Projet Foot
2I013

Nicolas Baskiotis
nicolas.baskiotis@lip6.fr

Université Pierre et Marie Curie (UPMC)
Laboratoire d'Informatique de Paris 6 (LIP6)

S2 (2014-2015)

Python : prise en main

Opérations usuelles

`=`	affectation
`+, -, *, /, %`	opérations arith.
`a ** b`	a^b
`==, <=, >=, !=`	comparaisons
`and, or, not`	opérations logiques
`#, """ """`	commentaire

```
>>> a = 3; b = 3.5; c = "Joueur"
>>> a + b
6.5
>>> a * b
10.5
>>> b**a
42.875
```

```
>>> c + c
'JoueurJoueur'
>>> c*3
'JoueurJoueurJoueur'
>>> str(a) + c; `a` + c;
'Joueur3'
'Joueur3'
```

Python : prise en main

Structures

`if condition :`
`  a = 2`
`  b = 3`
`else :`
`  a = 3`
	Bloc conditionel

`for i in range(10):`
`  a += i`
`  b += i`
	Boucle

`while i < 10:` `  i += 1`	Boucle conditionelle

`def fonction(x,y,z=0):`
`  a = x + y`
`  return a - z`
	Fonction

Python : prise en main

Type d'import	Utilisation
`from module import ma_fonction`	`ma_fonction()`
`from module import *`	`ma_fonction()`
`import module`	`module.ma_fonction()`

Modules et fonctions utiles

math				random	
	`sqrt`	`sin`	`cos`		`randint`
	`floor`	`abs`	`ceil`		`random`
	`pi`	`log`	`exp`		`uniform`
pickle	`dump`	`load`			

Structures de données

Listes

initialisation	`l = []`	`list()`
	`l = [0, 1, 2, 3, 4, 5]`	`range(5)`
accession	`l[1] -> 1`	`l[1:4] -> [1, 2, 3]`
	`l[:3] -> [0, 1, 2]`	`l[-1] -> 5`
opérations	`l.append(6)`	`l + [6,7,8]`
	`2 in l`	`len(l) -> 6`
	`max(l), min(l)`	`sorted(l), l.sort()`

Dictionnaires

```
d={}, dict()
d={'a':'avt','b':'bfr','c':'chx'}
d=dict([('a','avt'),('b','bfr'),('c','chx')])
d['a'] -> 'avt'
d.items(), d.keys(), d.values()
```

N-uplets, tuples

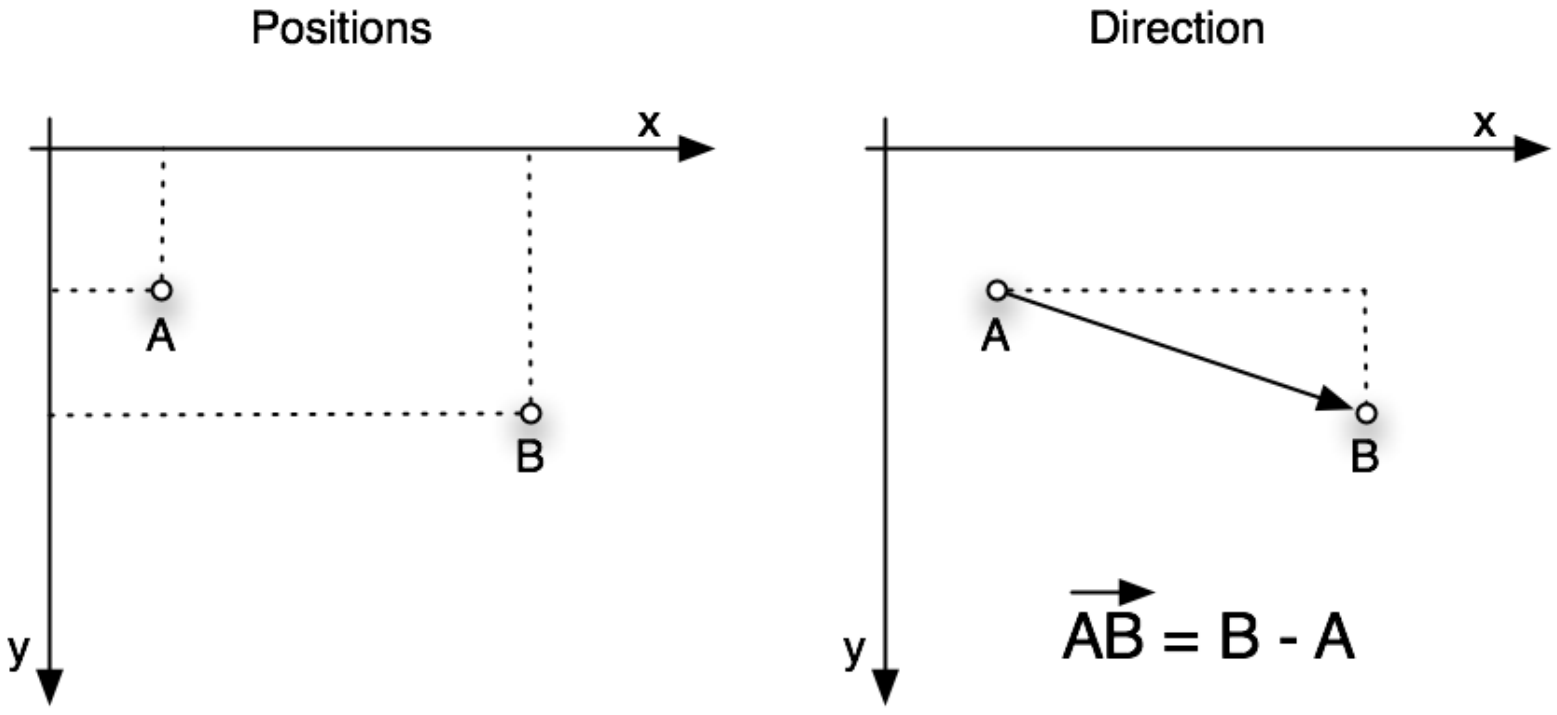
```
t=(1,2,3), t[1]
```

Objet

```
  class SoccerState:
    def __init__(self,team1,team2,ball):
        self.team1=team1
        self.team2=team2
        self._winning_team=0
        self.ball=ball
    def get_team(self,teamid):
        if teamid==1:
            return self.team1
        if teamid==2:
            return self.team2
    def get_player(self,num_team,player):
       return self.get_team(num_team).get_player(player)
class SoccerTeam:
    def __init__(self,name,soccer_club=None):
        self._name=name
        self._exceptions=[]
        self._players=dict()
    def add_player(self,player):
        self._players[player.name]=player
    @property
    def players(self):
        return self._players.values()
```

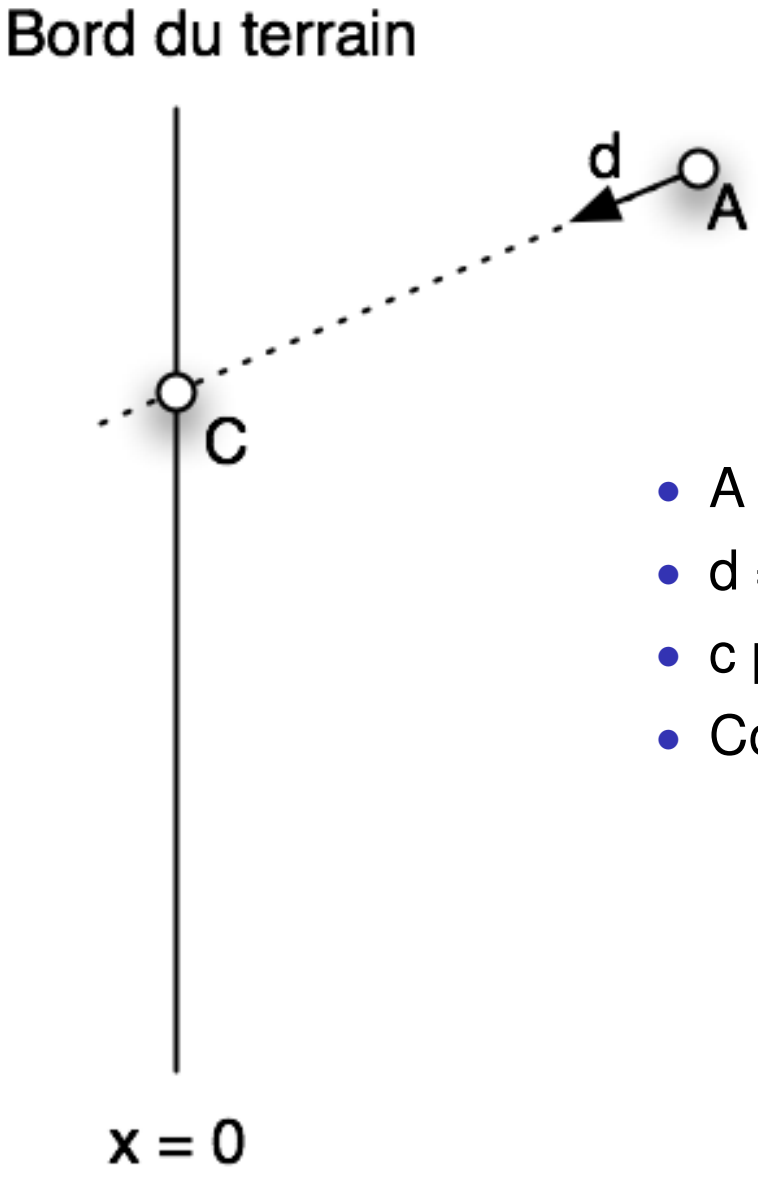
Géometrie

`Vector2D` est utilisée pour la position et pour la direction :



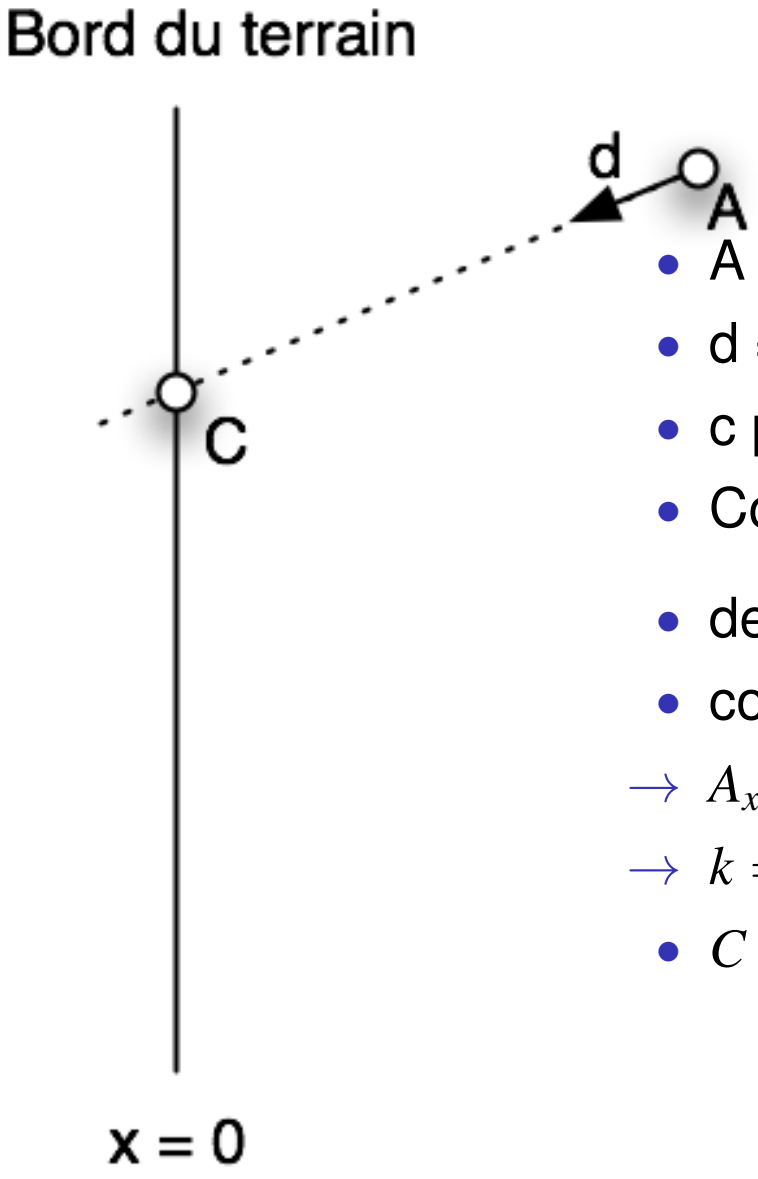

```
a = Vector2D(1,1); b = Vector2D(4,4);
ab = b - a
```

Géometrie



- A position courante
- d = direction
- c point de collision avec le bord du terrain
- Comment trouver c ?

Géometrie



- A position courante
- d = direction
- c point de collision avec le bord du terrain
- Comment trouver c ?

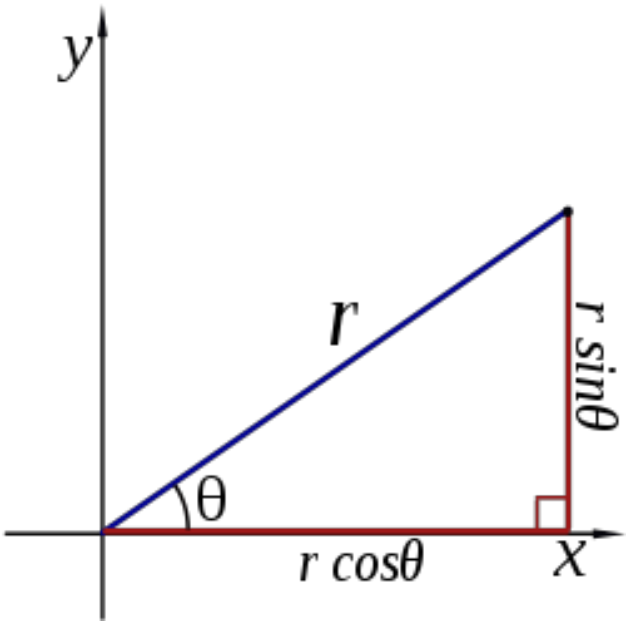
- demi-droite d'équation : $A + k * d$
- contrainte : $x = 0$

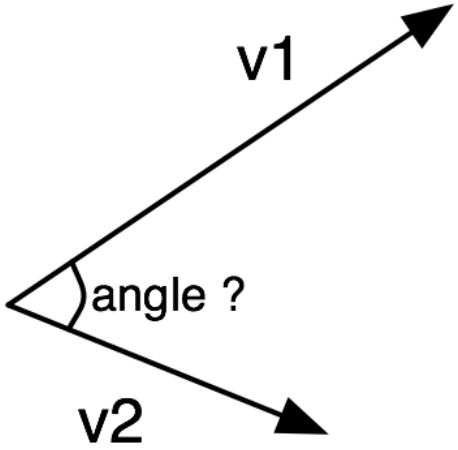
$\rightarrow A_x + k * d_x = 0$

$\rightarrow k = -A_x/d_x,\ y = a_y - A_x * d_y/d_x$

- $C = (0, A_y - A_x * d_y/d_x)$

Coordonnées polaires/cartésiennes





- Coordonnées cartésiennes : définies par un vecteur, $v = (v_x, v_y)$
- Coordonnées polaires : définies par un rayon R et un angle θ
- Cartésien $\rightarrow$ polaire :
  $R = ||v|| = \sqrt{v_x * v_x + v_y * v_y}$
  $\theta = atan2(v_y, v_x)$
- Polaire $\rightarrow$ cartésien
  $v_x = R * cos(\theta)$
  $v_y = R * sin(\theta)$
- angle α entre deux vecteurs v_1 et v_2:
  $v_1 \cdot v_2 = ||v_1|| \times ||v_2|| * cos(\alpha)$
  $\alpha = \frac{acos(v_1 \cdot v_2)}{(||v_1|| \times ||v_2||)}$
- Attention : angle en gradiant !