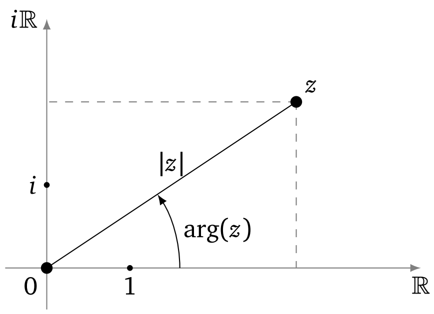
Nombres complexes II

On poursuit l'exploration des nombres complexes en se concentrant sur la forme module/argument.

Cours 1 (Nombres complexes).
Module/argument. Tout nombre complexe $z \in \mathbb{C}^*$, s'écrit :

$$z = r\,(\cos\theta + i\sin\theta)$$

où
- $r = |z|$ est le module de z,
- et $\theta \in \mathbb{R}$ est un ***argument***.



Unicité. Si θ est un argument, alors n'importe quel $\theta + 2k\pi$ est aussi un argument.
Pour éviter cette indécision, on peut imposer à θ d'appartenir à l'intervalle $]-\pi,+\pi]$, l'argument est alors unique. Pour $z \in \mathbb{C}^*$, il existe un unique couple (r,θ) avec $r > 0$ et $\theta \in]-\pi,+\pi]$ tel que :

$$z = r\,(\cos\theta + i\sin\theta).$$

Remarques.
- Une autre convention aurait été de choisir l'intervalle $[0, 2\pi[$.
- L'écriture (r,θ) s'appelle aussi l'écriture en coordonnées polaires d'un nombre complexe, par opposition à l'écriture $z = a + ib$ qui est l'écriture cartésienne.

Cours 2 (Module `cmath`).
Le module `cmath` fournit des outils supplémentaires pour les nombres complexes. Pour éviter les conflits avec le module `math` nous l'importerons par :

```
import cmath
```

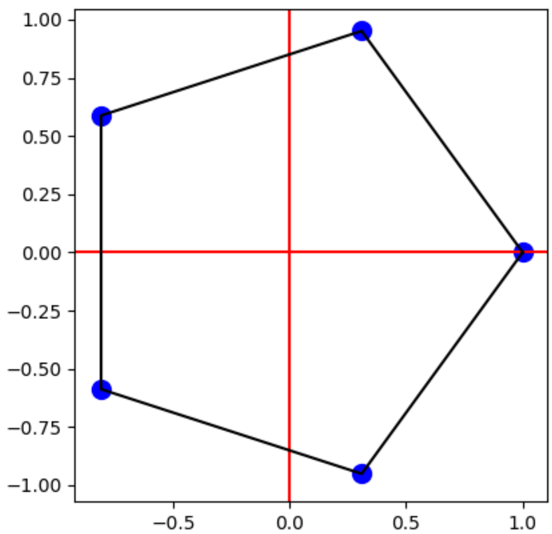
1. `cmath.phase(z)` renvoie l'argument $\theta \in]-\pi,+\pi]$ du nombre complexe z. Exemple : `cmath.phase(1-1j)` renvoie `-0.785...` qui correspond à la valeur $-\frac{\pi}{4}$.
2. Rappel : `abs(z)` renvoie le module $|z|$ (c'est une fonction interne à `Python`).

3. `cmath.polar(z)` renvoie le couple module/argument (r, θ). Exemple : `cmath.polar(1-1j)` renvoie `(1.414..., -0.785...)` qui correspond au couple $(r, \theta) = (\sqrt{2}, -\frac{\pi}{4})$.
4. `cmath.rect(r,theta)` renvoie le nombre complexe dont le module est r et l'argument θ. Exemple : `cmath.rect(2,pi/4)` renvoie `1.414... + 1.414...j` et correspond à $\sqrt{2} + i\sqrt{2}$.

Activité 1 (Module/argument)**.**

Objectifs : utiliser `Python` *pour calculer et mieux comprendre la forme module/argument.*

1. Pour un nombre complexe z, par exemple $z = 1 + 3i$ ou $z = 1 + i$, calcule son module et son argument à l'aide de `Python`.
2. Quel nombre complexe a pour module 2 et argument $\frac{\pi}{3}$? Même question pour le complexe de module 3 et d'argument $\frac{3\pi}{2}$. Essaie de deviner la réponse exacte à partir des valeurs approchées données par `Python`.
3. À l'aide du module `matplotlib`, place le point d'affixe z dont on te donne le module et l'argument, par exemple de module $\sqrt{2}$ et argument $\frac{\pi}{6}$.
4. Soit $n \geqslant 3$. Soit ω le nombre complexe de module 1 et d'argument $\frac{2\pi}{n}$. Trace le polygone ayant pour sommets les points d'affixes :
$$1, \omega, \omega^2, \ldots, \omega^{n-1}.$$
Quelle est la nature de ce polygone ?



Activité 2 (Module/argument (suite))**.**

Objectifs : créer tes propres fonctions qui permettent la conversion entre l'écriture cartésienne d'un nombre complexe et son écriture sous la forme module/argument.

Tu vas écrire tes propres fonctions pour calculer avec les modules et les arguments.

1. Programme une fonction `polaire_vers_cartesien(module,argument)` qui renvoie le nombre complexe z (sous la forme d'un nombre complexe `Python`) dont le module et l'argument sont donnés. Utilise la formule
$$z = r\cos\theta + ir\sin\theta.$$
Compare ton résultat avec la fonction `rect` du module `cmath`.

2. Programme une fonction `cartesien_vers_polaire(z)` qui renvoie le module et l'argument du nombre complexe z. Récupère d'abord la partie réelle x et la partie imaginaire y de z. Le module est alors facile à calculer. L'argument se calcule par la formule :
$$\theta = \operatorname{atan2}(y, x)$$
La fonction `atan2` est une variante de la fonction « arctangente » et est disponible dans le module `math`.
Compare ta fonction avec les fonctions `phase` et `polar` du module `cmath`.
3. Programme une fonction `argument_dans_intervalle(angle)` qui renvoie une mesure de l'angle dans l'intervalle $]-\pi, +\pi]$. Par exemple soit $\theta = \frac{5\pi}{2}$, comme $\theta = -\frac{\pi}{2} + 3 \cdot 2\pi$ alors $\theta' = -\frac{\pi}{2}$ est la mesure de l'angle dans l'intervalle $]-\pi, +\pi]$.
Indication. Commence par ramener l'angle dans l'intervalle $[0, 2\pi[$, puis discute selon la valeur.
Une fois terminé compare ton résultat avec la commande `angle % 2*pi`.

Cours 3 (Notation exponentielle)**.**
- **Notation exponentielle.** On note
$$e^{i\theta} = \cos(\theta) + i\sin(\theta).$$
C'est donc le nombre complexe de module 1 et d'argument θ.
- **Formules d'Euler.** Un petit calcul conduit à :
$$\cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2} \qquad \text{et} \qquad \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}.$$
- **Formule de Moivre.**
$$(\cos\theta + i\sin\theta)^n = \cos(n\theta) + i\sin(n\theta).$$
Avec la notation exponentielle, l'écriture de cette formule est très simple :
$$\left(e^{i\theta}\right)^n = e^{in\theta}.$$

Activité 3 (Euler, de Moivre, Gauss)**.**

Objectifs : mettre en œuvre plusieurs formules.

1. **Euler.** Programme deux fonctions `cosinus(t)` et `sinus(t)` qui calculent et renvoient le cosinus et le sinus d'un réel t donné en utilisant les formules d'Euler.
Indication. Utilise ta fonction `polaire_vers_cartesien()` de l'activité 1 pour calculer e^{it}.
Exemple. Retrouve le sinus et le cosinus de $t = \frac{\pi}{6}$.
2. **de Moivre.** Programme une fonction `puissance_bis(z,n)` qui calcule z^n à l'aide de la formule de Moivre selon le principe suivant :
   - Écrire z sous la forme $z = re^{i\theta}$ (utilise ta fonction `cartesien_vers_polaire()`).
   - Calculer r^n et $n\theta$.
   - Renvoyer z^n grâce à la formule de Moivre $z^n = r^n e^{in\theta}$ (utilise ta fonction `polaire_vers_cartesien()`).

   Exemple. Calcule $(2-3i)^{10}$.
Complexité. La formule de Moivre permet de remplacer le calcul d'une puissance d'un nombre complexe par le calcul de la puissance d'un nombre réel (son module).
3. **Gauss.** Comment calculer plus rapidement le produit de deux nombres complexes ? Soit $z = a + ib$

et $z' = c + id$. La formule naïve donnée par la définition est :

$$z \times z' = (ac - bd) + i(ad + bc).$$

Pour calculer un produit de deux nombres complexes, il faut donc calculer le produit de 4 nombres réels : ac, bd, ad, bc.

Nous allons voir deux méthodes, dues à Gauss, qui ne nécessitent que 3 multiplications de nombres réels.

Méthode 1. Calculer $r = ac$, $s = bd$, $t = (a+b)(c+d)$, alors $z = (r-s) + i(t-r-s)$.

Méthode 2. Calculer $r = c(a+b)$, $s = a(d-c)$, $t = b(c+d)$, alors $z = (r-t) + i(r+s)$.

Programme trois fonctions du type `multiplication(a,b,c,d)` qui renvoient la partie réelle et la partie imaginaire de $(a+ib) \times (c+id)$ par les trois méthodes décrites ici. Teste tes fonctions en calculant $(2+5i) \times (3-2i)$.

Activité 4 (Cercles et droites)**.**

Objectifs : tracer des cercles et des droites en utilisant les nombres complexes.

1. Programme une fonction `affiche_liste(zliste)` qui trace et affiche les points d'affixe z donnés dans la liste.

   Indication. Utilise la commande `plt.scatter(x,y)` provenant de `matplotlib`. C'est encore mieux si tu autorises les arguments optionnels avec une entête du type `affiche_liste(zliste,couleur='blue',taille=10)`.

2. Programme une fonction `trace_cercle(z0,r)` qui renvoie une liste de complexes z appartenant au cercle centré en z_0 et de rayon r.

   Indications.

   - Ces complexes z vérifient $|z - z_0| = r$ et sont donc de la forme :
   $$z = re^{2i\pi\theta}, \qquad 0 \leqslant \theta < 1.$$
   - C'est mieux d'avoir en argument optionnel le nombre de points avec une entête du type `trace_cercle(z0,r,numpoints=100)`.
   - Trace le cercle à l'aide de ta fonction `affiche_liste()`.

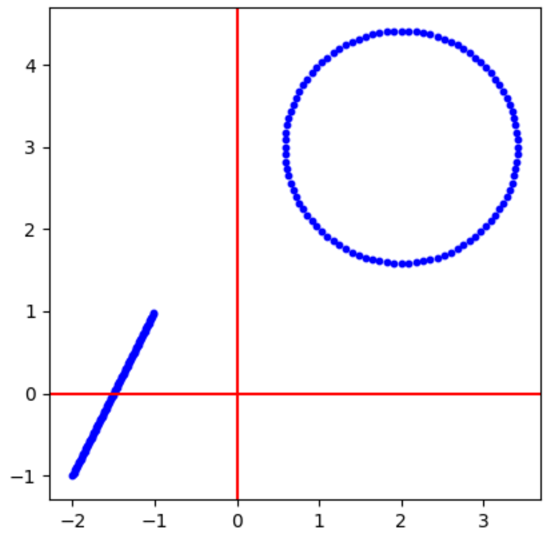


Figure. Voici le cercle de centre $2 + 3i$ et de rayon $\sqrt{2}$, ainsi que le segment entre les points d'affixes $-2 - i$ et $-1 + 3i$.

3. Programme une fonction `trace_segment(z0,z1)` qui renvoie une liste de complexes z appartenant au segment $[z_0, z_1]$.

   Indications.

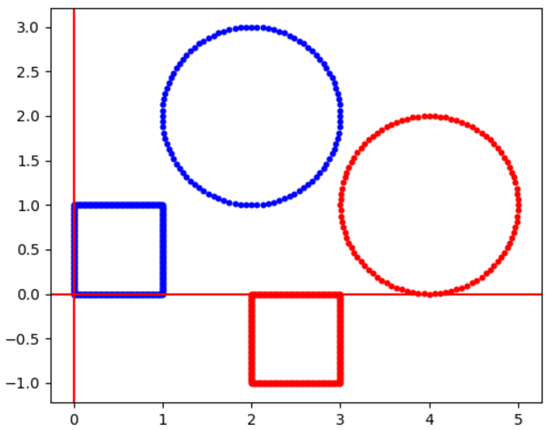
   - Ces complexes z vérifient $z \in [z_0, z_1]$ et sont donc de la forme :
   $$z = (1-t)z_0 + tz_1, \qquad 0 \leqslant t \leqslant 1.$$
   - C'est mieux d'avoir le nombre de points en argument optionnel avec une entête du type `trace_segment(z0,z1,numpoints=100)`.
   - Trace le segment à l'aide de ta fonction `affiche_liste()`.

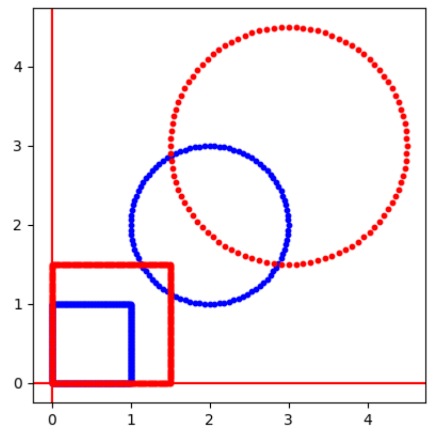
Activité 5 (Transformations du plan).

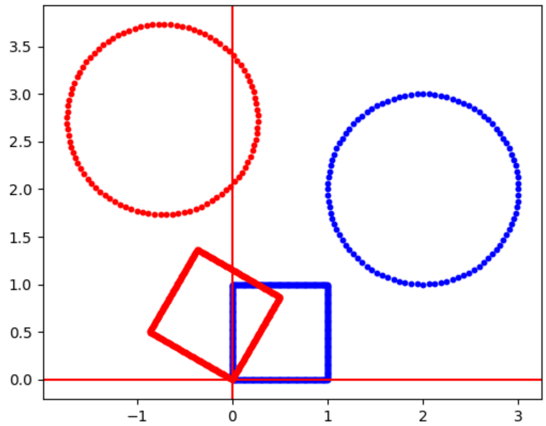
Objectifs : définir des transformations du plan à l'aide des nombres complexes.

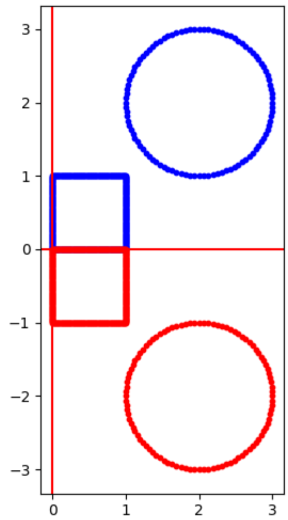
1. Programme les fonctions suivantes. Chaque fonction est du type `transformation(zliste)` et renvoie la liste des $f(z)$ pour z parcourant la liste donnée :
   - une fonction `translation(zliste,v)` qui correspond à la translation $z \mapsto z + v$, où $v \in \mathbb{C}$ est fixé,
   - une fonction `homothetie(zliste,k)` qui correspond à l'homothétie de centre 0 et de rapport $k \in \mathbb{R}$: $z \mapsto kz$,
   - une fonction `rotation(zliste,theta)` qui correspond à la rotation d'angle θ, centrée en 0 : $z \mapsto ze^{i\theta}$,
   - une fonction `symetrie(zliste)` qui correspond à la symétrie axiale $z \mapsto \bar{z}$.

   Affiche ensuite l'image d'un cercle et d'un carré pour chacune de ces transformations (un carré est formé de quatre segments !). Ci-dessous un cercle et un carré (en bleu) et leur image pour chaque transformation (en rouge).





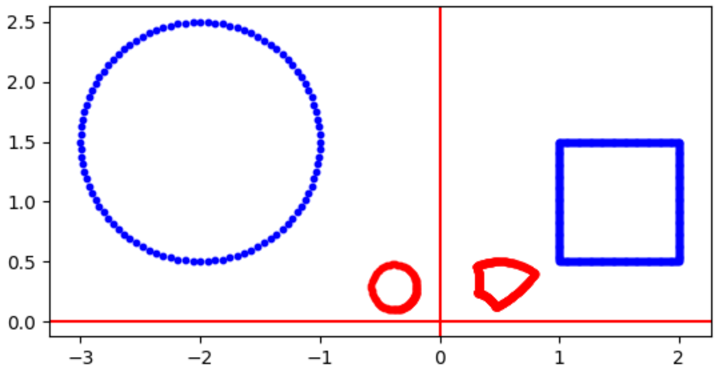




2. Programme une fonction `inversion(zliste)` qui correspond à l'inversion qui est l'application $z \mapsto \frac{1}{z}$ (pour $z \in \mathbb{C}^*$).

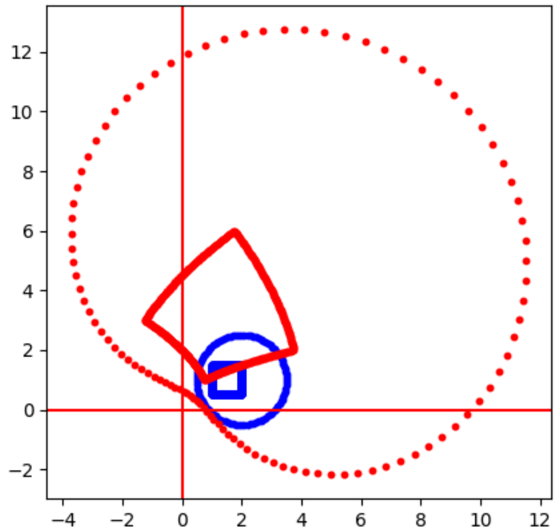
   En particulier essaie de conjecturer en quoi est transformée une droite, en quoi est transformé un cercle (les cas où la droite ou le cercle passent par l'origine sont spéciaux).

   Ci-dessous un cercle et un carré (en bleu) et leur image par l'inversion (en rouge).



3. Programme une fonction `au_carre(zliste)` qui correspond à l'application $z \mapsto z^2$.

   Ci-dessous un cercle et un carré (en bleu) et leur image (en rouge).


12
10
8
6
4
2
0
−2
−4
−2
0
2
4
6
8
10
12