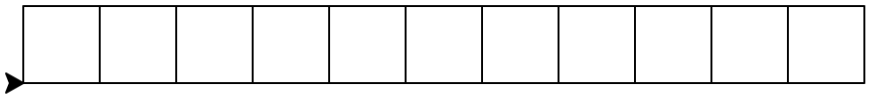
Tests



On a vu précédemment qu'une ligne de carrés était construite à l'aide d'une répétition codée par

```
def carre(cote):
    for _ in range(4):
        forward(cote)
        left(90)

def ligne(nbcarre,cote):
    for _ in range(nbcarre):
        carre(cote)
        forward(cote)
    #on remet la tortue dans
    #son état initial
    back(nbcarre*cote)
```

Pour insister que la boucle for est une simple boucle de répétitions on utilise le caractère _

On voudrait maintenant dessiner une ligne bicolore

Cette fois ci on va utiliser un compteur de boucle `i` dans la boucle de telle sorte que `i` est le nombre de construction d'un carré (0 étant associé au carré le plus à gauche possible) :

Le carré d'indice 0 a pour couleur noire, le carré d'indice 1 a pour couleur blanche, puis le carré d'indice 2 a pour couleur noire etc...

On peut résumer ainsi :

Si l'indice du carré est **pair alors** le carré est de couleur noire, **sinon** il est de couleur blanche

que l'on traduit en Python

```
BLANC = 'white'
NOIR = 'black'
def ligne_bicolore(nbcarre,cote):
    for i in range(nbcarre):
        # si i est pair
        if i % 2 == 0:
            filling(NOIR)
        else:
            filling(BLANC)
        begin_fill()
        carre(cote)
```

```
        end_fill()
        forward(cote)
    #on remet la tortue dans
    #son état initial
    back(nbcarre*cote)
```

`i % 2` est le reste de la division euclidienne par 2, donc 0 ou 1, par exemple

```
>>> 12 % 2
0
>>> 11 % 2
1
```

`i % 2 == 0` est une **expression booléenne** soit vraie soit fausse.

```
>>> 12 % 2 == 0
True
>>> 11 % 2 == 0
False
```

1 Test : Si ... Alors ... Sinon Si ... Alors

Dans le cas précédent on n'évalue qu'une expression booléenne, parfois on aimerait en évaluer plusieurs comme dans le cas suivant :

Les différentes mentions possibles au bac en fonction de la moyenne générale m à l'épreuve sont

1. Si la moyenne m vérifie $m \geqslant 16$ dans ce cas la mention est Très Bien
2. Si la moyenne m vérifie $14 < m \leqslant 16$ dans ce cas la mention est Bien
3. Si la moyenne m vérifie $12 < m \leqslant 14$ dans ce cas la mention est Assez Bien
4. Si la moyenne m vérifie $10 < m \leqslant 12$ dans ce cas la mention Passable
5. Si la moyenne m vérifie $8 < m \leqslant 10$ dans ce cas l'élève passe un oral de rattrapage
6. Si la moyenne m vérifie $m < 8$ dans ce cas l'élève n'a pas le diplôme

Dans un premier temps on pourrait écrire en Python :

```
def mention_bac(moyenne):
    if moyenne >= 16:
       mention =  "Mention Très Bien"
    if 14 <= moyenne < 16:
        mention =  "Mention Bien"
    if 12 <= moyenne < 14:
        mention =  "Mention Assez Bien"
    if 10 <= moyenne < 12:
        mention =  "Mention Passable"
    if 8 <= moyenne < 10:
        mention =  "Oral Second Tour"
```

```
    if moyenne < 8:
        mention =  "Refusé"
    return mention
```

Si l'on visualise sur Python Tutor pour une moyenne de 16 on observe que toutes les expressions booléennes sont évaluées, alors que **la première évaluation étant vraie, les autres n'auraient pas dû être évaluées** :

D'où cette nouvelle approche :

```
def mention_bac(moyenne):
    if moyenne >= 16:
        mention =  "Mention Très Bien"
    else:
        if 14 <= moyenne < 16:
            mention =  "Mention Bien"
        else:
            if 12 <= moyenne < 14:
                mention =  "Mention Assez Bien"
            else:
                if 10 <= moyenne < 12:
                    mention =  "Mention Passable"
                else:
                    if 8 <= moyenne < 10:
                        mention =  "Oral Second Tour"
                    else:
                        if moyenne < 8:
                            mention =  "Refusé"
    return mention
```

Le langage Python permet de contracter `else if` en `elif`.

Voici **la syntaxe du langage Python** nous permettant d'évaluer plusieurs expressions booléennes et en fonction du résultat exécuter des instructions ou pas :

8.1. L'instruction if

L'instruction `if` est utilisée pour exécuter des instructions en fonction d'une condition :

```
if_stmt ::=  "if" assignment_expression ":" suite
             ("elif" assignment_expression ":" suite)*
             ["else" ":" suite]
```

Elle sélectionne exactement une des suites en évaluant les expressions une par une jusqu'à ce qu'une soit vraie (voir la section Opérations booléennes pour la définition de vrai et faux) ; ensuite cette suite est exécutée (et aucune autre partie de l'instruction `if` n'est exécutée ou évaluée). Si toutes les expressions sont fausses, la suite de la clause `else`, si elle existe, est exécutée.

On va donc écrire une fonction `mention_bac(moyenne)` qui retourne une chaîne de caractères désignant la mention obtenue en fonction de la moyenne obtenue

```
from random import randint
def mention_bac(moyenne):
```

```
    if moyenne >= 16:
       mention =  "Mention Très Bien"
    elif 14 <= moyenne < 16:
        mention =  "Mention Bien"
    elif 12 <= moyenne < 14:
        mention =  "Mention Assez Bien"
    elif 10 <= moyenne < 12:
        mention =  "Mention Passable"
    elif 8 <= moyenne < 10:
        mention =  "Oral Second Tour"
    elif moyenne < 8:
        mention =  "Refusé"
    return mention

moy = randint(0,20)
print("la moyenne est ",moy," la mention est ",mention_bac(moy))
```

Quand on teste on observe que :

1. Tous les expressions booléennes vont être évaluées tant que la moyenne ne sera pas "encadrée" ainsi si la moyenne est de 6, toutes les expressions booléennes seront évaluées
2. Par contre si la note est de 13 les expressions `10 <= moyenne < 12")`, `8 <= moyenne < 10")` et `moyenne < 8")` ne seront pas évaluées.

Enfin si on utilise `return` directement après l'évaluation de l'expression booléenne :

```
def mention_bac(moyenne):
    if moyenne >= 16:
        return "Mention Très Bien"
    if 14 <= moyenne < 16:
        return "Mention Bien"
    if 12 <= moyenne < 14:
        return "Mention Assez Bien"
    if 10 <= moyenne < 12:
        return "Mention Passable"
    if 8 <= moyenne < 10:
        return  "Oral Second Tour"
    if moyenne < 8:
        return "Refusé"
```

2 Exercices

Ex 1

1. Corriger les erreurs dans le programme suivant

```
      x = 5
if x >= 2
print("Pi")
else
print("Thon")
```

2. Une fois corrigé et exécuté qu'y aura-t-il d'affiché à l'écran ?

Ex 2

1. Corriger les erreurs dans le programme suivant

```
x = 35
if x >= 40:
print(Bleu)
elif x >= 30:
print(Blanc)
elif x >= 20
print(Rouge)
```

2. Une fois corrigé et exécuté qu'y aura-t-il d'affiché à l'écran ?

Ex 3

Définir une fonction `position(x,a,b)` qui renvoie "plus petit que a" si x $<$ a, "compris entre a et b si $x \in [a;b]$ et "plus grand que b" si x $>$ b

Ex 4

Une année est bissextile si elle est un multiple de 4 mais pas de 100 ou si elle est un multiple de 400

Par exemple 2020 est bissextile car $2020 = 4 \times 505$, 2100 n'est pas bissextile car $2100 = 4 \times 525 = 21 \times 100$ et 2100 n'est pas un multiple de 400

Définir une fonction Python `est_bissextile(annee)` qui renvoie Vrai si l'année `annee` est bissextile, Faux sinon

Ex 5 : Inégalité triangulaire

Etant donné un triangle ABC alors chaque côté de ce triangle est plus petit ou égal à la somme des deux autres côtés

1. Donner un exemple de trois nombres positifs a, b et c tels que $a \leqslant b + c$ avec $b > a + c$
2. Définir une fonction `est_triangle(a,b,c)` qui renvoie vrai si les nombres a, b et c sont les longueurs possibles des côtés d'un triangle et faux sinon.

Ex 6

Définir une fonction `est_equilateral(a,b,c)` qui renvoie vrai si le triangle ayant pour longueurs `a`, `b` et `c` est équilatéral et faux sinon

Ex 7

Définir une fonction `est_isocele(a,b,c)` qui renvoie vrai si le triangle ayant pour longueurs `a`, `b` et `c` est isocèle et faux sinon

Ex 8

Quelles sont les différences entre les trois programmes

Programme 1

```
x = 35
if x >= 40:
    x = 2*x
elif x >= 30:
    x = x + 2
elif x >= 20:
    x = x - 2
else:
    x = x**2
```

Programme 2

```
x = 35
if x >= 40:
    x = 2*x
elif x >= 30:
    x = x + 2
elif x >= 20:
    x = x - 2
elif x >= 10
    x = x**2
```

Programme 3

```
x = 35
if x >= 40:
    x = 2*x
else:
    if x >= 30:
        x = x + 2
    if x >= 20:
        x = x - 2
    else:
        x = x**2
```

Changer la valeur initiale de x

Ex 9

Définir une fonction Python `max(a,b)` qui renvoie le plus grand des deux nombres a et b

Ex 10

Voici la règle d'un jeu de dés :

On lance quatre dés à six faces, si il y a au moins un six on gagne sinon on perd

Définir une fonction `gagne()` qui renvoie Vrai si on gagne à ce jeu et faux sinon

Ex 11

Ecrire une fonction `est_premier(n)` qui renvoie vrai si l'entier `n` est premier et faux sinon.

Ex 12

Ecrire une fonction `intersection(a,b,c,d)` qui renvoie Vrai si $[a, b] \cap [c, d] \neq \phi$ et Faux sinon.

Par exemple

```
>>> intersection(1,2,3,4)
False
>>> intersection(1,4,2,3)
True
```