Python est un langage très utilisé dans le domaine scientifique, comme le montre par exemple le choix de SAGE. Et les sciences, en particulier, font grand usage des nombres complexes, essentiellement depuis leur choix par Cauchy. Les physiciens et les électriciens notant *j* le nombre complexe dont le carré vaut -1, *Python* suit ce choix.

Sections

Instanciation d'un nombre complexe

Dans *Python*, il suffit d'écrire *complex(x,y)* pour avoir un nombre complexe :

```
z=complex(4,3) print(z)
```

Même si les coordonnées *x* et *y* du point sont entières ou des fractions, elles deviennent des réels lorsque *Python* instancie le complexe. Voir les propriétés du complexe ci-dessous pour le vérifier.

Si on veut quand même que la lettre *i* désigne le complexe de carré -1, il suffit de le déclarer comme tel :

```
i=complex(0,1) print(i**2)
```

Opérations

Les quatre opérations se notent respectivement +, -, * et /, et donnent toujours un complexe, même si celui-ci est réel (exemple de la soustraction ci-dessous) :

```
a=complex(2,3) b=complex(4,3) print(a+b) print(a-b) print(a*b) print(a/b)
```

L'élévation à un exposant se note de la même manière que pour les autres nombres, par **. Mais l'exposant peut même être un complexe!

```
i=complex(0,1) print(i**i)
```

On constate que ...

La racine carrée d'un complexe peut aussi s'obtenir par une élévation de celui-ci à la puissance 0,5 mais dans ce cas on n'obtient qu'une seule des deux racines carrées :

```
c=complex(7,24) print(c**0.5)
```

Mais *-4-3i* a aussi pour carré *7+24i*. Comment fait *Python* pour choisir entre les deux racines carrées?

Même -1 a deux racines carrées dans , et comme on s'en doute, *Python* ne choisit pas *-i* mais *i*... ou plutôt un complexe proche de celui-ci :

```
print((-1)**0.5)
```

Propriétés d'un nombre complexe

Les parties réelle et imaginaire d'un complexe sont des propriétés de l'objet :

```
z=complex(4,3) print(z.real) print(z.imag)
```

Par contre, le conjugué d'un complexe est une méthode de celui-ci :

```
z=complex(4,3) print(z.conjugate())
```

(on remarque la présence des parenthèses après *conjugate*)

Forme trigonométrique

Pour avoir le module d'un nombre complexe, on entre *abs* :

```
z=complex(4,3) print(abs(z))
```

Bien entendu, le résultat est réel.

Cependant, pour avoir l'argument de *a*, il faut charger le module (c'est le cas de le dire!) *cmath* :

```
from cmath import * z=complex(4,3) print(phase(z))
```

On remarque que *Python* utilise le mot *phase* et non le mot *argument*. *cmath* permet aussi de calculer d'un coup le module et l'argument d'un nombre complexe avec *polar* :

```
from cmath import * z=complex(4,3) print(polar(z))
```

Pour réaliser l'opération inverse (calculer l'exponentielle d'un nombre imaginaire), on utilise *rect* :

```
from cmath import * print(rect(2,pi/3))
```

Par exemple, si on veut calculer le plus petit angle et l'hypoténuse d'un triangle rectangle de côtés 12 cm et 5 cm, on peut faire ceci :

```
from cmath import * a=12 b=5 z=complex(a,b) print(phase(z)) print(abs(z))
```

Fonctions

Avec *cmath*, on peut appliquer certaines fonctions de la variable réelle à des complexes.

Exponentielles

Pour vérifier numériquement que , on peut utiliser l'exponentielle d'un nombre complexe (en l'occurrence, imaginaire) :

```
from cmath import * t=complex(0,pi/3) z=exp(t) print(z.real==0.5)
print(z.real-0.5) print(z.imag==sqrt(3)/2)
```

On voit que la partie réelle n'est pas tout-à-fait égale à 0,5 (la différence est minime mais non nulle), c'est encore une conséquence de la représentation binaire des nombres en machine, puisque le développement binaire de 0,5 est infini, contrairement à son développement décimal.

Le script suivant calcule et affiche les fonctions trigonométriques hyperboliques d'un complexe :

```
from cmath import * z=complex(4,3) print(cosh(z)) print(sinh(z))
print(tanh(z))
```

Logarithmes

On peut même calculer le logarithme d'un nombre complexe :

Le script suivant calcule et affiche les fonctions trigonométriques hyperboliques d'un complexe :

```
from cmath import * z=complex(4,3) print(log(z))
```

Le script suivant calcule et affiche les arguments des fonctions trigonométriques hyperboliques d'un complexe :

```
from cmath import * z=complex(4,3) print(acosh(z)) print(asinh(z))
print(atanh(z))
```

Fonctions trigonométriques

Directes

Le script suivant calcule et affiche les fonctions trigonométriques d'un complexe :

```
from cmath import * z=complex(4,3) print(cos(z)) print(sin(z))
print(tan(z))
```

Inverses

Le script suivant calcule et affiche les arcs des fonctions trigonométriques d'un complexe :

```
from cmath import * z=complex(4,3) print(acos(z)) print(asin(z))
print(atan(z))
```