# Εφαρμογές Η/Υ

4ο Μάθημα

- Αντικειμενοστραφής προγραμματισμός

Ν. Λαγαρός, Θ. Στάμος, Χ. Φραγκουδάκης

# Εισαγωγή στην Python

Γενικά για τον αντικειμενοστραφή προγραμματισμό

- Προγραμματισμός προσανατολισμένος στις διαδικασίες
    - Σύνολο εντολών που εκτελούνται σειριακά
    - Υλοποιείται μια λογική που επιδρά στην είσοδο και παράγει την έξοδο
    - Χρήσιμη προσέγγιση για τα περισσότερα απλά προγράμματα
    - Η λογική είναι στο κέντρο και επιδρά στα αντικείμενα
- Προγραμματισμός προσανατολισμένος στα αντικείμενα (OOP)
    - Τα αντικείμενα είναι στο κέντρο και ενσωματώνουν λογική

# Εισαγωγή στην Python

Γενικά για τον αντικειμενοστραφή προγραμματισμό

- Στον OOP ξεκινάμε με τον ορισμό των αντικειμένων που θα χειριστούμε
- Τα αντικείμενα ορίζονται με την περιγραφή των χαρακτηριστικών τους
- Ορίζουμε ένα πρότυπο για τα αντικείμενα που ονομάζεται κλάση
- Κάθε αντικείμενο ανήκει σε μια κλάση και κληρονομεί τα χαρακτηριστικά της
- Οι μέθοδοι είναι διαδικασίες που επιδρούν στο αντικείμενο ή περιγράφουν το τρόπο που αλληλεπιδρά με άλλα αντικείμενα της ίδια ή άλλης κλάσης
- Κάθε αντικείμενο είναι ξεχωριστή οντότητα από άλλα αντικείμενα της κλάσης

Λέμε ότι ένα αντικείμενο είναι στιγμιότυπο μιας κλάσης

- Παράδειγμα κλάσης είναι ο άνθρωπος με τα χαρακτηριστικά όνομα και ηλικία
  - Ένα αντικείμενο της κλάσης είναι η Ελένη που είναι 23 χρονών
  - Ταυτόχρονα, άλλο αντικείμενο είναι ο Γιώργος που είναι 11 χρονών
  - Μέθοδοι της κλάσης είναι οι διαδικασίες `περπατάει`, `μιλάει`, ...

# Εισαγωγή στην Python

Γενικά για τον αντικειμενοστραφή προγραμματισμό

Έχουμε ήδη χρησιμοποιήσει κλάσεις και αντικείμενα

| | |
|---|---|
| Αρχικοποίηση αντικειμένου: | `alist = list()` |
| Χρήση μεθόδου της κλάσης: | `alist.append('something')` |
| Χρήση αντικειμένου με τελεστές: | `'something' in alist` |
| Χρήση γενικών συναρτήσεων: | `len(alist)` |
| Δημιουργία νέου ονόματος: | `otherlist = alist` |
| Διαγραφή ενός ονόματος: | `del otherlist` |

# Εισαγωγή στην Python

Γενικά για τον αντικειμενοστραφή προγραμματισμό

Μεταβλητές ≡ Αναφορές ≡ Ονόματα

Στην Python κάθε μεταβλητή είναι ένα όνομα που αναφέρεται σε ένα αντικείμενο (ένα κομμάτι μνήμης). Ένα αντικείμενο μπορεί να έχει οσεσδήποτε αναφορές.

Το `==` εξετάζει την ισοδυναμία

- Εξετάζει αντικείμενα (περιοχές της μνήμης)
- Είναι `True` αν τα αντικείμενα έχουν το ίδιο περιεχόμενο

Το `is` εξετάζει την ταυτότητα

- Εξετάζει τις μεταβλητές (ονόματα), δηλαδή τις αναφορές τους
- Είναι `True` αν οι αναφορές είναι προς το ίδιο αντικείμενο

# Εισαγωγή στην Python

Γενικά για τον αντικειμενοστραφή προγραμματισμό

Duck Typing: If it walks like a duck, swims like a duck and quacks like a duck, it must be a duck.

Το στύλ της Python είναι να μην ελέγχουμε τους τύπους των αντικειμένων αλλά να καλούμε τις μεθόδους `walk`, `swim` και `quak`. Αν το αντικείμενο ανταποκρίνεται τότε είναι μια πάπια αλλιώς το πρόγραμμα κάνει run-time error.

EAFP principle: Easier to Ask for Forgiveness than Permission

```
try:
    obj.walk()
    obj.swim()
    obj.quak()
except:
    # obj is not a duck
```

# Εισαγωγή στην Python

## Ορισμός κλάσης και χαρακτηριστικά

```
class Pet:
    "A class that abstracts pets"

lady = Pet()
lady.name = "Lady"
lady.number_of_legs = 4
print("{} has {} legs.".format(lady.name, lady.number_of_legs))

nemo = Pet()
nemo.name = "Nemo"
nemo.number_of_legs = 0
print("{} has {} legs.".format(nemo.name, nemo.number_of_legs))
```

- Μια κλάση έχει ένα όνομα και τα περιεχόμενά της γράφονται με εσοχές
- Το όνομα της κλάσης δεν ακολουθείται από παρενθέσεις
- Δημιουργία στιγμιότυπου
  - Ανάθεση του ονόματος της κλάσης σε άλλο όνομα
  - Μπορεί να πάρει παραμέτρους γι' αυτό και οι παρενθέσεις `()`
  - Μπορεί να έχει οποιοδήποτε χαρακτηριστικό (`name`, `color`, ...)
  - Πρόσβαση στα χαρακτηριστικά του στιγμιότυπου με την τελεία

# Εισαγωγή στην Python

## Μέθοδοι των κλάσεων

```
class Pet:
    "A class that abstracts pets"

    def sleep(self):
        print("Zzzzz")

lady = Pet()
lady.name = "Lady"
lady.number_of_legs = 4
print("{} has {} legs.".format(lady.name, lady.number_of_legs))
lady.sleep()

nemo = Pet()
nemo.name = "Nemo"
nemo.number_of_legs = 0
print("{} has {} legs.".format(nemo.name, nemo.number_of_legs))
nemo.sleep()
```

- Οι μέθοδοι εφοδιάζουν τα αντικείμενα με λογική
- Οι μέθοδοι είναι συναρτήσεις εμφωλευμένες μέσα σε κλάσεις
- Μια μέθοδος μιας κλάσης έχει πάντα σαν πρώτο όρισμα το `self`
- Όταν η Python καλεί τη μέθοδο εισάγει στη θέση του `self` το στιγμιότυπο

# Εισαγωγή στην Python

## Η ειδική μέθοδος `__init__`

```
class Pet:
    "A class that abstracts pets"

    def __init__(self, name, number_of_legs):
        self.name = name
        self.number_of_legs = number_of_legs

    def sleep(self):
        print(self.name, "says Zzzzz...")
```

```
lady = Pet("Lady", 4)
print("{} has {} legs.".format(lady.name, lady.number_of_legs))
lady.sleep()

nemo = Pet("Nemo", 0)
print("{} has {} legs.".format(nemo.name, nemo.number_of_legs))
nemo.sleep()
```

- Η ειδική μέθοδος `__init__` δέχεται τις παραμέτρους του στιγμιότυπου
- Συνήθως αναθέτει τις παραμέτρους στα χαρακτηριστικά του στιγμιότυπου
- Καλείται αυτόματα, αν υπάρχει, αμέσως μετά τη δημιουργία του στιγμιότυπου

# Εισαγωγή στην Python

## Η ειδική μέθοδος `__repr__`

```
class Pet:
    "A class that abstracts pets"

    def __init__(self, name, number_of_legs):
        self.name = name
        self.number_of_legs = number_of_legs

    def __repr__(self):
        return "{} has {} legs.".format(self.name, self.number_of_legs)

    def sleep(self):
        print(self.name, "says Zzzzz...")

lady = Pet("Lady", 4)
print(lady)
lady.sleep()

nemo = Pet("Nemo", 0)
print(nemo)
nemo.sleep()
```

- Η ειδική μέθοδος `__repr__` επιστρέφει την αναπαράσταση του στιγμιότυπου σαν συμβολοσειρά

# Εισαγωγή στην Python

## Κληρονομικότητα στις κλάσεις

```
class Pet:
    "A class that abstracts pets"

    def __init__(self, name, number_of_legs):
        self.name = name
        self.number_of_legs = number_of_legs

    def __repr__(self):
        return "{} has {} legs.".format(self.name, self.number_of_legs)

    def sleep(self):
        print(self.name, "says Zzzzz...")

class Dog(Pet):
    "A class that inherits from Pet and abstracts dogs"

    def bark(self):
        print(self.name, "says Woof!")

lady = Dog("Lady", 4)
lady.bark()
lady.sleep()
```

# Εισαγωγή στην Python

## Μια κλάση που περιγράφει σημεία στο επίπεδο

```
class Point2(object):
    " A class for 2D points "
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __repr__(self):
        return "Point2(%s, %s)" % (self.x, self.y)

a = Point2()
b = Point2(-1.2, 9)

print(a, b)
```

# Εισαγωγή στην Python

Μια εμπλουτισμένη κλάση που περιγράφει σημεία στο επίπεδο

```
class Point2(object):

    def __init__(self, x, y):
        self.x, self.y = x, y

    def __repr__(self):
        return "Point2(%s, %s)" % (self.x, self.y)

    @classmethod
    def from_point2(cls, point2):
        return cls(point2.x, point2.y)

    @classmethod
    def from_tuple(cls, tup):
        return cls(tup[0], tup[1])

    @property
    def coordinates(self):
        return self.x, self.y
    @coordinates.setter
    def coordinates(self, tup):
        self.x, self.y = tup[0], tup[1]
```

# Εισαγωγή στην Python

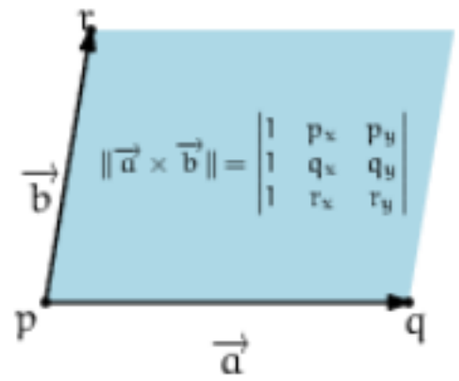
Υλοποίηση λεξικογραφικής διάταξης των σημείων του επιπέδου

```
class Point2(object):

        ...

    def __eq__(self, other):
        return (self.x, self.y) == (other.x, other.y)

    def __ne__(self, other):
        return (self.x, self.y) != (other.x, other.y)

    def __lt__(self, other):
        return (self.x, self.y) < (other.x, other.y)

    def __gt__(self, other):
        return (self.x, self.y) > (other.x, other.y)

    def __le__(self, other):
        return  (self.x, self.y) <= (other.x, other.y)

    def __ge__(self, other):
        return  (self.x, self.y) >= (other.x, other.y)

        ...
```

# Εισαγωγή στην Python

Εμβαδό παραλληλογράμμου

- Ένα παραλληλόγραμμο δίνεται σαν δύο διανύσματα με κοινή αρχή
- Τρία σημεία $p, q, r = (p_x, p_y), (q_x, q_y), (r_x, r_y)$
- Το εξωτερικό γινόμενο δύο διανυσμάτων έχει μέτρο ίσο με το εμβαδό του παραλληλογράμμου που ορίζουν



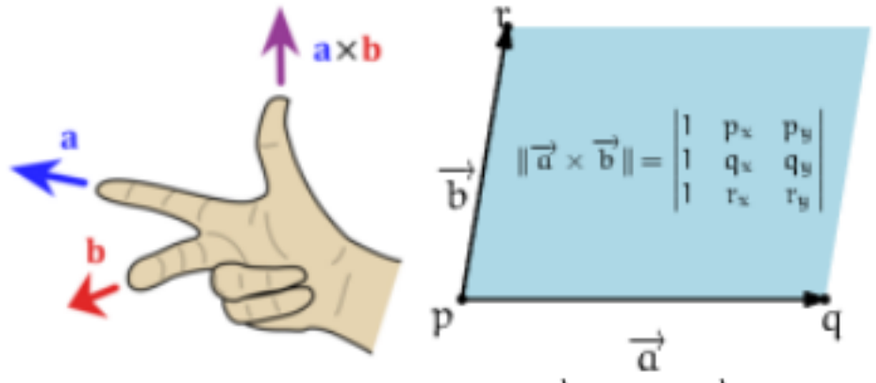

$$\| \overrightarrow{a} \times \overrightarrow{b} \| = \begin{vmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{vmatrix} = \cdots = (r_y - p_y)(q_x - p_x) - (q_y - p_y)(r_x - p_x)$$

```
def area(p, q, r):
    return (r.y-p.y) * (q.x-p.x) - (q.y-p.y) * (r.x-p.x)
```

# Εισαγωγή στην Python

Πρόσημο του εμβαδού του παραλληλογράμμου

- Κανόνας του δεξιού χεριού




- Θετικό εμβαδό σημαίνει ότι το διάνυσμα $\overrightarrow{a} = \overrightarrow{pq}$ πρέπει να στρίψει αριστερά για να βρει το $\overrightarrow{b} = \overrightarrow{pr}$.
- Με άλλα λόγια, θετικό εμβαδό σημαίνει ότι το σημείο $r$ βρίσκεται στα αριστερά του $\overrightarrow{pq}$
- Αρνητικό εμβαδό σημαίνει ότι το σημείο $r$ βρίσκεται στα δεξιά του $\overrightarrow{qp}$
- Μηδενικό εμβαδό σημαίνει ότι τα σημεία $p, q, r$ είναι συνευθειακά.

# Εισαγωγή στην Python

Μια κλάση για τον προσανατολισμό τριών σημείων στο επίπεδο

```
class Orientation:
    " Represents the orientation of a point triple "
    def __init__(self, p, q, r):
        self.p, self.q, self.r = p, q, r

    def __repr__(self):
        if self.area() > 0:
            orientationstr = "COUNTERCLOCKWISE"
        elif self.area() < 0:
            orientationstr = "CLOCKWISE"
        else: orientationstr = "COLLINEAR"
        return "{} {} {} are oriented {}".format \
            (self.p, self.q, self.r, orientationstr)

    def area(self):
        return (self.r.y - self.p.y) * (self.q.x - self.p.x) - \
            (self.q.y - self.p.y) * (self.r.x - self.p.x)

p, q, r = Point2(0,0), Point2(3,0), Point2(0,4)

print(Orientation(p,q,r))
print(Orientation(q,p,r))
print(Orientation(p,q,q))
```

# Ερωτήσεις;