ParcourSup.py

Version alpha

févr. 08, 2020

Contenu :

CHAPITRE 1

:fr : Un clone en Python 3 de Parcoursup, écrit à but didactique

Écrit par deux doctorants de l'équipe SCEE de CentraleSupélec, campus de Rennes, Lilian Besson (@Naereen) et Bastien Trotobas (@BastienTr), et d'autres collaborateur-trice-s.

Open Source ? Oui ! Implementation Version de Python

1.1 Présentation

Ce dépôt contient un clone (presque complet) des algorithmes régissant la plateforme Parcoursup, qui gère depuis 2018 les affectations des élèves de classe de Terminale (dans les lycées en France) dans leurs formations dans l'enseignement supérieur.

Les algorithmes et l'implémentation officielle (en Java) ont été distribués en accès libre, et sous licence libre (GPL), en mai 2018. En 2018, ils étaient hébergés sur ce site (framagit.org/parcoursup/algorithmes-de-parcoursup).

- Nous proposons ici une implémentation complète des différents algorithmes de Parcoursup, écrite en Python 3, dans un style très clair, avec des commentaires, et une documentation. Documentation Status
- Nous avons pour objectif de comprendre et d'expliquer ces algorithmes, en utilisant au maximum des notions et des modules Python qui soient abordables et compréhensibles par des élèves de classes préparatoires scientifiques (typiquement des MPSI).
- *Note* : Vous pouvez contribuer si vous le souhaiter ! Une erreur à signaler ?, ou une contribution possible ? :clap : Merci d'avance !

1.2 Plan de bataille

Pour le développement en cours, cf. ce fichier `TODO.md` !

Commits of https://github.com/Naereen/ParcourSup.py/ / Date of last commit of https://github.com/Naereen/ParcourSup.py/ Issues of https://github.com/Naereen/ParcourSup.py/ : Open issues of https://github.com/Naereen/ParcourSup.py/ / Closed issues of https://github.com/Naereen/ParcourSup.py/

1.3 Explications

— Pour l'"instant, nous avons implémenté dans le dossier *parcoursup/* un clone complet du code Java initial, écrit en Python 3.
— Et dans le dossier `notebooks/` nous proposons des implémentations simplifiées des principaux algorithmes, écrites sans dépendances et dans un style très didactique, avec des visualisations interactives afin de permettre à tout le monde d'"expérimenter un peu et de visualiser le comportement des algorithmes. L'"accent est mis sur la compréhension rapide de l'"influence des différents paramètres numériques.
— TODO expliquer l'"algorithme dans les grandes lignes, avec notre propre vocabulaire, ici.

1.4 Organisation de ce dépôt

— Des visualisations sont dans le dossier `notebooks/`. TODO encore à travailler !
— Le code des algorithmes est *dans le dossier parcoursup/*, comme le code Java initial, c'"est découpé en deux modules, `ordreappel` et `propositions`,
— Les (exemples de) données synthétiques générées sont dans le dossier `donnees/`,
— Des tests (plusieurs centaines) sont présents dans le dossier `tests/`, inspirés par ce projet, Build Status
— Une documentation de notre implémentation complète est disponible en ligne, sur la page suivante, construite avec Sphinx à partir des fichiers présents dans le dossier `docs/`,
— Des utilitaires sont dans le dossier `utils/`,

1.5 Démonstration dans un notebook Jupyter

— Des visualisations sont dans le dossier `notebooks/`.

Binder

Google Colab

1.6 Documentation officielle et ressources

— La page officielle de présentation de Parcoursup est ici (en 2018).

Communications journalistiques, entre le 23 mai 2018 et le 15 juillet 2018 :
— Les indicateurs quotidiennement publiés par le ministère sont sur cette page là (en juin 2018).
— Cette carte qui montre jour après jour les résultats donnés par Parcoursup : statistiques.parcoursup.fr.

    Nous voulons proposer notre propre carte de visualisation, c'"est en cours. . .

Des détails sur les algorithmes :
— Ce document texte et cet autre document PDF donnent plein d'"explications.
— Ce texte du Journal Officiel montre l'"autorisation donnée par la CNIL pour la création de la base de données pour Parcoursup, et détaille un peu toutes les informations stockées pour le service. Il est important de garder en tête que ces données ne sont **pas** utilisées par les algorithmes de Parcoursup, qui n'"utilisent qu'"un identifiant unique et anonyme pour identifier chaque candidat-e.

Autres ressources, moins techniques mais plus pédagogiques :
— Le dossier de presse du ministère pour Parcoursup
— Ces articles sur des blogs du Monde : sur ingenuingenieur.blog.lemonde.fr, sur enseigner.blog.lemonde.fr ou sur binaire.blog.lemonde.fr.
— Cet autre article par Clémence Réda est instructif.

1.7 Exemples

1.7.1 Installation

Ces lignes de Bash (à exécuter sur une machine type GNU/Linux ou un Mac avec les outils standards) clone ce dépôt, et installent un `virtualenv` Python et installent les dépendances dans cet environnement virtuel :

```
cd /tmp/
git clone https://GitHub.com/Naereen/ParcourSup.py
cd Parcoursup.py/
make install
```

> Note : Il n"est pas nécessaire d"utiliser un `virtualenv`, mais c"est recommandé. Vous pouvez simplement installer les modules requis avec `sudo pip install -r requirements.txt`.
>
> Note : notre code n"est pas spécifiquement écrit pour une machine utilisant GNU/Linux, et il devrait fonctionner sur n"importe quelle plateforme qui supporte Python 3.6 (Microsoft Windows et Mac OS X notamment). Il est testé sous GNU/Linux (XUbuntu) *et* sous Microsoft Windows 7. N"hésitez pas à signaler un problème, si besoin. :clap : Merci d"avance !

1.7.2 Tests Build Status

Les tests qui reproduisent (presque) parfaitement les données d"exemples peuvent être exécutés avec les deux commandes suivantes :

— Ordres d"appel :

```
$ . env/bin/activate ; python3 ./parcoursup/ordreappel/__init__.py
...
```

— Proposition de vœux :

```
$ . env/bin/activate ; python3 ./parcoursup/propositions/__init__.py
...
```

— Ces deux tests prennent environ 30 secondes chacun.

> Note : Il n"est pas nécessaire d"utiliser un `virtualenv`, mais c"est recommandé. Vous pouvez simplement faire les tests avec `python3 ./parcoursup/ordreappel/__init__.py` et `python3 ./parcoursup/propositions/__init__.py`.

— Des tests supplémentaires ont été récemment ajoutés (voir #3).

1.8 Construire la documentation ? Documentation Status

— Demande d"avoir le module `sphinx` installé. (`sudo pip3 install sphinx` si besoin).
— Puis, dans le dossier principal, il suffit de faire :

```
$ make docs
```

— Sous Windows ou si GNU Make n"est pas disponible, vous pouvez construire la documentation manuellement avec les deux commandes suivantes :

```
$ sphinx-apidoc -f -o ./docs -e -M ./parcoursup/
$ sphinx-build -M html ./docs ./_build
$ ./docs/.fixes_html_in_doc.sh
```

1.9 À propos

1.9.1 Language et versions ?

Python v3.6+. Avec les modules suivants :
— Numpy pour les tableaux,
— La bibliothèque standard pour tout le reste.
— ipython, Jupyter pour les notebooks.
— tqdm sont optionnels.

1.9.2 :scroll : Licence ? GitHub license

Open Source ? Oui ! Implementation Version de Python

ForTheBadge uses-badges ForTheBadge uses-git forthebadge made-with-python ForTheBadge built-with-science

CHAPITRE 2

parcoursup

2.1 parcoursup package

Module principal de https://github.com/Naereen/ParcourSup.py.

- Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
- Adresse : https://github.com/Naereen/ParcourSup.py
- Licence : MIT License (http://lbesson.mit-license.org).

2.1.1 Subpackages

parcoursup.ordreappel package

Calcul de l'ordre d'appel, pour https://github.com/Naereen/ParcourSup.py.

- Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
- Adresse : https://github.com/Naereen/ParcourSup.py
- Licence : MIT License (http://lbesson.mit-license.org).

Submodules

parcoursup.ordreappel.AlgoOrdreAppel module

AlgoOrdreAppel, pour https://github.com/Naereen/ParcourSup.py.

- Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
- Adresse : https://github.com/Naereen/ParcourSup.py
- Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.ordreappel.AlgoOrdreAppel.**AlgoOrdreAppel**(*groupesClassements : List[parcoursup.ordreappel.GroupeClassement.*

Bases : object

Stocke les entrées et sorties de l'algorithme de calcul d'ordre d'appel.

__init__(*groupesClassements : List[parcoursup.ordreappel.GroupeClassement.GroupeClassement]*)
Stocke la liste non-vide de classements.

calculeOrdresAppels() → None
Calcule l'ordre d'appels de chaque groupes de classements.

exporteEntree_XML() → xml.etree.ElementTree.Element
Converti l'entrée en un arbre XML.

exporteEntree_JSON() → Dict
Converti l'entrée en un dictionnaire.

exporteSortie_XML() → xml.etree.ElementTree.Element
Converti les résultats de la sortie en un arbre XML.

exporteSortie_JSON() → Dict
Converti les résultats de la sortie en un dictionnaire.

__dict__ = mappingproxy({'__module__': 'parcoursup.ordreappel.AlgoOrdreAppel', '__doc_

__module__ = 'parcoursup.ordreappel.AlgoOrdreAppel'

__weakref__
list of weak references to the object (if defined)

parcoursup.ordreappel.GroupeClassement module

GroupeClassement, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.ordreappel.GroupeClassement.**GroupeClassement**(*C_GP_COD : int*, *tauxMinBoursiersPourcents : int*, *tauxMinResidentsPourcents : int*)

Bases : object

Classe représentant un groupe d'appel.

__init__(*C_GP_COD : int*, *tauxMinBoursiersPourcents : int*, *tauxMinResidentsPourcents : int*)
Initialize self. See help(type(self)) for accurate signature.

C_GP_COD = None
C_GP_COD

tauxMinBoursiersPourcents = None
tauxMinBoursiersPourcents

tauxMinResidentsPourcents = None
tauxMinResidentsPourcents

voeuxClasses = None
La liste des vœux du groupe de classement

__repr__() → str
Return repr(self).

ajouterVoeu(*voeu : parcoursup.ordreappel.VoeuClasse.VoeuClasse*) → None
Ajouter ce voeu à la liste actuelle.

calculerOrdreAppel(*verbeux : bool = False*) → parcoursup.ordreappel.OrdreAppel.OrdreAppel
Calcule de l'ordre d'appel.

__dict__ = mappingproxy({'__module__': 'parcoursup.ordreappel.GroupeClassement', '__do

__module__ = 'parcoursup.ordreappel.GroupeClassement'

__weakref__
list of weak references to the object (if defined)

parcoursup.ordreappel.OrdreAppel module

OrdreAppel, pour https://github.com/Naereen/ParcourSup.py.
- — Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
- — Adresse : https://github.com/Naereen/ParcourSup.py
- — Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.ordreappel.OrdreAppel.**OrdreAppel**

Bases : list

Classe représentant un ordre d'appel. Juste une liste avec une méthode en plus.

coefficientDivergence() → float

Calcule une mesure de la différence entre le classement original et l'ordre d'appel, c'est le nombre d'inversions ramené au nombre maximal d'inversions.

Le nombre maximal d'inversions est obtenu si le classement est totalement inversé (cas hypothétique), auquel cas il y a autant d'inversions que de paires non-ordonnées de candidat-e c'est-à-dire `n * (n - 1) / 2`.

Note : C'est la distance de Kendall-tau.

__dict__ = mappingproxy({'__module__': 'parcoursup.ordreappel.OrdreAppel', '__doc__':

__module__ = 'parcoursup.ordreappel.OrdreAppel'

__weakref__

list of weak references to the object (if defined)

parcoursup.ordreappel.OrdreAppel.**main**(*taille=10*, *nbEssaisAleatoires=100000*)

Trace un histogramme de la répartition de ce `coefficientDivergence()` pour des listes de taille `taille`.

parcoursup.ordreappel.VoeuClasse module

VoeuClasse, pour https://github.com/Naereen/ParcourSup.py.
- — Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
- — Adresse : https://github.com/Naereen/ParcourSup.py
- — Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.ordreappel.VoeuClasse.**TypeCandidat**

Bases : enum.Enum

Les différents types de candidats

BoursierNonResident = 2

BoursierResident = 1

NonBoursierNonResident = 4

NonBoursierResident = 3

__module__ = 'parcoursup.ordreappel.VoeuClasse'

parcoursup.ordreappel.VoeuClasse.**typeCandidat_vers_str**(*typeCandidat : parcoursup.ordreappel.VoeuClasse.TypeCandidat*) → str

parcoursup.ordreappel.VoeuClasse.**typeCandidat_si_Boursier_etou_Resident**(*estBoursier : bool*, *estResident : bool*) → parcoursup.ordreappel.VoeuClasse.

Donne le type de candidat selon qu'il/elle soit boursier-e et/ou résident-e.

class parcoursup.ordreappel.VoeuClasse.**VoeuClasse**(*G_CN_COD : int*, *rang : int*, *estBoursier : bool*, *estResident : bool*)

Bases : object

Classe représentant un vœu d'un candidat.

__init__(*G_CN_COD : int*, *rang : int*, *estBoursier : bool*, *estResident : bool*)
Initialize self. See help(type(self)) for accurate signature.

G_CN_COD = None
G_CN_COD

rang = None
rang

typeCandidat = None
typeCandidat

__repr__() → str
Return repr(self).

estBoursier() → bool
Pour savoir si le candidat est boursier-e.

estResident() → bool
Pour savoir si le candidat est résident-e.

__lt__(*voeu*) → bool
Comparateur permettant de trier les vœux par ordre du groupe de classement.

__eq__(*voeu*) → bool
Comparateur permettant de trier les vœux par ordre du groupe de classement.

__dict__ = mappingproxy({'__module__': 'parcoursup.ordreappel.VoeuClasse', '__doc__':

__ge__(*other*, *NotImplemented=NotImplemented*)
Return a >= b. Computed by @total_ordering from (not a < b).

__gt__(*other*, *NotImplemented=NotImplemented*)
Return a > b. Computed by @total_ordering from (not a < b) and (a != b).

__hash__ = None

__le__(*other*, *NotImplemented=NotImplemented*)
Return a <= b. Computed by @total_ordering from (a < b) or (a == b).

__module__ = 'parcoursup.ordreappel.VoeuClasse'

__weakref__
list of weak references to the object (if defined)

parcoursup.ordreappel.exemples module

ordreappel.exemples, pour https://github.com/Naereen/ParcourSup.py.

— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.ordreappel.exemples.**Exemple**

Bases : object

Un exemple.

__init__()

Initialize self. See help(type(self)) for accurate signature.

nom = None

Nom pour le fichier .xml ou .json de test.

initialise() → None

Construit l'attribut groupes, il ne doit pas être vide.

exporte(*contenu*, *entree=True*, *xml=False*, *debug=False*) → bool

Exporte l'entrée ou la sortie, dans un fichier XML ou JSON.

execute() → None

Calcule les ordre d'appels et sauvegarde les fichiers.

__dict__ = mappingproxy({'__module__': 'parcoursup.ordreappel.exemples', '__doc__': ' t

__module__ = 'parcoursup.ordreappel.exemples'

__weakref__

list of weak references to the object (if defined)

class parcoursup.ordreappel.exemples.**exempleA1**

Bases : *parcoursup.ordreappel.exemples.Exemple*

Exemple A1 avec une contrainte de 20% de boursiers-ère-s et de 0% de non résidents-e-s.

initialise() → None

Construit l'attribut groupes, il ne doit pas être vide.

__module__ = 'parcoursup.ordreappel.exemples'

class parcoursup.ordreappel.exemples.**exempleA2**

Bases : *parcoursup.ordreappel.exemples.Exemple*

Exemple A2 avec une contrainte de 2% de boursiers-ère-s et de 0% de non résidents-e-s.

initialise() → None

Construit l'attribut groupes, il ne doit pas être vide.

__module__ = 'parcoursup.ordreappel.exemples'

class parcoursup.ordreappel.exemples.**exempleA3**

Bases : *parcoursup.ordreappel.exemples.Exemple*

Exemple A3 avec une contrainte de 10% de boursiers-ère-s et de 0% de non résidents-e-s.

initialise() → None

Construit l'attribut groupes, il ne doit pas être vide.

__module__ = 'parcoursup.ordreappel.exemples'

class parcoursup.ordreappel.exemples.**exempleA4**

Bases : *parcoursup.ordreappel.exemples.Exemple*

Exemple A4 avec une contrainte de 10% de boursiers-ère-s et de 0% de non résidents-e-s.

initialise() → None

Construit l'attribut groupes, il ne doit pas être vide.

__module__ = 'parcoursup.ordreappel.exemples'

class parcoursup.ordreappel.exemples.**exempleA5**

Bases : *parcoursup.ordreappel.exemples.Exemple*

Exemple A5 avec une contrainte de 10% de boursiers-ère-s et de 95% de non résidents-e-s.

initialise() → None

Construit l'attribut groupes, il ne doit pas être vide.

__module__ = 'parcoursup.ordreappel.exemples'

class parcoursup.ordreappel.exemples.**exempleA6**
Bases : *parcoursup.ordreappel.exemples.Exemple*
Exemple A6 avec une contrainte de 10% de boursiers-ère-s et de 95% de non résidents-e-s.
initialise() → None
Construit l'attribut groupes, il ne doit pas être vide.
__module__ = 'parcoursup.ordreappel.exemples'

parcoursup.ordreappel.exemples.**tous_les_exemples = [<class 'parcoursup.ordreappel.exemples**
Liste de tous les exemples.

parcoursup.propositions package

Calcul des propositions à envoyer, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

Submodules

parcoursup.propositions.AlgoPropositions module

AlgoPropositions, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

parcoursup.propositions.AlgoPropositions.**log**(**args*, ***kwargs*)
Affiche avec une heure.

parcoursup.propositions.AlgoPropositions.**str_de_bool**(*b : bool*) → str
`True` ou `False` en minuscule.

class parcoursup.propositions.AlgoPropositions.**AlgoPropositions**(*groupesAffectations : List[parcoursup.propositions.GroupeAf = None*, *internats : List[parcoursup.propositions.GroupeIn = None*)
Bases : `object`
Stocke les entrées et sorties de l'algorithme de calcul d'ordre d'appel.
__init__(*groupesAffectations : List[parcoursup.propositions.GroupeAffectation.GroupeAffectation] = None*, *internats : List[parcoursup.propositions.GroupeInternat.GroupeInternat] = None*)
Stocke la liste non-vide de classements.
groupesAffectations = None
La liste des groupes d'affectation, contenant leurs voeux respectifs.
internats = None
La liste des internats, contenant leurs vœux respectifs.
internats_sortie = None
Liste des internats, permettant de récupérer les positions max d'admission
propositions = None
Liste des propositions à faire.
enAttentes = None
Liste des vœux restant en attente.

verifierIntegrite() → bool
Vérifie l'intégrité des données d'entrée, et lève une exception si nécessaire.

Propriétés :
— a) tous les vœux sont en attente,
— b) pas deux vœux distincts avec la même id,
— c) pas deux candidats distincts avec le même classement, formation et internat,
— d) pas le même candidat avec deux classements distincts, formation et internat,
— e) classements positifs,
— f) chaque voeu avec internat se retrouve dans l'internat correspondant.

> **Avertissement :** Une exception `AssertionError` est lancée avec un message commençant par `a)` ou `...` ou `f)`.

calculePropositions() → None
Calcule les propositions à envoyer.

exporteEntree_XML() → xml.etree.ElementTree.Element
Converti l'entrée en un arbre XML.

exporteEntree_JSON() → Dict
Converti l'entrée en un dictionnaire.

exporteSortie_XML() → xml.etree.ElementTree.Element
Converti les résultats de la sortie en un arbre XML.

exporteSortie_JSON() → Dict
Converti les résultats de la sortie en un dictionnaire.

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.AlgoPropositions', '__

__module__ = 'parcoursup.propositions.AlgoPropositions'

__weakref__
list of weak references to the object (if defined)

parcoursup.propositions.Candidat module

Candidat, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.Candidat.**Candidat**
Bases : object
Objet minuscule pour numéroter un candidat.

last_G_CN_COD = 1

__init__()
Initialize self. See help(type(self)) for accurate signature.

__repr__() → str
Return repr(self).

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.Candidat', '__doc__':

__module__ = 'parcoursup.propositions.Candidat'

__weakref__
list of weak references to the object (if defined)

parcoursup.propositions.Etablissement module

Etablissement, pour https://github.com/Naereen/ParcourSup.py.

— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

parcoursup.propositions.Etablissement.**randbool**() → bool
Pile ou face, True avec probabilité 1/2 et False avec probabilité 1/2.

parcoursup.propositions.Etablissement.**avecproba**(*p*) → bool
Pile ou face biaisé, True avec probabilité p et False avec probabilité 1-p.

class parcoursup.propositions.Etablissement.**GroupeClassement**(*nbEtudiants : int = 100*)
Bases : object
Classe pour représentation un groupe de classement.

last_C_G_COD = 1

__init__(*nbEtudiants : int = 100*)
Initialize self. See help(type(self)) for accurate signature.

plusHautRangAffecte = None
le rang le plus haut dans l'ordre d'appel d'un candidat recruté

__repr__() → str
Return repr(self).

ajouterCandidat(*c : parcoursup.propositions.Candidat.Candidat*, *maxEtapes : int = 1000*) → int
Ajoute un candidat et renvoie son rang.

> **Avertissement :** Ici, on ajoute un compteur maxEtapes pour borner le nombre de tentative aléatoire.

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.Etablissement', '__doc

__module__ = 'parcoursup.propositions.Etablissement'

__weakref__
list of weak references to the object (if defined)

class parcoursup.propositions.Etablissement.**FormationAffectation**
Bases : object
Classe pour représentation une formation en affectation.

last_G_TA_COD = 1

capaciteMaxFormationNormale = 100

capaciteMaxFormationCC = 200

__init__()
Initialize self. See help(type(self)) for accurate signature.

__repr__() → str
Return repr(self).

ajouterGroupe(*c : parcoursup.propositions.Etablissement.GroupeClassement*, *G_TI_COD : int*, *G_TA_COD : int*, *isConcoursCommun : bool*) → None
Ajoute un groupe de classement.

ajouterVoeu(*candidat : parcoursup.propositions.Candidat.Candidat*, *avecInternat : bool*) → None
Ajoute un vœu (un candidat et une demande d'internat).

capacite() → int
Capacité d'une formation = somme des capacités de ses groupe.

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.Etablissement', '__doc

__module__ = 'parcoursup.propositions.Etablissement'

__weakref__
list of weak references to the object (if defined)

class parcoursup.propositions.Etablissement.**Etablissement**(*nbEtudiants : int = 100*)

Bases : object

Classe comprenant les caractéristiques d'un établissement (aléatoire).

last_G_TI_COD = 1

maxNbVoeuxParConcoursCommun = 80

proportionConcoursCommuns = 0.1

nbFormationsParConcours = 100

proportionInternatsCommuns = 0.5

proportionInternats = 0.5

nbFormationsParEtablissement = 5

capaciteMaxInternat = 30

maxNbGroupesParFormation = 5

__init__(*nbEtudiants : int = 100*)

Initialize self. See help(type(self)) for accurate signature.

__repr__() → str

Return repr(self).

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.Etablissement', '__doc

__module__ = 'parcoursup.propositions.Etablissement'

__weakref__

list of weak references to the object (if defined)

capacite() → int

Capacité d'un établissement = somme des capacités de ses formations.

ajouterVoeux(*candidat : parcoursup.propositions.Candidat.Candidat*) → int

Ajoute un vœu (id du candidat).

parcoursup.propositions.Etablissement.**random**() → x in the interval [0, 1).

parcoursup.propositions.GroupeAffectation module

GroupeAffectation, pour https://github.com/Naereen/ParcourSup.py.

— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.

— Adresse : https://github.com/Naereen/ParcourSup.py

— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.GroupeAffectation.**GroupeAffectation**(*capacite : int, uid : parcoursup.propositions.GroupeAffectationU rangLimite : int*)

Bases : object

Classe comprenant les caractéristiques identifiant de manière unique un groupe d'affectation dans la base de données.

__init__(*capacite : int, uid : parcoursup.propositions.GroupeAffectationUID.GroupeAffectationUID, rangLimite : int*)

Initialize self. See help(type(self)) for accurate signature.

id = None

Le id d'affectation identifiant de manière unique le groupe dans la base.

capacite = None

La capacité de la formation.

rangLimite = None
Le rang limite des candidats (dans l'ordre d'appel) : tous les candidats de rang inférieur reçoivent une proposition.

voeux = None
la liste initiale des voeux du groupe, triée dans l'ordre d'appel du candidat. Remarque : c'est un ordre partiel car il peut y avoir deux voeux du même candidat, un avec internat et l'autre sans.

voeuxSontTries = None
`True` si et seulement si les vœux ont été triés.

candidatsAffectes = None
Ensemble des candidats affectés.

__repr__() → str
Return repr(self).

ajouterVoeu(*voe : parcoursup.propositions.VoeuEnAttente.VoeuEnAttente*) → None
Ajoute un voeu dans le groupe.

ajouterCandidatAffecte(*G_CN_COD : int*) → None
Ajoute un candidat affecté.

estAffecte(*G_CN_COD : int*) → bool
Teste si un candidat est affecté.

mettreAJourPropositions() → None
Met a jour le statut `aProposer`, pour chaque voeu du groupe.

nbPlacesVacantes() → int
Le nombre de places vacantes au lancement du calcul.

> **Avertissement :** Peut être négatif en cas de modification à la baisse des paramètres de surréservation.

estInitialementEnSurcapacite() → bool
La formation était elle initialement en surcapacité ?

voeuxTries() → List[parcoursup.propositions.VoeuEnAttente.VoeuEnAttente]
Trie les voeux dans l'ordre d'appel.

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.GroupeAffectation', '_

__module__ = 'parcoursup.propositions.GroupeAffectation'

__weakref__
list of weak references to the object (if defined)

parcoursup.propositions.GroupeAffectationUID module

GroupeAffectationUID, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.GroupeAffectationUID.**GroupeAffectationUID**(*C_GP_COD : int*, *G_TI_COD : int*, *G_TA_COD : int*)

Bases : `object`

Classe comprenant les caractéristiques identifiant de manière unique une affectation dans la base de données.

__init__(*C_GP_COD : int*, *G_TI_COD : int*, *G_TA_COD : int*)
Initialize self. See help(type(self)) for accurate signature.

C_GP_COD = None
L'identifiant unique du groupe de classement pédagogique dans la base de données.

G_TI_COD = None
L'identifiant unique de la formation d'inscription dans la base de données.

G_TA_COD = None
L'identifiant unique de la formation d'affectation dans la base de données.

__repr__() → str
Return repr(self).

__eq__(*groupeaffectationuid*) → bool
Return self==value.

__hash__() → int
FIXME il n'y a aucune chance qu'on obtienne les mêmes hashCode qu'en Java...
— Je ne crois pas que ça posera problème, mais peut-être...

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.GroupeAffectationUID',
__module__ = 'parcoursup.propositions.GroupeAffectationUID'
__weakref__
list of weak references to the object (if defined)

parcoursup.propositions.GroupeInternat module

GroupeInternat, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.GroupeInternat.**GroupeInternat**(*uid : parcoursup.propositions.GroupeInternatUID.GroupeI capacite : int*, *pourcentageOuverture : int*)

Bases : object

Classe comprenant les caractéristiques identifiant de manière unique un internat dans la base de données.

nbJoursCampagne = 1
Le nombre de jours depuis l'ouverture de la campagne, 1 le premier jour.

__init__(*uid : parcoursup.propositions.GroupeInternatUID.GroupeInternatUID*, *capacite : int*, *pourcentageOuverture : int*)
Initialize self. See help(type(self)) for accurate signature.

id = None
L'identifiant unique de l'internat dans la base de données

capacite = None
Le nombre total de places

pourcentageOuverture = None
Le pourcentage d'ouverture fixé par le chef d'établissement

contingentAdmission = None
Le nombre de demandes d'internat considérées Bmax dans le document de spécification

positionAdmission = None
La position d'admission dans cet internat, calculée par l'algorithme

positionMaximaleAdmission = None
La position maximale d'admission dans cet internat, calculée par l'algorithme

groupesConcernes = None
La liste des groupes de classement concernés par cet internat

voeux = None
La liste des voeux du groupe. Après le calcul de la position initiale d'admission cette liste est triée par ordre de classement internat

estInitialise = None
`True` si et seulement si la position maximale d'admission a été calculée, ce qui implique que la liste des voeux est triée par ordre de classement internat.

candidatsAffectes = None
Ensemble des candidats affectés.

candidatsEnAttente = None
Ensemble des candidats en attente.

__repr__() → str
Return repr(self).

nbPlacesVacantes() → int
Le nombre de places vacantes dans cet internat.

ajouterVoeu(*voeu : parcoursup.propositions.VoeuEnAttente.VoeuEnAttente*, *groupe : parcoursup.propositions.GroupeAffectation.GroupeAffectation*) → None
Ajouter ce vœu à ce groupe d'affectation.

ajouterCandidatAffecte(*G_CN_CODE : int*) → None
Ajoute un candidat affecté.
— Supprime le candidat de la liste des candidats en attente , si il y a lieu.

estAffecte(*G_CN_CODE : int*) → bool
Vérifie si le candidat est affecté.

calculeAssietteAdmission(*M : int*, *L : int*, *t : int*, *p : int*) → int
Calcule l'assiette d'admission Bmax comme décrit dans l'algorithme.

initialiserPositionAdmission() → None
Initialise la position d'admission à son maximum Bmax dans le document de référence.

mettreAJourPositionAdmission() → bool
Met à jour la position d'admission si nécessaire.
— Renvoie `True` si la position d'admission a été effectivement mise à jour.

__annotations__ = {'nbJoursCampagne': <class 'int'>}

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.GroupeInternat', '__an

__module__ = 'parcoursup.propositions.GroupeInternat'

__weakref__
list of weak references to the object (if defined)

parcoursup.propositions.GroupeInternatUID module

GroupeInternatUID, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.GroupeInternatUID.**GroupeInternatUID**(*C_GI_COD : int*, *G_TA_COD : int*)
Bases : `object`
Classe comprenant les caractéristiques identifiant de manière unique un internat dans la base de données.

__init__(*C_GI_COD : int*, *G_TA_COD : int*)
Initialize self. See help(type(self)) for accurate signature.

C_GI_COD = None
L'identifiant unique de l'internat dans la base de données

G_TA_COD = None
L'identifiant unique de la formation d'affectation dans la base de données. Positionné à 0 pour un internat commun à plusieurs formations.

__repr__() → str
Return repr(self).

__eq__(*groupeinternatuid*) → bool
Return self==value.

__hash__() → int
FIXME il n'y a aucune chance qu'on obtienne les mêmes hashCode qu'en Java...
— Je ne crois pas que ça posera problème, mais peut-être...

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.GroupeInternatUID', '_

__module__ = 'parcoursup.propositions.GroupeInternatUID'

__weakref__
list of weak references to the object (if defined)

parcoursup.propositions.VerificationsResultats module

VerificationsResultats, pour https://github.com/Naereen/ParcourSup.py.
— Permet de vérifier un certain nombre de propriétés statiques des sorties de l'algorithme.
— Sans garantir la correction du code, cela garantit que les résultats produits satisfont les principales propriétés énoncées dans le document.

Avertissement : FIXME Des tests complémentaires sont effectués en base par des scripts PL/SQL.

— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.
— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

parcoursup.propositions.VerificationsResultats.**log**(**args*, ***kwargs*)
Affiche avec une heure.

parcoursup.propositions.VerificationsResultats.**verifierRespectOrdreAppelVoeuxSansInternat**(*f*
l
c
s

P1 : respect ordre appel pour les voeux sans internat.
Si un candidat C1 précède un candidat C2 dans l'ordre d'appel d'une formation F et si C1 a un voeu en attente pour F sans demande d'internat alors C2 n'a pas de proposition pour F.

parcoursup.propositions.VerificationsResultats.**verifierVoeuxAvecInternat**(*formation : parcoursup.propositions.GroupeA*

P2 respect ordre appel et classement internat pour les voeux avec internat.
Si un candidat C1 précède un candidat C2 à la fois dans l'ordre d'appel d'une formation F et dans un classement d'internat I et si C1 a un voeu en attente pour F avec internat I alors C2 n'a pas de proposition pour F avec internat I.

parcoursup.propositions.VerificationsResultats.**verifierRespectClassementInternat**(*formation : parcoursup.propositio*

P3 respect classement internat pour les candidats avec une proposition sans internat.

Si un candidat C1 a un voeu en attente pour une formation F avec demande d'internat I et une proposition acceptée ou en attente de réponse de sa part pour la formation F sans demande d'internat, et si C2 est un candidat moins bien classé que C1 à l'internat I et si une des nouvelles propositions du jour offre l'internat I à C2 alors que C2 n'avait pas de propositions pour I auparavant alors une des nouvelles propositions du jour offre la formation F et l'internat I à C1.

parcoursup.propositions.VerificationsResultats.**verifierSurcapaciteEtRemplissage**(*formation : par-cour-sup.proposition*

P4 remplissage maximal des formations dans le respect des positions d'admission à l'internat.

Le nombre de propositions doit être inférieur au nombre de places vacantes.

Si le nombre de nouvelles propositions dans une formation est strictement inférieur au nombre de places vacantes dans cette formation, alors tous les voeux en attente pour cette formation sont des voeux avec internat, effectués par des candidats dont le rang de classement dans l'internat correspondant est strictement supérieur à la position d'admission dans cet internat.

parcoursup.propositions.VerificationsResultats.**verifierSurcapaciteEtRemplissage_avec_rangDe**

P5 remplissage maximal des internats dans le respect des ordres d'appel.

Le nombre de propositions doit être inférieur au nombre de places vacantes.

Si le nombre de nouvelles propositions dans un internat I est strictement inférieur au nombre de places vacantes dans I, alors tous les voeux en attente pour une formation F et demande d'internat I sont soit effectués par des candidats dont le classement à l'internat I est strictement supérieur à la position d'admission dans I ou bien situés plus bas dans l'ordre d'appel de F que tous les candidats ayant reçu une estAProposer de F ce jour là.

parcoursup.propositions.VerificationsResultats.**verifierMaximalitePositionsAdmission**(*internat : par-cour-sup.propo*

P6 : maximalité des positions d'admission.

Pour tout internat, la position d'admission est inférieure ou égale à la position maximale d'admission. Dans le cas où elle est strictement inférieure, augmenter d'une unité la position d'admission entrainerait une surcapacité d'un des internats.

Avertissement : FIXME le code de référence ne l'a pas fait (« Non-implémenté. »), mais nous pouvons essayer ?

parcoursup.propositions.VoeuEnAttente module

VoeuEnAttente, pour https://github.com/Naereen/ParcourSup.py.
— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.

— Adresse : https://github.com/Naereen/ParcourSup.py
— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.VoeuEnAttente.**VoeuEnAttente**(*uid : parcoursup.propositions.VoeuUID.VoeuUID*, *groupe*, *internat=None*, *rangInternat : int = 0*, *ordreAppel : int = 0*)

Bases : object

Classe comprenant les caractéristiques identifiant de manière unique un vœu dans la base de données.

voeuxCrees = {}

verificationUnicite = False

__init__(*uid : parcoursup.propositions.VoeuUID.VoeuUID*, *groupe*, *internat=None*, *rangInternat : int = 0*, *ordreAppel : int = 0*)
Initialize self. See help(type(self)) for accurate signature.

id = None
Caractéristiques identifiant de manière unique le voeu dans la base de données

groupe = None
Groupe d'affectation du voeu

ordreAppel = None
Rang du voeu dans l'ordre d'appel

internat = None
le groupe de classement internat, qui donne accès à la position d'admission

rangInternat = None
Le rang du candidat au classement internat

rangListeAttente = None
Rang sur liste attente

rangListeAttenteInternat = None
Rang sur liste attente internat

G_CN_COD

__repr__() → str
Return repr(self).

static ajouterVoeu(*G_CN_COD : int*, *groupe*, *ordreAppel : int*, *internat=None*, *rangInternat : int = 0*, *avecInternat : bool = False*)

avecInternat() → bool
Y a-t-il une demande d'internat sur ce voeu ?

avecClassementInternat() → bool
Y a-t-il une demande d'internat avec classement sur ce voeux ?

internatDejaObtenu() → bool
Le-la candidat-a a-t-il/elle déjà une offre dans cet internat (pour une autre formation)

formationDejaObtenue() → bool
Le-la candidat-a a-t-il/elle déjà une offre dans cette formation (sans internat)

internatID() → Union[None, int]
Identifiant de l'internat obtenu, None sinon.

estAProposer() → bool
Méthode triviale, pour accéder au résultat du calcul : fait-on une proposition sur ce vœu ?

proposer() → None
Méthode triviale, pour accéder au résultat du calcul : fait-on une proposition sur ce vœu ?

conserverEnAttente() → None
Méthode triviale, pour accéder au résultat du calcul : fait-on une proposition sur ce vœu ?

estDesactiveParPositionAdmissionInternat() → bool
Vérifie si le voeu est désactivé du fait d'une demande d'internat

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.VoeuEnAttente', '__doc

__module__ = 'parcoursup.propositions.VoeuEnAttente'

__weakref__

list of weak references to the object (if defined)

parcoursup.propositions.VoeuUID module

VoeuUID, pour https://github.com/Naereen/ParcourSup.py.

— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.

— Adresse : https://github.com/Naereen/ParcourSup.py

— Licence : MIT License (http://lbesson.mit-license.org).

class parcoursup.propositions.VoeuUID.**VoeuUID**(*G_CN_COD : int*, *G_TA_COD : int*, *avecInternat : bool*)

Bases : object

Classe comprenant les caractéristiques identifiant de manière unique un vœu dans la base de données.

voeuxCrees = {}

verificationUnicite = False

__init__(*G_CN_COD : int*, *G_TA_COD : int*, *avecInternat : bool*)

Initialize self. See help(type(self)) for accurate signature.

G_CN_COD = None

L'identifiant unique du candidat dans la base de données

G_TA_COD = None

L'identifiant unique de la formation d'affectation dans la base de données. Positionné à -1 pour les internats commun à plusieurs formations

I_RH_COD = None

tauxMinBoursiersPourcents

__repr__() → str

Return repr(self).

__eq__(*voeuuid*) → bool

Return self==value.

__hash__() → int

FIXME il n'y a aucune chance qu'on obtienne les mêmes hashCode qu'en Java...

— Je ne crois pas que ça posera problème, mais peut-être...

debuterVerificationUnicite() → None

Vérifie qu'un identifiant est créé au plus une fois.

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.VoeuUID', '__doc__': '

__module__ = 'parcoursup.propositions.VoeuUID'

__weakref__

list of weak references to the object (if defined)

parcoursup.propositions.exemples module

propositions.exemples, pour https://github.com/Naereen/ParcourSup.py.

— Auteurs : Lilian Besson, Bastien Trotobas et al, (C) 2018.

— Adresse : https://github.com/Naereen/ParcourSup.py

— Licence : MIT License (http://lbesson.mit-license.org).

parcoursup.propositions.exemples.**log**(**args*, ***kwargs*)

Affiche avec une heure.

parcoursup.propositions.exemples.**randbool**() → bool

Pile ou face, True avec probabilité 1/2 et False avec probabilité 1/2.

class parcoursup.propositions.exemples.**Exemple**

Bases : object

Un exemple.

__init__()

Initialize self. See help(type(self)) for accurate signature.

n = None

Nombre total de candidats

nom = None

Nom pour le fichier .xml ou .json de test.

donneesEntree()

Construit l'attribut groupes et internat, il ne doit pas être vide.

exporte(*contenu*, *entree=True*, *xml=False*, *debug=False*)

Exporte l'entrée ou la sortie, dans un fichier XML ou JSON.

execute(*sauvegarde=True*)

Calcule les ordre d'appels et sauvegarde les fichiers.

__dict__ = mappingproxy({'__module__': 'parcoursup.propositions.exemples', '__doc__':

__module__ = 'parcoursup.propositions.exemples'

__weakref__

list of weak references to the object (if defined)

class parcoursup.propositions.exemples.**exempleB7base**

Bases : *parcoursup.propositions.exemples.Exemple*

Exemple B7 de base.

donneesEntree()

Construit l'attribut groupes et internat, il ne doit pas être vide.

__module__ = 'parcoursup.propositions.exemples'

class parcoursup.propositions.exemples.**exempleB7Jour1**

Bases : *parcoursup.propositions.exemples.exempleB7base*

Exemple B7 du jour 1.

donneesEntree() → None

Construit l'attribut groupes et internat, il ne doit pas être vide.

__module__ = 'parcoursup.propositions.exemples'

class parcoursup.propositions.exemples.**exempleB7Jour2**

Bases : *parcoursup.propositions.exemples.exempleB7base*

Exemple B7 du jour 2.

donneesEntree() → None

Construit l'attribut groupes et internat, il ne doit pas être vide.

__module__ = 'parcoursup.propositions.exemples'

class parcoursup.propositions.exemples.**exempleB7Jour3**

Bases : *parcoursup.propositions.exemples.exempleB7base*

Exemple B7 du jour 3.

donneesEntree() → None

Construit l'attribut groupes et internat, il ne doit pas être vide.

__module__ = 'parcoursup.propositions.exemples'

class parcoursup.propositions.exemples.**exempleAleatoire**(*nbEtudiants : int = 100*)

Bases : *parcoursup.propositions.exemples.exempleB7base*

Exemple aléatoire très complet et compliqué.

maxNbVoeuxParCandidat = 40

__init__(*nbEtudiants : int = 100*)

Initialize self. See help(type(self)) for accurate signature.

donneesEntree() → None
Construit l'attribut groupes et internat, il ne doit pas être vide.

__module__ = 'parcoursup.propositions.exemples'

parcoursup.propositions.exemples.**tous_les_exemples = [<class 'parcoursup.propositions.exemp**
Liste de tous les exemples.

parcoursup.propositions.exemples.**random**() → x in the interval [0, 1).

CHAPITRE 3

Index et tables

— genindex
— modindex
— search

Index des modules Python

p

Index

Symboles

A

B

C

F

G

I

L

M

N

O

P

R

S

T

V