GILLIARD Tallulah
LABARCHÈDE Mélody

Licence 3
MIASHS

Informatique théorique

Complexes

19/11/2015

SIGNATURE / AXIOME / COMPLEXITÉ

Signature, Axiome et Complexité pour chaque fonction. Les fonctions sont testées à la fin du programme.

__abs__(self):

Valeur absolue d'un complexe

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{R}\acute{e}el$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne $\sqrt{reel^2 + image^2}$

∀ dict { module, argument }

Alors retourne __abs__(polaireACartesien(dict))

Complexité : return, appels et opérations élémentaires → O(1)

__add__(self, other):

Addition de deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne dict'' {reel + reel', imag + imag'}

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne __add__(dict, polaireACartesien(dict'))

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne dict''

{

$\sqrt{module^2 + \ module'^2 + 2 * module * module' * \cos(argument - argument')}$,

$\operatorname{atan}\left(\frac{module*sin(argument)+module'*sin(argument')}{module*cos(argument)+module'*cos(argument')}\right)$}

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne __add__(dict, cartesienAPolaire(dict'))

Complexité : O(1)

__mul__(self, other):

Multiplication de deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne dict'' {reel * reel' – image*image', réel*image'+image*réel'}

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne __mul__(dict, polaireACartesien(dict'))

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne dict'' { $module * module', argument + argument'$}

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne __mul__(dict, cartesienAPolaire(dict'))

Complexité : O(1)

__sub__(self, other):

Soustraction de deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne __add__ (dict,oppose(dict'))

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne __add__(dict, polaireACartesien(oppose(dict')))

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne __mul__(polaireACartesien(dict),dict') }

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne __mul__(polaireACartesien(dict), polaireACartesien(dict')) }

Complexité : O(1)

__truediv__(self, other):

Division de deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne __mul__ (dict,inverse(dict'))

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne __mul__(dict, polaireACartesien(inverse(dict')))

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne __mul__(polaireACartesien(dict),inverse(dict')) }

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne __mul__(polaireACartesien(dict), inverse(polaireACartesien(dict'))) }

Complexité : O(1)

__pow__(self, other):

Puissance de deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne dict''(

cos(réel'*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(réel),réel') ,

sin(réel'*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(réel),réel'))

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne dict''(

cos(polaireACartesien(module')*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(réel, polaireACartesien(module'))) ,

sin(polaireACartesien(module')*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(réel,polaireACartesien(module'))))

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne dict''(

cos(réel'*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(polaireACartesien(module)),réel') ,

sin(réel'*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(polaireACartesien(module)),réel'))

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne dict''(

cos(polaireACartesien(module')*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(polaireACartesien(module), polaireACartesien(module'))) ,

sin(polaireACartesien(module')*(($\frac{\pi}{2} - atan2(réel, image)\ mod\ 2\pi$) * pow(__abs__(polaireACartesien(module),polaireACartesien(module'))))

Complexité : O(1)

__lt__(self, other):

Opérateur « supérieur à » entre deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathcal{B}ooléen$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne ((réel < réel') ou (réel=réel' et imag<imag'))

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne ((réel < polaireACartesien(module')) ou
(réel= polaireACartesien(module') et imag< polaireACartesien(argument')))

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne ((polaireACartesien(module) < réel') ou
(polaireACartesien(module)=réel' et polaireACartesien(argument)<imag'))

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne ((polaireACartesien(module) < polaireACartesien(module'))
ou (polaireACartesien(module)= polaireACartesien(module') et
polaireACartesien(argument)< polaireACartesien(argument')))

Complexité : O(1)

__eq__(self, other):

Opérateur « égale à » entre deux complexes

Signature : $\mathbb{C}omplexe \rightarrow \mathcal{B}ooléen$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} et dict' {reel',imag'}

Alors retourne (dict = dict')

∀ dict {reel,imag} et dict' { module',argument' }

Alors retourne (dict = polaireACartesien(dict'))

∀ dict { module,argument } et dict' { module',argument' }

Alors retourne (dict = dict' mod 2π)

∀ dict { module,argument } et dict' { reel',imag' }

Alors retourne (cartesienAPolaire(dict) = dict' mod 2π)

Complexité : O(1)

polaireACartesien(self):

Transformation d'un complexe polaire en cartésien

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne dict'(cos(argument*module) , sin(argument*module))

Complexité : O(1)

cartesienAPolaire(self):

Transformation d'un complexe cartésien en polaire

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne dict'(__abs__(dict), atan(imag/réel))

Complexité : O(1)

isCartesien(self):

Teste si le complexe est cartésien

Signature : $\mathbb{C}omplexe \rightarrow \mathcal{B}ooléen$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne (∃ réel et ∃ imag)

Complexité : O(1)

isPolaire(self):

Teste si le complexe est polaire

Signature : $\mathbb{C}omplexe \rightarrow \mathcal{B}ooléen$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {module, argument}

Alors retourne (∃module et ∃ argument)

Complexité : O(1)

cartesien(self):

Teste si le complexe est cartésien

Signature : $\mathbb{C}omplexe \rightarrow (Réel, Réel) / String$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne (réel, imag)

∀ dict {module, argument}

Alors retourne Message Erreur

Complexité : O(1)

polaire(self):

Teste si le complexe est polaire

Signature : $\mathbb{C}omplexe \rightarrow (Réel, Réel) / String$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {module, argument}

Alors retourne (module, argument)

∀ dict {reel,imag}

Alors retourne Message Erreur

Complexité : O(1)

oppose(self):

Opposé d'un complexe

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne dict'(-réel, -imag)

∀ dict {module, argument}

Alors retourne (oppose(polaireACartesien(dict)))

Complexité : O(1)

inverse(self):

Inverse d'un complexe

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag} avec réel≠0 et imag≠0

Alors retourne dict'(-réel, -imag)

∀ dict {module, argument} avec module≠0 et argument≠0

Alors retourne dict'($\frac{1}{module}$, -argument)

Complexité : O(1)

conjugue(self):

Conjugué d'un complexe

Signature : $\mathbb{C}omplexe \rightarrow \mathbb{C}omplexe$

Axiome :

∀ dict {reel,imag} ou dict {module, argument} Є Complexe

∀ dict {reel,imag}

Alors retourne dict'(réel, -imag)

∀ dict {module, argument}

Alors retourne conjugue(polaireACartesien(dict))

Complexité : O(1)

Opérations sur les listes

« Sorted » trie par ordre croissant des complexes et « in liste » vérifie que l'élément est dans la liste

Nous avons choisi de ne pas créer de fonction de tri, car les fonctions intégrées à python s'appliquent directement à la liste.

Signature1 : $Liste\ de\ complexes \rightarrow Booléen$

Axiome1 :

Pour i allant de 1 à nombre d'éléments de la liste

Afficher liste

Trier liste

Pour i allant de 1 à nombre d'éléments de la liste triée

Afficher liste triée

Complexité1 : O(n)

Signature2 : $\mathbb{C}omplexe \rightarrow Liste\ de\ complexes$

Axiome2 :

Si complexe présent dans liste

Afficher vrai

Sinon

Afficher faux

Complexité2 : O(1)

Autres axiomes

Commutativité : $\mathbb{C} \cup \mathbb{C}' = \mathbb{C}' \cup \mathbb{C}$ et $\mathbb{C} \cap \mathbb{C}' = \mathbb{C}' \cap \mathbb{C}$ avec $\mathbb{C}$ et $\mathbb{C}'$ complexes

Associativité : $(\mathbb{C}_1 \cup \mathbb{C}_2) \cup \mathbb{C}_3 = \mathbb{C}_1 \cup (\mathbb{C}_2 \cup \mathbb{C}_3)$ et $(\mathbb{C}_1 \cap \mathbb{C}_2) \cap \mathbb{C}_3 = \mathbb{C}_1 \cap (\mathbb{C}_2 \cap \mathbb{C}_3)$ avec $\mathbb{C}_1$, $\mathbb{C}_2$ et $\mathbb{C}_3$ complexes ; bien qu'il n'y ait dans le programme que des opérations entre deux complexes.