

## Instructions conditionnelles

## C5 Initiation à Python avec turtle

### Instructions conditionnelles

- La syntaxe d'une instruction conditionnelle en Python est :

```
1  if <condition >:  
2      <instructions1 >  
3  else:  
4      <instructions2 >
```

Cela permet d'exécuter les <instructions1> si la condition est vérifiée, sinon on exécute les <instructions2>.


## C5 Initiation à Python avec turtle

### Instructions conditionnelles

- La syntaxe d'une instruction conditionnelle en Python est :

```
1  if <condition >:  
2      <instructions1 >  
3  else:  
4      <instructions2 >
```

Cela permet d'exécuter les <instructions1> si la condition est vérifiée, sinon on exécute les <instructions2>.

-  On fera bien attention à la syntaxe du langage, et notamment à l'usage du caractère **:** qui suit la condition (et le else) et à l'**indentation**, c'est à dire le décalage des instructions qui doivent s'exécuter.

## C5 Initiation à Python avec turtle

### Exemples

- Ecrire l'instruction permettant de tester si la variable `erreurs` vaut 0

## C5 Initiation à Python avec turtle

### Exemples

- 1 Ecrire l'instruction permettant de tester si la variable `erreurs` vaut 0
- 2 On suppose qu'une variable `longueur` peut être positive ou négative, si cette variable est positive alors on fait avancer la tortue de `longueur`, sinon on la fait reculer de `-longueur`. Ecrire les instructions python correspondantes.

## C5 Initiation à Python avec turtle

### Exemples

- ❶ Ecrire l'instruction permettant de tester si la variable `erreurs` vaut 0

```
1  if erreur==0:
```

- ❷ On suppose qu'une variable `longueur` peut être positive ou négative, si cette variable est positive alors on fait avancer la tortue de `longueur`, sinon on la fait reculer de `-longueur`. Ecrire les instructions python correspondantes.

```
1      if longueur > 0:
2          crayon.forward(longueur)
3      else:
4          crayon.backward(-longueur)
```

## Boucles while

## C5 Initiation à Python avec turtle

### Boucles while

- La syntaxe d'une boucle **while** en Python est :

```
1 while <condition>:  
2     <instruction>
```

Cela permet d'exécuter les <instructions> tant que la <condition> est vérifiée.



## C5 Initiation à Python avec turtle

### Boucles while

- La syntaxe d'une boucle **while** en Python est :

```
1  while <condition>:  
2      <instruction>
```

Cela permet d'exécuter les <instructions> tant que la <condition> est vérifiée.

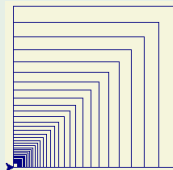
- On ne sait pas a priori combien de fois cette boucle sera exécutée (et elle peut même être infinie), on dit que c'est une boucle **non bornée**.

## C5 Initiation à Python avec turtle

### Exemple d'une boucle `while`

On suppose déjà crée une fonction `carre(c)` qui dessine un carré de côté `c` à partir de la position courante de la tortue. Ecrire un programme Python, permettant de tracer la figure suivante sachant que :

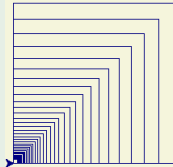
- le carré initial à 200 pixels de côté
- le côté des carrés intérieur diminue de dix pourcents à chaque étape
- le plus petit carré a un côté mesurant plus de 5 pixels.



# C5 Initiation à Python avec turtle

## Exemple d'une boucle while

```
1 cote = 200
2 while cote > 5:
3     carre(cote)
4     cote = cote * 0.9
```



## C5 Initiation à Python avec turtle

### Fonction renvoyant un résultat

En plus d'exécuter un bloc d'instructions, une fonction peut transmettre une valeur au reste du programme à l'aide d'une instruction `return`. On utilise alors la syntaxe suivante :

```
1     def <nom_fonction>(<arguments>):  
2         <instruction>  
3         return <valeur>
```

## C5 Initiation à Python avec turtle

### Exemple de fonction contenant un `return`

La fonction ci-dessous, renvoie la moyenne des deux nombres donnés en argument

```
1  def moyenne(x, y):  
2      m = (x+y)/2  
3      return m
```