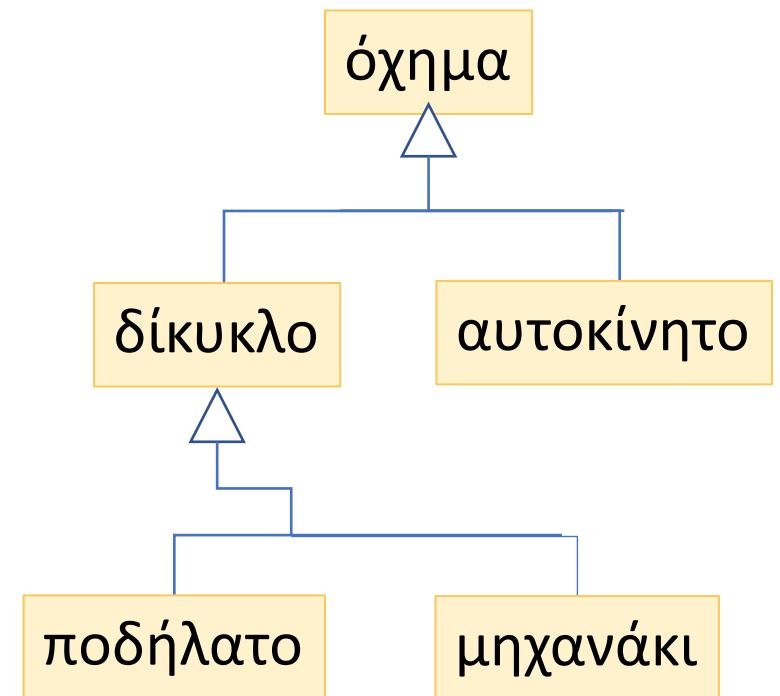
κληρονομικότητα κλάσεων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών


Οι κλάσεις κληρονομούν γνωρίσματα και μεθόδους από άλλες κλάσεις

Ένα βασικό χαρακτηριστικό του αντικειμενοστραφούς μοντέλου προγραμματισμού είναι η δυνατότητα να ορίσουμε ιεραρχία κλάσεων που επιτρέπουν εξειδίκευση-γενίκευση των γνωρισμάτων τους.

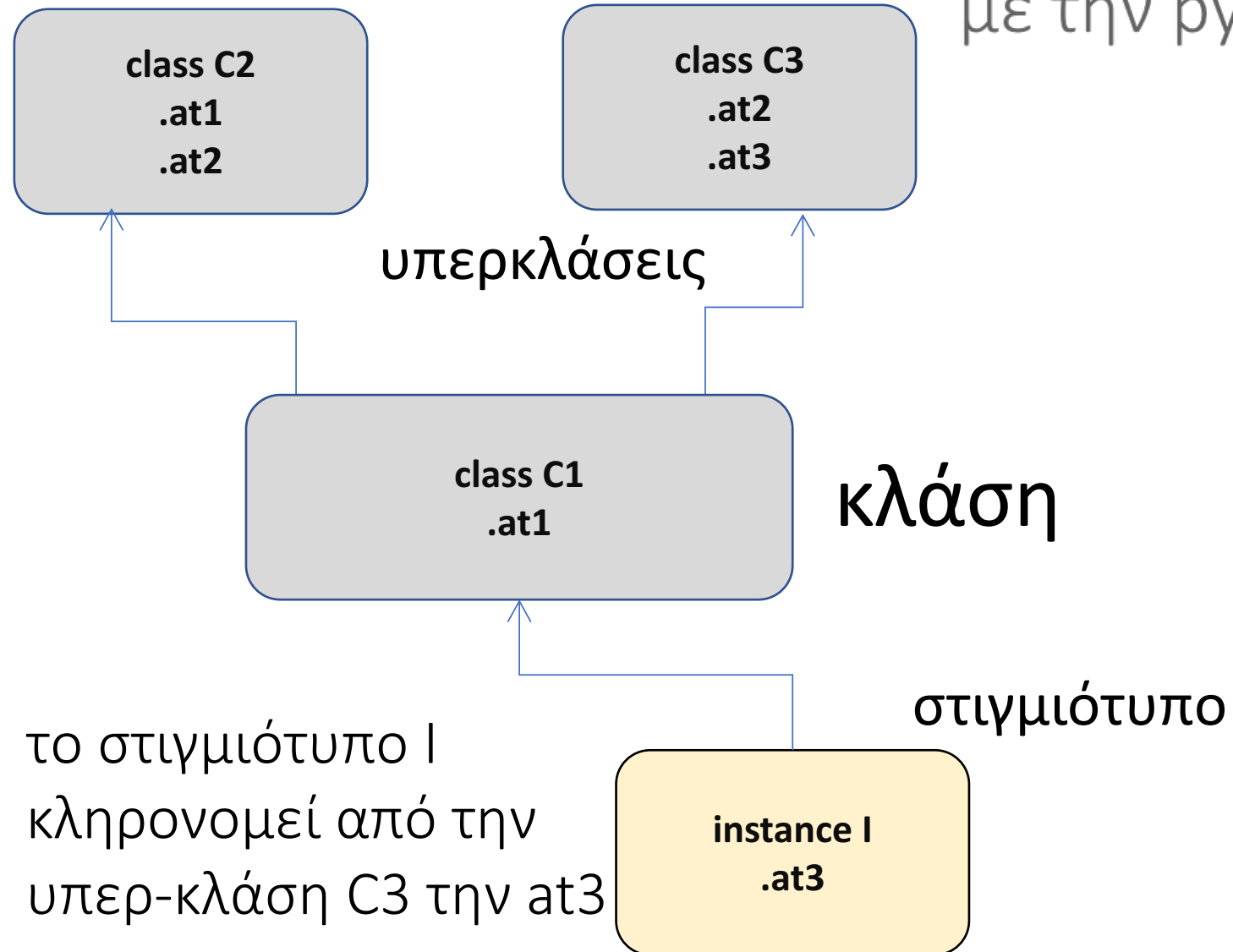
Προγραμματίζω
με την python 



πολλαπλή κληρονομικότητα


Προγραμματίζω
με την python 

MRO: method resolution order:
κάθε γνώρισμα
αναζητείται στις
υπερ-κλάσεις από
κάτω προς τα
πάνω, και από
αριστερά στα δεξιά



το στιγμιότυπο I
κληρονομεί από την
υπερ-κλάση C3 την at3

ορισμός κλάσης που κληρονομεί από υπερ-κλάση

Προγραμματίζω
με την python 

```
class NewClass(BaseClass):  
    ...
```


```
>>> class B(object):  
        a = 1  
>>> class C(B):  
        b = 3  
  
>>> x = C()  
>>> x.a  
1
```

η κλάση C κληρονομεί την
κλάση B (η υπερ-κλάση της)

άσκηση


Ποια η τιμή των
παρακάτω
εκφράσεων:

1. `x.a`
2. `x.b`
3. `x.c`
4. `x.d`
5. `x.f`
6. `x.f()`
7. `C.f`
8. `B.h`

Προγραμματίζω
με την python 

```
class B(object):  
    a = 10  
    b = 20  
    def f(self): print('method f in class B')  
    def g(self): print('method f in class B')  
  
class C(B):  
    b = 30  
    c = 40  
    d = 50  
    def g(self): print('method g in class C')  
    def h(self): print('method h in class c')  
  
x = C()  
x.d = 60  
x.e = 70
```

Άσκηση: αποτέλεσμα

Προγραμματίζω
με την python 

```
class B(object):
    a = 10
    b = 20
    def f(self): print('method f')
    def g(self): print('method g')

class C(B):
    b = 30
    c = 40
    d = 50
    def g(self): print('method g')
    def h(self): print('method h')

x = C()
x.d = 60
x.e = 70
```

```
>>> x.a
10
>>> x.b
30
>>> x.c
40
>>> x.d
60
>>> x.f
<bound method B.f of <__main__.C object at 0x105625b00>>
>>> x.f()
method f in class B
>>> C.f
<function B.f at 0x100660e18>
>>> B.h
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    B.h
AttributeError: type object 'B' has no attribute 'h'
>>>
```

V2.3.2

παράδειγμα: μισθοδοσία

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

παράδειγμα

Παράδειγμα Μισθοδοσία υπαλλήλων μιας επιχείρησης
Έστω μια επιχείρηση στην οποία ορίζονται διαφορετικοί κανόνες για τις αυξήσεις διαφόρων κατηγοριών υπαλλήλων, τα στελέχη - εκτός από την αύξηση που δίνεται σε όλους - παίρνουν μπόνους 10%.
Ορίζουμε μια υπερ-κλάση `Person` για όλους τους υπαλλήλους και μια υπο-κλάση `Manager` για τα στελέχη.

κλάσεις Person και Manager


Προγραμματίζω
με την python



```
class Person():
    employees = []
    def __init__(self, name, job='', salary=0):
        self.name = name.strip()
        self.job = job.strip()
        self.salary = float(salary)
        Person.employees.append(self)
    def give_raise(self, percent):
        '''percent of salary increase with values between 0 and 1'''
        self.salary = float(self.salary*(1+percent))
    def __str__(self):
        sal = "{:.2f}".format(self.salary) if self.salary > 0 else ""
        return self.name+' '+self.job+ ': '+sal

class Manager(Person):
    def give_raise(self, percent, bonus = 0.10):
        Person.give_raise(self, percent+bonus)
```


επαναχρησιμοποίηση κώδικα

Προγραμματίζω
με την python 

```
class Manager(Person):  
    def give_raise(self, percent, bonus = 0.10):  
        Person.give_raise(self,percent+bonus)
```

Θα μπορούσαμε να είχαμε δημιουργήσει μια ολότελα νέα συνάρτηση `give_raise` για τα αντικείμενα της κλάσης `Manager`. Όμως είναι καλύτερη πρακτική να επαναχρησιμοποιούμε τον κώδικα της υπερ-κλάσης

Ερώτηση πώς θα εξασφαλίσουμε όλα τα αντικείμενα της υπο-κλάσης manager να έχουν job='Διευθυντής'

Προγραμματίζω
με την python 


```
def __init__(self, name, salary=0):  
    Person.__init__(self, name, 'Διευθυντής', salary)
```

V2.3.3

μέθοδοι κλάσεων

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

μέθοδοι κλάσεων


Προγραμματίζω
με την python 

υπάρχουν περιπτώσεις που μέθοδοι χρειάζεται να οριστούν στο επίπεδο της κλάσης και όχι των αντικειμένων

```
class C():
    num_instances = 0
    def __init__(self):
        C.num_instances += 1
    def print_num_instances(): # μέθοδος κλάσης
        print("Αριθμός στιγμιότυπων: {}".format( C.num_instances))

>>> a = C()
>>> b = C()
>>> C.print_num_instances()
Αριθμός στιγμιότυπων: 2
```

στατικές μέθοδοι κλάσεων


Προγραμματίζω
με την python 

για να κληθεί μια μέθοδος κλάσης μέσω στιγμιότυπων της κλάσης πρέπει να χρησιμοποιηθεί ο προσδιορισμός `staticmethod()`.

```
class C:
    num_instances = 0
    def __init__(self):
        C.num_instances += 1
    def print_num_instances(): # μέθοδος κλάσης
        print("Αριθμός στιγμιότυπων: {}".format( C.num_instances))
    print_num_instances = staticmethod (print_num_instances)
```

```
>>> a = C()
>>> b = C()
>>> a.print_num_instances()
Αριθμός στιγμιότυπων: 2
>>> C.print_num_instances()
Αριθμός στιγμιότυπων: 2
```

@staticmethod

Προγραμματίζω
με την python 

Εναλλακτικά για να χρησιμοποιήσουμε μια στατική κλάση ή κλάση μεθόδου εκτός της κλάσης, πχ από αντικείμενα της κλάσης, μπορεί να χρησιμοποιηθεί διακοσμητής @staticmethod

```
class C:
    num_instances = 0
    def __init__(self):
        C.num_instances += 1

    @staticmethod #decorator
    def print_num_instances(): # μέθοδος κλάσης
        print("Αριθμός στιγμιότυπων: {}".format( C.num_instances))

>>> a = C()
>>> b = C()
>>> a.print_num_instances()
Αριθμός στιγμιότυπων: 2
>>> C.print_num_instances()
Αριθμός στιγμιότυπων: 2
```


L2.4

Παράδειγμα : το παιχνίδι 31

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Στην διάλεξη L2.4 θα δούμε βήμα-βήμα την ανάπτυξη μιας εφαρμογής ακολουθώντας το αντικειμενοστραφές μοντέλο. Πρόκειται για το παιχνίδι 31.

week 2

κλάσεις και κληρονομικότητα

week 2

L2.1 Γνωρίσματα κλάσεων


L2.2 Παράδειγμα: Οι κλάσεις Card και Deck

L2.3 Κληρονομικότητα κλάσεων

L2.4 Παράδειγμα: το παιχνίδι 31

Μάθημα L2.4

Παράδειγμα: το παιχνίδι 31

Προγραμματίζω
με την python 

V2.4.1 Η κλάση Game

V2.4.2 Η κλάση Player


V2.4.1

Παράδειγμα: το παιχνίδι 31

Μέρος (α): η κλάση Game

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

Να γράψετε πρόγραμμα που παίζει το παιχνίδι 31


Προγραμματίζω
με την python 

Να ορίσετε κλάσεις **Player**, **Game**. Να χρησιμοποιήσετε τις κλάσεις **Card**, **Deck**

Κανόνες:

- παίζουν 2 έως 8 παίκτες με ονόματα Παίκτης-1, 2, 3, ...
- ο Παίκτης 1 είναι ο υπολογιστής (η μάνα)
- οι παίκτες τραβάνε φύλλα όσο πιο κοντά στο 31, αν το περάσουν καίγονται.
- Σε κάθε γύρο κερδίζει ο παίκτης με το υψηλότερο σκορ

η κλάση Game

Προγραμματίζω
με την python 

```
class Game():
```

```
    '''ξεκινάει το παιχνίδι, ανακατεύει την τράπουλα, δίνει τη σειρά στους παίκτες και αποφασίζει ποιος νίκησε'''
```

```
    def __init__(self):
```

```
        print('Παίζουμε 31 !!!')
```

```
        self.d = pc.Deck() # η τράπουλα
```

```
        self.d.shuffle()
```

```
        self.n_players = self.number_of_players() # αριθμός παικτών
```

```
        self.players = [] # λίστα με τους παίκτες
```

```
        for i in range(self.n_players):
```

```
            if i == 0:
```

```
                self.players.append(ComputerPlayer('Παίκτης-' + str(i + 1), self.d))
```

```
            else:
```

```
                self.players.append(Player('Παίκτης-' + str(i+1), self.d))
```

```
        self.show_players()
```

```
        self.play_game()
```

```
    def play_game(self):
```

```
        for p in self.players:
```

```
            print(50*'*', '\nΠαίζει ο παίκτης {}'.format(p))
```

```
            p.plays()
```

```
        self.show_winner()
```


V2.4.2

Παράδειγμα: το παιχνίδι 31

Μέρος (β): η κλάση Player

Νίκος Αβούρης, Πανεπιστήμιο Πατρών

η κλάση Player

Προγραμματίζω
με την python 

```
class Player():
    ''' ο παίκτης του 31. Ξέρει τους κανόνες '''
    def __init__(self, name, deck):
        self.name = name
        self.deck = deck
        self.myscore = 0
    def plays(self):
        card = self.deck.draw() # ο παίκτης τραβάει φύλλο
        print('O {} τράβηξε: {}'.format(self.name, card.detailed_info()))
        card_value = self.calculate_value(card)
        self.myscore += card_value
        self.check_if_exceeded()
        if self.myscore == -1 :
            return
        else:
            if self.strategy(): self.plays()
            else: return
    def strategy(self):
        reply = input('Σκορ: {} συνεχίζεις (v/o):'.format(self.myscore))
        if not reply or reply.lower() not in 'oo':
            return True
        return False
```



V2.4.3

Παράδειγμα: το παιχνίδι 31

Μέρος (γ): η κλάση ComputerPlayer


Νίκος Αβούρης, Πανεπιστήμιο Πατρών

η κλάση ComputerPlayer

Προγραμματίζω
με την python 

```
class ComputerPlayer(Player):  
    '''παίκτης που τραβάει μόνος του φύλλα, έχει στρατηγική'''  
    def plays(self):  
        card = self.deck.draw() # ο παίκτης τραβάει φύλλο  
        print('Ο υπολογιστής ({}), τράβηξε: {}'.format(self.name, card.detailed_info()))  
        card_value = self._calculate_value(card)  
        self.myscore += card_value  
        self._check_if_exceeded()  
        if self._computer_strategy(): self.plays()  
        else:  
            print('ΥΠΟΛΟΓΙΣΤΗΣ:', self)  
    def _computer_strategy(self):  
        return False if self.myscore >= 25 or self.myscore == -1 else True
```

άσκηση

Προγραμματίζω
με την python 

- Η τρέχουσα έκδοση μετράει τον άσσο ως 1, να το τροποποιήσετε ώστε να μετράει είτε 1 είτε 11.
- Να τροποποιήσετε τη στρατηγική του υπολογιστή με βάση τον κανόνα αυτό