

Remarques

- Il existe *plusieurs milliers* de langages informatique.

Remarques

- Il existe *plusieurs milliers* de langages informatique.
- Parmi les plus importants historiquement et les plus utilisés, on peut citer :

Remarques

- Il existe *plusieurs milliers* de langages informatique.
- Parmi les plus importants historiquement et les plus utilisés, on peut citer :
 - Cobol, Fortran, Lisp,

Remarques

- Il existe *plusieurs milliers* de langages informatique.
- Parmi les plus importants historiquement et les plus utilisés, on peut citer :
 - Cobol, Fortran, Lisp,
 - le langage C et ses dérivés, Pascal, Ada

Remarques

- Il existe *plusieurs milliers* de langages informatique.
- Parmi les plus importants historiquement et les plus utilisés, on peut citer :
 - Cobol, Fortran, Lisp,
 - le langage C et ses dérivés, Pascal, Ada
 - Perl, Javascript, PHP, Python, Java.

Remarques

- Il existe *plusieurs milliers* de langages informatique.
- Parmi les plus importants historiquement et les plus utilisés, on peut citer :
 - Cobol, Fortran, Lisp,
 - le langage C et ses dérivés, Pascal, Ada
 - Perl, Javascript, PHP, Python, Java.
- Ne pas confondre langages de programmations et algorithmes. Un algorithme est une méthode permettant de résoudre un problème. Un langage informatique permet de formuler un algorithme pour l'exécuter sur un ordinateur.

A propos de Python

- Langage créé par un informaticien néerlandais : [Guido Van Rossum](#). La première version date de 1991, nous en sommes actuellement à la version 3.

A propos de Python

- Langage créé par un informaticien néerlandais : [Guido Van Rossum](#). La première version date de 1991, nous en sommes actuellement à la version 3.
- C'est un langage sous licence libre, gratuit, multiplateforme.

C3 Initiation à Python avec turtle

A propos de Python

- Langage créé par un informaticien néerlandais : [Guido Van Rossum](#). La première version date de 1991, nous en sommes actuellement à la version 3.
- C'est un langage sous licence libre, gratuit, multiplateforme.
- C'est un langage extensible grâce à l'ajout de nombreuses bibliothèques.

C3 Initiation à Python avec turtle

A propos de Python

- Langage créé par un informaticien néerlandais : [Guido Van Rossum](#). La première version date de 1991, nous en sommes actuellement à la version 3.
- C'est un langage sous licence libre, gratuit, multiplateforme.
- C'est un langage extensible grâce à l'ajout de nombreuses bibliothèques.
- C'est un langage interprété.

C3 Initiation à Python avec turtle

A propos de Python

- Langage créé par un informaticien néerlandais : [Guido Van Rossum](#). La première version date de 1991, nous en sommes actuellement à la version 3.
- C'est un langage sous licence libre, gratuit, multiplateforme.
- C'est un langage extensible grâce à l'ajout de nombreuses bibliothèques.
- C'est un langage interprété.
- Choisi pour l'enseignement de la spécialité NSI au lycée.

Programmation en python : généralités

- Python renvoie un message d'erreur lorsqu'il n'arrive pas à interpréter les instructions de votre programme. Prendre l'habitude de **lire attentivement** ces messages, qui sont de premiers indices pour déterminer la source de l'erreur

Programmation en python : généralités

- Python renvoie un message d'erreur lorsqu'il n'arrive pas à interpréter les instructions de votre programme. Prendre l'habitude de **lire attentivement** ces messages, qui sont de premiers indices pour déterminer la source de l'erreur
- En Python, les **commentaires** s'écrivent en faisant commencer la ligne par le caractère **#**.

Programmation en python : généralités

- Python renvoie un message d'erreur lorsqu'il n'arrive pas à interpréter les instructions de votre programme. Prendre l'habitude de **lire attentivement** ces messages, qui sont de premiers indices pour déterminer la source de l'erreur
- En Python, les **commentaires** s'écrivent en faisant commencer la ligne par le caractère **#**.
- Le respect de la syntaxe du langage est **fondamental** et demande de la **rigueur**.

Programmation en python : généralités

- Python renvoie un message d'erreur lorsqu'il n'arrive pas à interpréter les instructions de votre programme. Prendre l'habitude de **lire attentivement** ces messages, qui sont de premiers indices pour déterminer la source de l'erreur
- En Python, les **commentaires** s'écrivent en faisant commencer la ligne par le caractère **#**.
- Le respect de la syntaxe du langage est **fondamental** et demande de la **rigueur**.
- Attention aussi à bien surveiller les correspondances entre les parenthèses mais aussi les guillemets ou les apostrophes qui sont souvent source d'erreurs.

C3 Initiation à Python avec turtle

Création du papier et du crayon

```
1 import turtle
2 papier = turtle.Screen()
3 crayon = turtle.Turtle()
```


C3 Initiation à Python avec turtle

Création du papier et du crayon

```
1 import turtle
2 papier = turtle.Screen()
3 crayon = turtle.Turtle()
```

Remarques

C3 Initiation à Python avec turtle

Création du papier et du crayon

```
1 import turtle
2 papier = turtle.Screen()
3 crayon = turtle.Turtle()
```

Remarques

- les noms papier et crayon sont des **noms de variables**, choisis par le programmeur.

C3 Initiation à Python avec turtle

Création du papier et du crayon

```
1 import turtle
2 papier = turtle.Screen()
3 crayon = turtle.Turtle()
```

Remarques

- les noms papier et crayon sont des **noms de variables**, choisis par le programmeur.
- On peut créer plusieurs crayons différents (`style1 = turtle.Turtle()`).

C3 Initiation à Python avec turtle

Création du papier et du crayon

```
1 import turtle
2 papier = turtle.Screen()
3 crayon = turtle.Turtle()
```

Remarques

- les noms papier et crayon sont des **noms de variables**, choisis par le programmeur.
- On peut créer plusieurs crayons différents (`stylo = turtle.Turtle()`).
- l'instruction `crayon.reset()` permet d'effacer la totalité des tracés de la tortue nommée crayon.

C3 Initiation à Python avec turtle

Création du papier et du crayon

```
1 import turtle
2 papier = turtle.Screen()
3 crayon = turtle.Turtle()
```

Remarques

- les noms papier et crayon sont des **noms de variables**, choisis par le programmeur.
- On peut créer plusieurs crayons différents (`style = turtle.Turtle()`).
- l'instruction `crayon.reset()` permet d'effacer la totalité des tracés de la tortue nommée crayon.
- l'instruction `crayon.undo()` permet d'effacer le dernier tracé de la tortue nommée crayon.

C3 Initiation à Python avec turtle

Attention

Dans la suite, on supposera que la tortue a été nommée `crayon` et l'écran `papier`. Mais, ces noms sont **choisis par le programmeur**.

Propriétés de la tortue

C3 Initiation à Python avec turtle

⚠ Attention

Dans la suite, on supposera que la tortue a été nommée `crayon` et l'écran `papier`. Mais, ces noms sont **choisis par le programmeur**.

Propriétés de la tortue

- `crayon.pensize(size)` fixe l'épaisseur de la tortue à `size`.

C3 Initiation à Python avec turtle

Attention

Dans la suite, on supposera que la tortue a été nommée `crayon` et l'écran `papier`. Mais, ces noms sont **choisis par le programmeur**.

Propriétés de la tortue

- `crayon.pensize(size)` fixe l'épaisseur de la tortue à `size`.
- `crayon.color(color)` change à `color` la couleur de la tortue.

C3 Initiation à Python avec turtle

Attention

Dans la suite, on supposera que la tortue a été nommée `crayon` et l'écran `papier`. Mais, ces noms sont **choisis par le programmeur**.

Propriétés de la tortue

- `crayon.pensize(size)` fixe l'épaisseur de la tortue à `size`.
- `crayon.color(color)` change à `color` la couleur de la tortue.
- `crayon.penup()` et `crayon.pendown()` permettent respectivement de relever ou d'abaisser la tortue.

C3 Initiation à Python avec turtle

⚠ Attention

Dans la suite, on supposera que la tortue a été nommée `crayon` et l'écran `papier`. Mais, ces noms sont **choisis par le programmeur**.

Propriétés de la tortue

- `crayon.pensize(size)` fixe l'épaisseur de la tortue à `size`.
- `crayon.color(color)` change à `color` la couleur de la tortue.
- `crayon.penup()` et `crayon.pendown()` permettent respectivement de relever ou d'abaisser la tortue.
- `crayon.showturtle()` et `crayon.hideturtle()` permettent respectivement de faire apparaître ou non la tortue.

C3 Initiation à Python avec turtle

⚠ Attention

Dans la suite, on supposera que la tortue a été nommée `crayon` et l'écran `papier`. Mais, ces noms sont **choisis par le programmeur**.

Propriétés de la tortue

- `crayon.pensize(size)` fixe l'épaisseur de la tortue à `size`.
- `crayon.color(color)` change à `color` la couleur de la tortue.
- `crayon.penup()` et `crayon.pendown()` permettent respectivement de relever ou d'abaisser la tortue.
- `crayon.showturtle()` et `crayon.hideturtle()` permettent respectivement de faire apparaître ou non la tortue.
- `crayon.speed(s)` pour modifier la vitesse de tracé.

Exemples

Ecrire les instructions permettant d'obtenir un crayon abaissé, rouge, d'épaisseur 3, caché et se déplaçant à la vitesse maximale.

C3 Initiation à Python avec turtle

Exemples

Ecrire les instructions permettant d'obtenir un crayon abaissé, rouge, d'épaisseur 3, caché et se déplaçant à la vitesse maximale.

```
1 crayon . pendown ()
2 crayon . pensize (3)
3 crayon . color ("red")
4 crayon . hideturtle ()
5 crayon . speed (10)
```

C3 Initiation à Python avec turtle

Orientation de la tortue

C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.

C3 Initiation à Python avec turtle

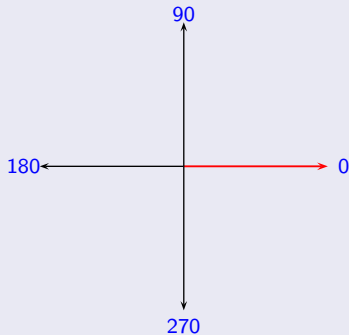
Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.

C3 Initiation à Python avec turtle

Orientation de la tortue

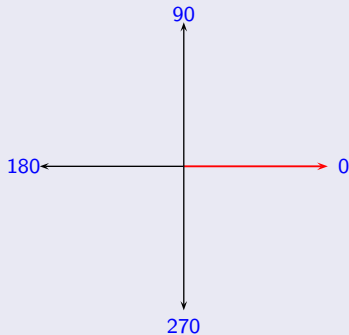
- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation



C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation

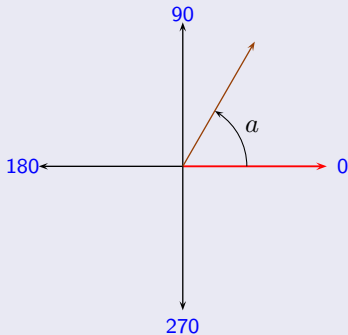


- `crayon.setheading(a)` pour fixer l'orientation de la tortue à l'angle `a`.

C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation

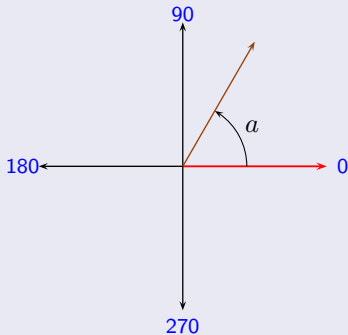


- `crayon.setheading(a)` pour fixer l'orientation de la tortue à l'angle a .

C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation

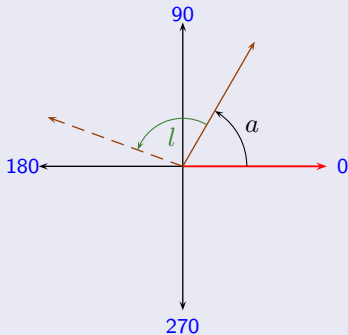


- `crayon.setheading(a)` pour fixer l'orientation de la tortue à l'angle a .
- `crayon.left(1)` pour faire tourner la tortue de 1 degré à gauche à partir de son orientation actuelle.

C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation

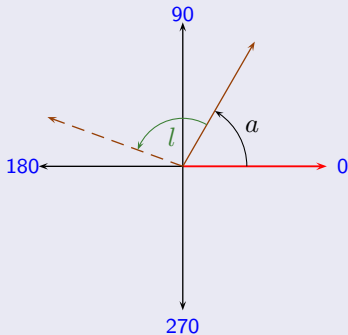


- `crayon.setheading(a)` pour fixer l'orientation de la tortue à l'angle a .
- `crayon.left(1)` pour faire tourner la tortue de 1 degré à gauche à partir de son orientation actuelle.

C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation

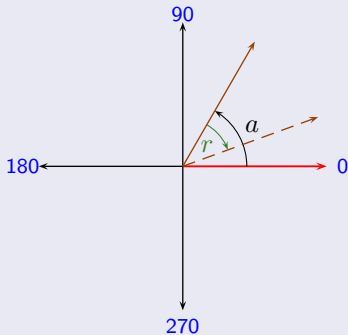


- `crayon.setheading(a)` pour fixer l'orientation de la tortue à l'angle a .
- `crayon.left(l)` pour faire tourner la tortue de l degrés à gauche à partir de son orientation actuelle.
- `crayon.right(r)` pour faire tourner la tortue de r degrés à droite à partir de son orientation actuelle.

C3 Initiation à Python avec turtle

Orientation de la tortue

- L'orientation de la tortue est définie par l'angle qu'elle fait avec l'axe horizontale et est initialement fixé à 0.
- Les instructions suivantes permettent de modifier cette orientation



- `crayon.setheading(a)` pour fixer l'orientation de la tortue à l'angle a .
- `crayon.left(1)` pour faire tourner la tortue de 1 degrés à gauche à partir de son orientation actuelle.
- `crayon.right(r)` pour faire tourner la tortue de r degrés à droite à partir de son orientation actuelle.

C3 Initiation à Python avec turtle

Déplacement de la tortue

C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.

C3 Initiation à Python avec turtle

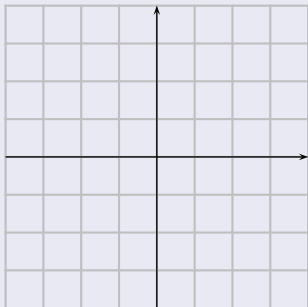
Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.

C3 Initiation à Python avec turtle

Déplacement de la tortue

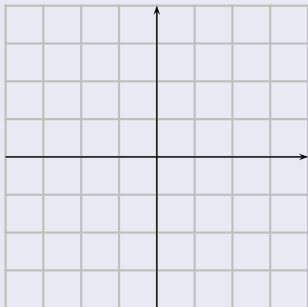
- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position



C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position

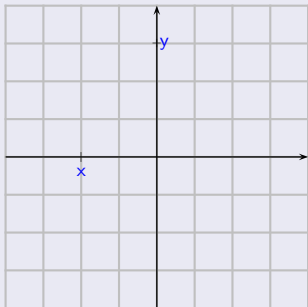


- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .

C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position

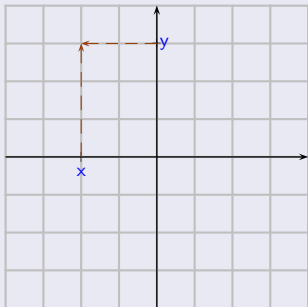


- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .

C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position

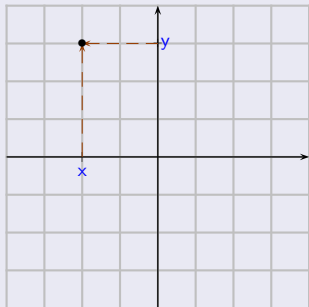


- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .

C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position

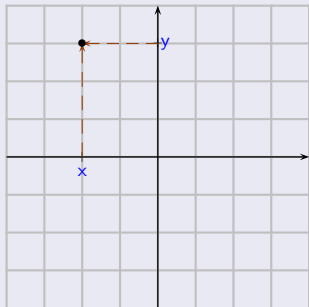


- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .

C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position

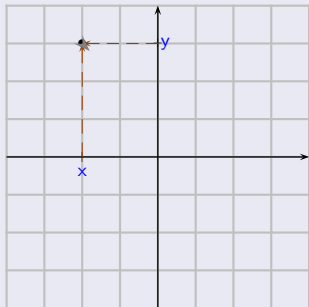


- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .
- `crayon.forward(1)` pour faire avancer la tortue d'une distance 1 dans sa direction actuelle.

C3 Initiation à Python avec turtle

Déplacement de la tortue

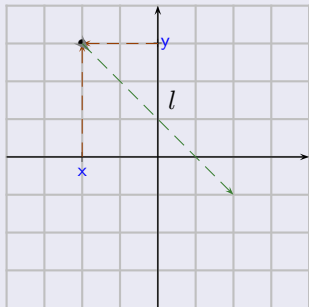
- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position



- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .
- `crayon.forward(1)` pour faire avancer la tortue d'une distance 1 dans sa direction actuelle.

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position

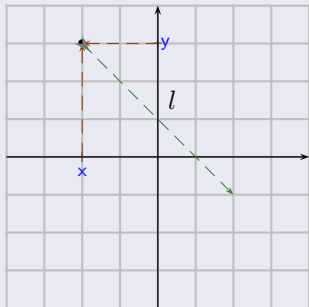


- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .
- `crayon.forward(1)` pour faire avancer la tortue d'une distance 1 dans sa direction actuelle.

C3 Initiation à Python avec turtle

Déplacement de la tortue

- La position de la tortue est définie par ses coordonnées dans un repère (comme en mathématiques) et est initialement l'origine du repère.
- Les instructions suivantes permettent de modifier cette position



- `crayon.goto(x,y)` pour déplacer la tortue au point de coordonnées (x,y) .
- `crayon.forward(1)` pour faire avancer la tortue d'une distance 1 dans sa direction actuelle.
- `crayon.backward(1)` pour faire reculer la tortue d'une distance 1 dans la direction opposée à sa direction actuelle.

Fonctions

Fonctions

- Les fonctions sont des blocs d'instructions destinés à accomplir une tâche lors de leur **appel** (par exemple avec `turtle`, tracé un carré).

Fonctions

- Les fonctions sont des blocs d'instructions destinés à accomplir une tâche lors de leur **appel** (par exemple avec `turtle`, tracé un carré).
- Leurs résultats peut dépendre de valeurs appelées **paramètres** de la fonction (par exemple, le côté du carré).

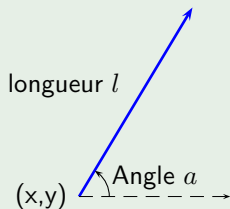
Fonctions

- Les fonctions sont des blocs d'instructions destinés à accomplir une tâche lors de leur **appel** (par exemple avec `turtle`, tracé un carré).
- Leurs résultats peut dépendre de valeurs appelées **paramètres** de la fonction (par exemple, le côté du carré).
- Pour définir une fonction en Python, on utilise la syntaxe suivante :

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

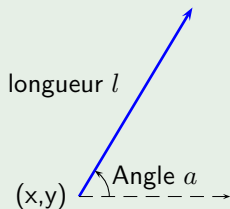
Un exemple de fonction

Tracer un trait à partir du point de coordonnées (x,y) en donnant la longueur l et la direction a .



Un exemple de fonction

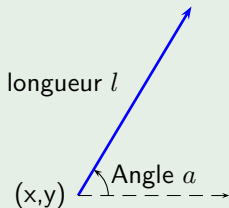
Tracer un trait à partir du point de coordonnées (x,y) en donnant la longueur l et la direction a .



- Relever le crayon

Un exemple de fonction

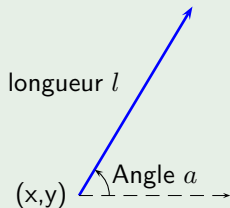
Tracer un trait à partir du point de coordonnées (x,y) en donnant la longueur l et la direction a .



- Relever le crayon
- Aller en (x,y)

Un exemple de fonction

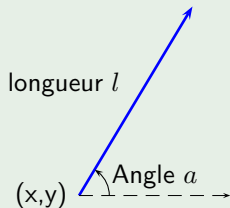
Tracer un trait à partir du point de coordonnées (x, y) en donnant la longueur l et la direction a .



- Relever le crayon
- Aller en (x, y)
- S'orienter dans la direction d'angle a

Un exemple de fonction

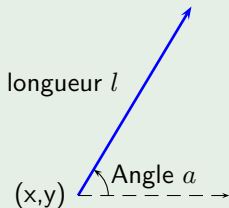
Tracer un trait à partir du point de coordonnées (x,y) en donnant la longueur l et la direction a .



- Relever le crayon
- Aller en (x,y)
- S'orienter dans la direction d'angle a
- Abaisser le crayon

Un exemple de fonction

Tracer un trait à partir du point de coordonnées (x,y) en donnant la longueur l et la direction a .

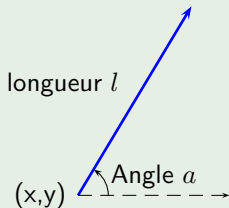


- Relever le crayon
- Aller en (x,y)
- S'orienter dans la direction d'angle a
- Abaisser le crayon
- Avancer de la longueur l

C3 Initiation à Python avec turtle

Un exemple de fonction

Tracer un trait à partir du point de coordonnées (x,y) en donnant la longueur l et la direction a .



- Relever le crayon
- Aller en (x,y)
- S'orienter dans la direction d'angle a
- Abaisser le crayon
- Avancer de la longueur l

```
1 def trait(x,y,l,a):
2     crayon.penup()
3     crayon.goto(x,y)
4     crayon.setheading(a)
5     crayon.pendown()
6     crayon.forward(l)
```

Boucles for

Boucles for

- Les instructions :

```
1  for <variable> in range(<entier>):  
2      <instructions>
```

permet de créer une variable parcourant les entiers de 0 à <entier> (exclu).

Boucles for

- Les instructions :

```
1  for <variable> in range(<entier>):  
2      <instructions>
```

permet de créer une variable parcourant les entiers de 0 à <entier> (exclu).

- Les <instructions> indentées qui suivent seront exécutées pour chaque valeur prise par la variable.

Boucles for

- Les instructions :

```
1  for <variable> in range(<entier>):  
2      <instructions>
```

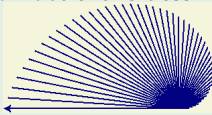
permet de créer une variable parcourant les entiers de 0 à <entier> (exclu).

- Les <instructions> indentées qui suivent seront exécutées pour chaque valeur prise par la variable.
- La boucle for permet donc de répéter un nombre prédéfini de fois des instructions, on dit que c'est une boucle bornée.

C3 Initiation à Python avec turtle

Une boucle avec la fonction trait

En faisant varier la direction et la longueur dans la fonction `trait` définie ci-dessus à l'aide d'une boucle on obtient le dessin suivant :



```
1 direction = 0
2 longueur = 10
3 for t in range(51):
4     trait(0,0, longueur , direction)
5     direction+=3.6
6     longueur+=5
7 papier.exitonclick()
```

Instructions conditionnelles

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>.

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.

Exemples

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

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.

Exemples

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

```
1 if erreur==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.

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