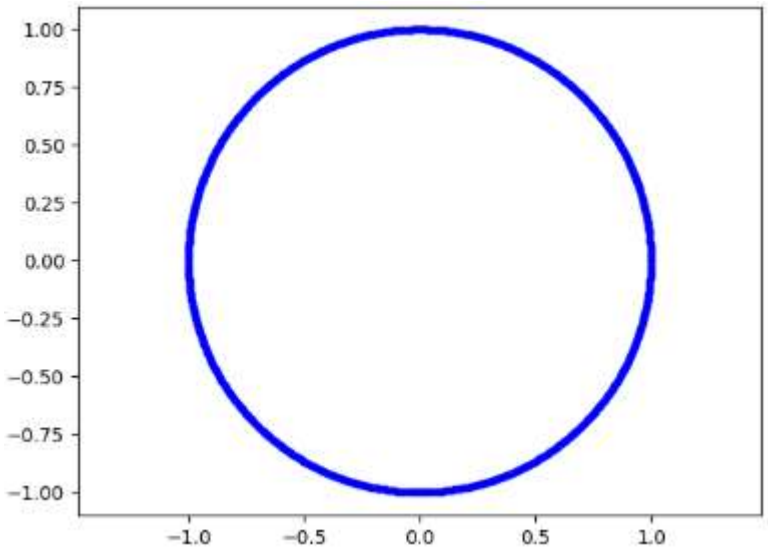
Tracé du cercle trigonométrique

On propose ci-dessous une fonction « cercle(n) » affichant « n » points situés sur le cercle trigonométrique. Compléter le script afin d'obtenir ce cercle.

```
1  def cercle(n):
2    for i in range(...):
3      angle=...
4      x=...
5      y=...
6      plt.plot(x,y,"b.")
7    plt.axis("equal")
8    plt.show()
```



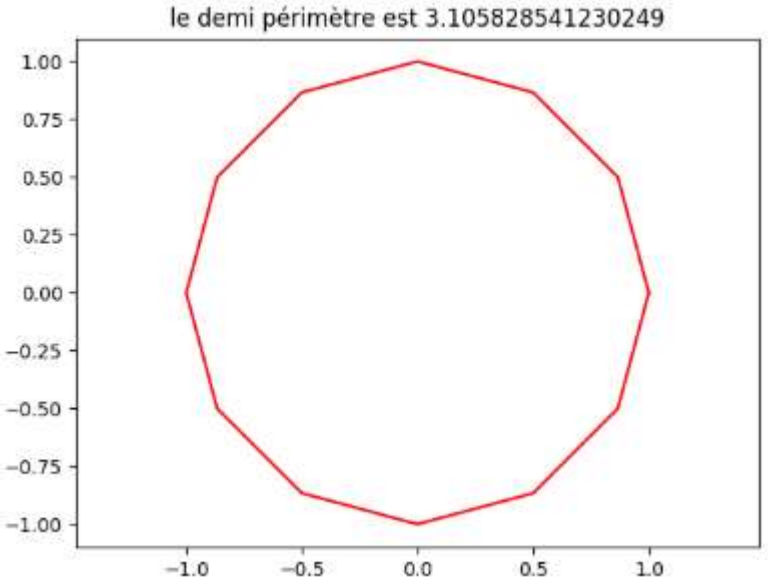
Modifier le script afin qu'il affiche les quatre cadrans du cercle avec quatre couleurs différentes.

Demi-périmètre d'un polygone régulier

Dans un repère (O, I, J) on considère le polygone régulier à « n » côtés.

Compléter le script de la fonction « coordpolygone(n) » permettant de stocker dans deux listes les coordonnées des sommets du polygone.

```
5  def coordpolygone(n):
6    lx=[]
7    ly=[]
8      for i in range(n):
9      angle=...
10     lx.append(...)
11     ly.append(...)
12   return lx,ly
```



On souhaite écrire une fonction « perimetre(n) » qui prend pour argument le nombre de sommets du polygone et renvoie le périmètre de ce polygone. On rappelle la formule :

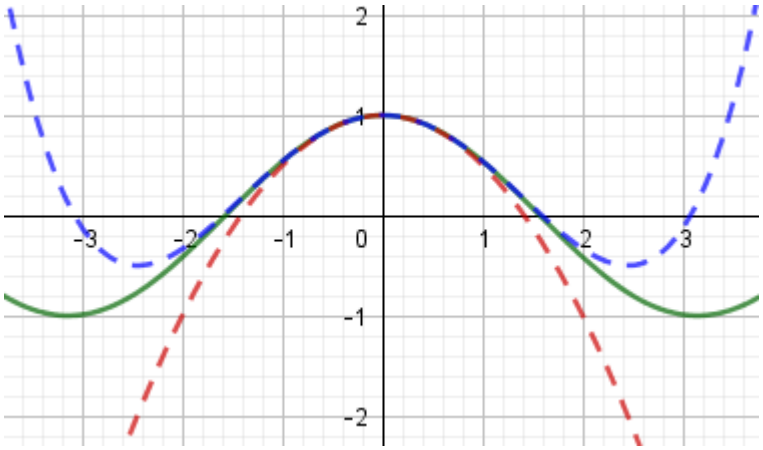
$$AB=\sqrt{\left(x_B-x_A\right)^2+\left(y_B-y_A\right)^2}$$

```
15 def perimetre(n):
16   p=0
17   lx,ly=coordpolygone(n)
18   lx.append(lx[0])
19   ly.append(ly[0])
20   for i in range(n):
21     p=...
22   return p
```

Compléter la ligne 21 du script précédent puis insérer les deux fonctions dans un algorithme permettant d'afficher le polygone ainsi que son demi-périmètre.

```
25 def archimede(n):
26   lx,ly=coordpolygone(n)
27   lx.append(lx[0])
28   ly.append(ly[0])
29   plt.plot(lx,ly,"r")
30   plt.axis("equal")
31   plt.title('le demi périmètre est '+str(perimetre(n)/2))
32   plt.show()
```

Approximation du cosinus



On considère trois fonctions :

- $f(x)=\cos(x)$
- $g(x)=1-\frac{x^2}{2}$
- $h(x)=1-\frac{x^2}{2}+\frac{x^4}{24}$

On propose ci-contre un algorithme que l'on exécute et on observe l'affichage suivant :

```
Powered by trinket
0.8
1.4
```

```
from math import *

def g(x):
    return 1-x**2/2

def h(x):
    return 1-x**2/2+x**4/24

def erreurcommise(f,x):
    return abs((cos(x)-f(x)))

def domainevalid(f,e):
    x=0
    while erreurcommise(f,x)<e:
        x=x+0.1
    return x

print(domainevalid(g,0.01))
print(domainevalid(h,0.01))
```

Sauriez-vous déterminer à quoi correspondent ces deux résultats ? Déterminer de manière précise à quoi peut servir cet algorithme.

On admet la formule suivante :

$$\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$$

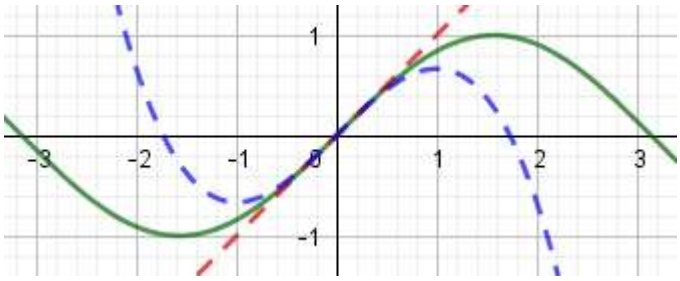
Dans cette expression, $n!$ représente la « factorielle de n » définie par $n!=n\times(n-1)\times\ldots\times2\times1$.

Le but est de construire un algorithme permettant de calculer une valeur approchée du cosinus d'un nombre réel x à l'aide de la formule (développement limité du cos en 0).

```
def factorielle(n):
    if n==0:
        return(1)
    else:
        return(n*factorielle(n-1))
```

On construira pour cela une fonction « approximationcos(x,n) » prenant pour arguments un angle et le nombre d'itérations souhaitées et affichant une valeur approchée du cosinus de l'angle.

Approximation du sinus



On donne ci-dessous le développement limité de la fonction sinus en 0.

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots$$

Modifier la fonction précédente en « approximationsin(x,e) » prenant pour arguments un angle et un nombre d'itérations souhaitées et affichant une valeur approchée du sinus de l'angle.