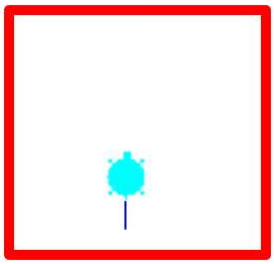
La boucle Tant que

1- Découverte de la boucle Tant que

Exemple 1:



La tortue est lâchée au hasard dans un carré rouge.
La distance entre la tortue et le bord du carré est donc inconnue.
Elle est dirigée vers le nord. A chaque déplacement, elle avance d'un pas.
La tortue s'arrête quand elle rencontre la ligne rouge.

Pierre a écrit un algorithme qui simule le déplacement de la tortue.

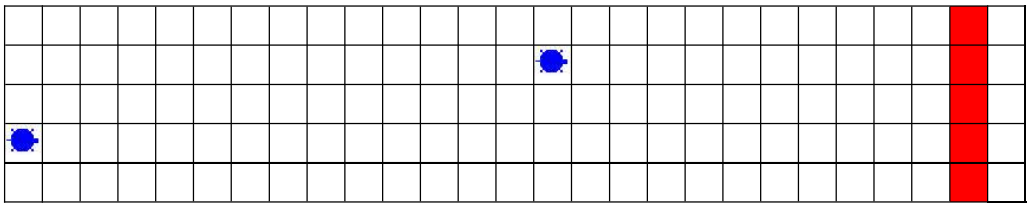
Tester l'algorithme de Pierre dans les cas suivants:

```
n=20
Pour i=1 à n par pas de 1 faire
    Si tortue ≠ rouge alors
        tortue avance(1)
    Fin Si
Refaire
```

La tortue est lâchée à 11 pixels du bord du carré.

La tortue est lâchée à 25 pixels du bord du carré.

La simulation de Pierre est-elle correcte? Sinon pour quelle(s) raison(s) échoue-t-elle? Que faudrait-il modifier?

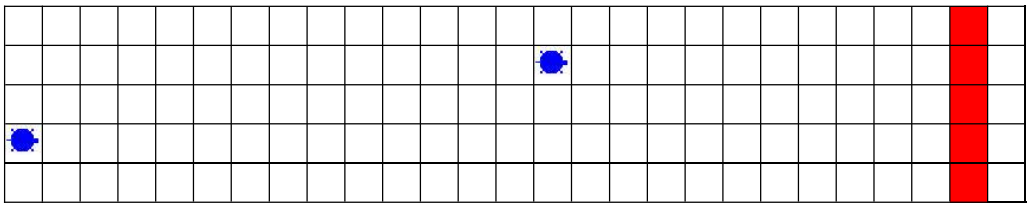


Sonia propose l'algorithme suivant:

```
Tant que tortue ≠ rouge
faire
        tortue avance(1)
```

Tester l'algorithme de Sonia dans les cas précédents.

Quels sont les avantages de cet algorithme?



Sonia a utilisé une **boucle Tant que** qui remplace la **boucle Pour** quand on ne connaît pas le nombre d'itérations.

On a implémenté cet algorithme. Cf CarreRouge.py du dossier WORKSPACE/BoucleTantque

Exemple 2:

On répète les trois instructions suivantes:

- On tire un entier n compris entre 0 et 25.
- La tortue avance de n pixels.
- La tortue tourne à droite de 15°.

Elle s'arrête définitivement quand n=0.

Parmi les propositions suivantes utilisant la boucle *Tant que*, lesquelles ne fonctionnent pas? Pourquoi?
Que doit-on vérifier impérativement avant d'exécuter l'agorithme?

-1-

```
Tant que n ≠ 0 faire
        Tirage de n
        tortue avance(n)
        tortue droite(15)
Fin tant que
```

-2-

```
Tirage de n
Tant que n ≠ 0 faire
        tortue avance(n)
        tortue droite(15)
        Tirage de n
Fin tant que
```

-3-

```
Tirage de n
Tant que n ≠ 0 faire
        tortue avance(n)
        tortue droite(15)
Fin tant que
```

La boucle Tant que

La boucle while est utilisée quand on ne connaît pas le nombre d'itérations à effectuer et quand on n'est pas sûr que la boucle sera au moins effectuée une fois.

```
Tant que test est vrai faire
    instructions
Fin tant que
```

```
while (test):
    instructions
```

Toujours vérifier que les variables utilisées pour le test :

- **ont été initialisées avant la boucle**
- **sont modifiées dans la boucle.**

Si ce n'est pas le cas, soit le programme ne peut pas se lancer, soit il initialise les variables avec des valeurs quelconques, soit il ne s'arrêtera jamais.

Conditions d'utilisation:

- ***Une série d'instructions se répète.***
- ***Le nombre de répétitions (d'itérations) est connu ou inconnu.***
- ***La série d'instructions peut ne jamais être exécutée.***

Transformer une boucle **Pour** en boucle **Tant que**

Langage naturel

```
Pour i=départ à fin par pas de p faire
    instructions
Fin Pour
```

⇳

```
i=départ
Tant que i<= fin faire
    instructions
    i=i+p
Fin tant que
```

Langage Python

```
for i in range (départ,N,p):
    instructions
```

⇳

```
i=départ
while (i<N):
    instructions
    i=i+p
```

```
for i in range (N):
    instructions
```

⇳

```
i=0
while (i<N):
    instructions
    i=i+1
```

2- Lancer TigerJython.

Ouvrir le fichier hasard.py du dossier BoucleTantque et le compléter par le programme 2 de l'exemple 2.

L'instruction **int (random.random()*N)** donne un entier aléatoire de l'intervalle [0;N**[**.

L'instruction **int (random.random()*N)+5** donne un entier aléatoire de l'intervalle [5;N+5**[**.

Par exemple, **int (random.random()*101)+5** donne un entier aléatoire de l'intervalle [5;100**]**.

Exécuter le programme.

3- *Réaliser les défis suivants:*

Défi A01: Dessiner un carré dont les côtés mesurent 100 pixels. Appeler votre fichier DefiA01.

Défi A02: Dessiner un pentagone de côté 100 pixels.

Défi A03: Dessiner un hexagone de côté 100 pixels.

La boucle Tant que

Défi A04: Dessiner un octogone de côté 100 pixels.

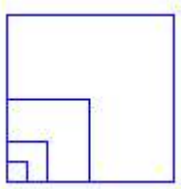
Défi A05: Dessiner une spirale telle que l'angle formé par deux côtés consécutifs soit de 30° et la longueur de chaque côté est un entier aléatoire d de [5;40]. La quantité de couleur est limitée à 400 pixels. On affichera la valeur de d et la quantité de couleur restante.

On pourra utiliser les instructions:

joe.addStatusBar(50) qui affiche une zone de texte en bas de la fenêtre de hauteur 50 pixels,

et **joe.setStatusText("d= "+str(d)+" pixels")** qui affiche le texte dans la zone de texte.

ou **print("d= "+str(d)+" pixels")**



Défi A06: On dessine un carré dont le côté est un nombre entier aléatoire de [50; 350]. On dessine un second carré dont le côté mesure la moitié du précédent et on répète l'opération tant que le côté est supérieur à 5 . La position initiale de la tortue est (-175;-175) (cf figure)

Défi A07: La tortue Joe est placée sur l'une des extrémités d'un chemin rouge. L'autre extrémité du chemin est en vert. Joe doit suivre la ligne. (contraintes: la tortue se déplace de 2 pixels en 2 pixels et tourne de 90°). Ouvrir le fichier DefiA07.java et le compléter.

Défi A08: La tortue Joe est placée sur l'une des extrémités d'un chemin rouge. L'autre extrémité du chemin est en vert. Joe doit mesurer la ligne rouge au pixel près. (contraintes: la tortue se déplace de 2 pixels en 2 pixels et tourne de 90°)