ACADÉMIE DE NICE
Liberté
Égalité
Fraternité

Groupe Numérique Disciplinaire (GND)

en Mathématiques

Février 2021

BENRIDA Mustapha
Professeur de mathématiques
Lycée Jean AICARD – HYERES – Var

PONSONNET Luc
Professeur de mathématiques
Lycée BONAPARTE – TOULON – Var

Objectifs pédagogiques :

De la classe de seconde à la classe de terminale, il est très difficile de mener des TP de programmation pour traiter les algorithmes exigibles dans les programmes scolaires.

Bien souvent les élèves ne possèdent pas les bases de programmation Python. Cet article a pour objectif de proposer aux collègues une méthode afin de mettre en place des séquences de programmation hors de la classe avec l'outil Jupyter Notebook.

Outils utilisés :

Conception des notebooks à l'aide de Jupyter Notebook. Récupération éventuelle des travaux des élèves grâce aux espaces de dépôt Moodle ou Pronote.

Utilisation (seulement sur l'ordinateur de l'enseignant) :

- D'Anaconda afin d'installer un environnement de programmation Python et Jupyter Notebook.
- De la plateforme GitHub dans le but de stocker et partager les Notebooks.
- De Binder qui propose de déployer un environnement Jupyter en ligne sans aucune installation de la part des élèves.
- Pour ceux qui le souhaitent, d'un éditeur Latex (Texmaker ou MikTex) afin de créer des pdf à partir des Jupyter notebooks.

NB : L'élève n'a besoin que d'un accès internet, et il commence le TP de programmation Python par un simple clic sur un lien hypertexte Binder.

Voie : générale ou technologique

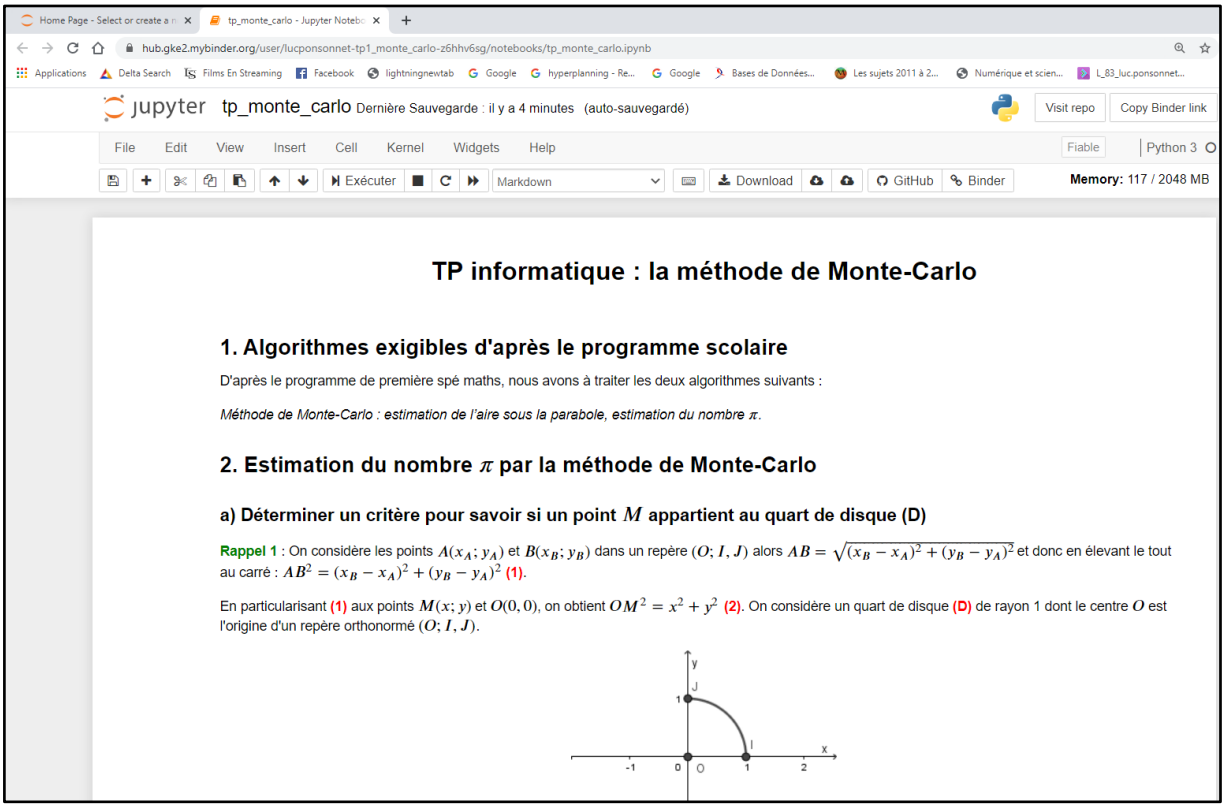
Niveau(x) de classe : tous niveaux du lycée

Thématique(s) du programme : algorithmique et programmation Python

SOMMAIRE

1) Des exemples de TP Jupyter Notebooks et des mises en œuvre possibles

Une fois le TP Jupyter Notebook créé, il est très facile de le mettre à disposition des élèves par l'intermédiaire d'un lien Binder (exemple : https://mybinder.org/v2/gh/lucponsonnet/TP1_Monte_Carlo.git/HEAD). Après avoir cliqué sur ce lien, l'élève lancera le TP Jupyter Notebook en cliquant sur le fichier d'extension *.ipynb* (ici *tp_monte_carlo.ipynb*) qui s'ouvrira à l'aide du navigateur par défaut (Microsoft Edge, Chrome ou Mozilla).



Il est préférable de traiter un premier TP Jupyter Notebook avec vos élèves. Ils apprendront ainsi à :

- lancer le TP en cliquant sur le fichier d'extension *.ipynb*.
- enregistrer de temps en temps leur travail à l'aide de l'icône disquette  .
- Actualiser la page si la connexion avec le serveur s'est interrompue (touche F5 ou l'icône du navigateur).

- A bien comprendre la nature des différentes cellules qui composent le TP (*cellule Markdown*, *cellule Code* et *Cellule Texte brut*) pour :
    - Exécuter une *cellule Code* par l'icône Exécuter 

ou le réaccourci clavier *CTRL + Entrée*.
    - Apprendre à écrire leurs réponses dans les *cellules Texte brut*.

- A enregistrer leur travail au format *.ipynb, .html* ou *.pdf* dans le but de le rendre sur un espace de dépôt Moodle ou Pronote (voir ***annexe 1 : consignes élèves concernant le premier TP Jupyter Notebook***).

Une fois cet « aspect technique dépassé », les élèves pourront travailler de chez eux les différents TP Jupyter Notebook, et apprendre à consolider seuls les bases de la programmation Python. Une correction pourra leur être proposée soit au format *.ipynb* soit au format *.pdf* (voir paragraphe ***4) Comment créer des pdf à partir de Jupyter Notebooks***).

Voici 7 TP Jupyter Notebook dont le but est de faire retravailler l'ensemble des bases de la programmation Python :

1. TP sur les booléens : https://mybinder.org/v2/gh/lucponsonnet/tp_booleen/HEAD
2. TP sur les instructions conditionnelles : https://mybinder.org/v2/gh/lucponsonnet/tp_if_else/HEAD
3. TP sur la boucle for : https://mybinder.org/v2/gh/lucponsonnet/tp_boucle_for/HEAD
4. TP sur la boucle while : https://mybinder.org/v2/gh/lucponsonnet/tp_boucle_while/HEAD
5. TP sur les fonctions : https://mybinder.org/v2/gh/lucponsonnet/tp_fonctions/HEAD
6. TP sur les listes : https://mybinder.org/v2/gh/lucponsonnet/tp_listes/HEAD
7. TP sur les chaînes de caractères : https://mybinder.org/v2/gh/lucponsonnet/tp_chaines/HEAD

Ces TP conviendront tout particulièrement aux élèves des classes de première et terminale. Une fois les contenus compris, ils pourront refaire ces TP plusieurs fois dans l'année pour assimiler véritablement les structures algorithmiques de base voire même développer certains automatismes liés à la programmation Python.

Certains TP pourront être commencés en classe (dans une salle informatique ou une salle normale avec des tablettes), puis les élèves devront les terminer chez eux avec remise éventuelle à l'enseignant du TP fini au format *.ipynb*, *.html* ou *.pdf*.

En voici deux exemples :

1. TP tracer une courbe avec Matplotlib : https://mybinder.org/v2/gh/gitbenrida/Tracer_Courbe_de_Fonction_avec_Python.git/HEAD
2. TP méthode de Monte-Carlo : https://mybinder.org/v2/gh/lucponsonnet/TP1_Monte_Carlo.git/HEAD

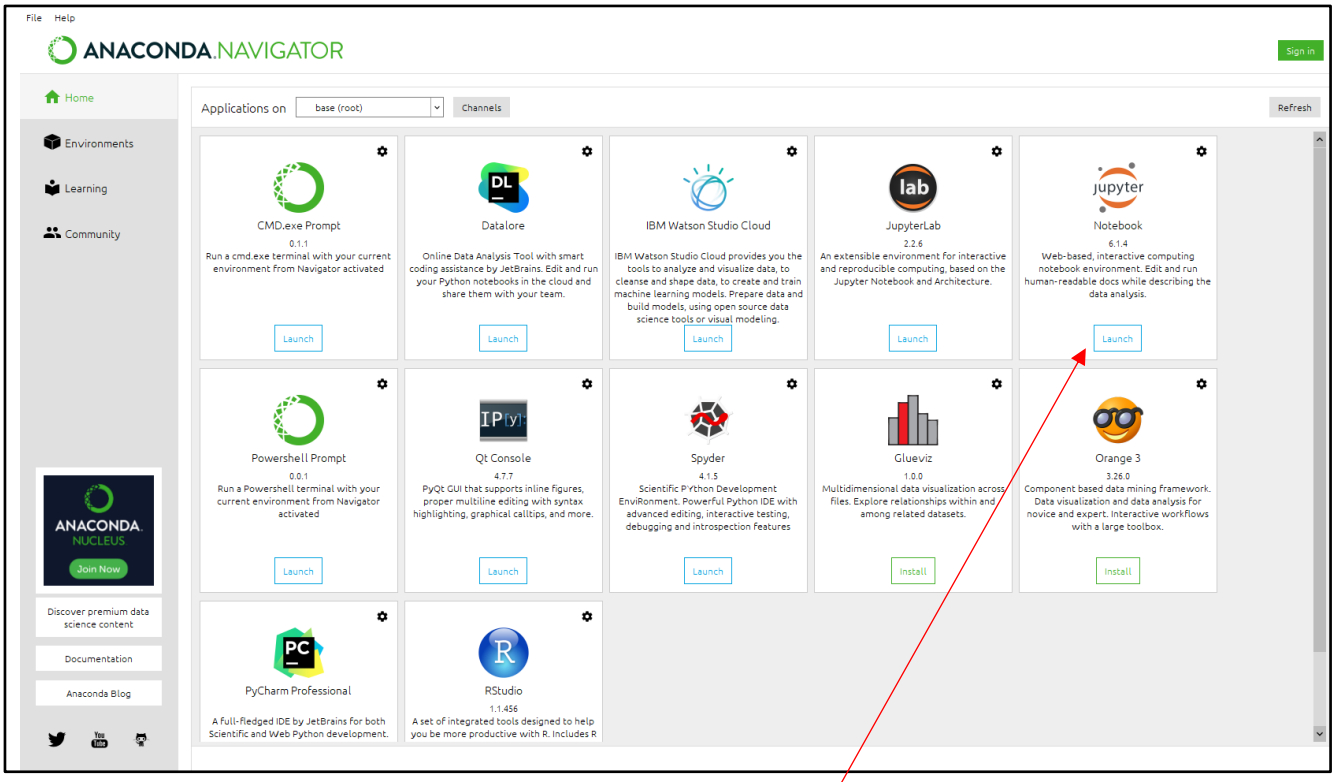
Quelques remarques :

- L'utilisation du TP Jupyter Notebook est très efficace auprès des élèves et d'une grande facilité d'utilisation.
- Les élèves n'ont pas à retaper le script. Cela permet de gagner du temps et d'éviter les trop nombreuses erreurs de syntaxes lors de l'écriture des lignes du programme. Mais il est parfois utile que les élèves réécrivent le code, cela permet de faciliter la mémorisation. C'est pourquoi, l'énoncé devra contenir des scripts sous forme d'images pour éviter les copier-coller.
- Les explications sont « dynamiques » et les pages du TP interactives. On peut illustrer très facilement un exemple ou nos propos à l'aide d'une *cellule Code* qu'ils auront juste à exécuter. Les images et les liens hypertextes sont facilement intégrables dans le TP.
- Tout est présent sur un même support, les énoncés (*cellule Markdown*), les zones de programmation ou de Shell (*cellule Code*), les zones de réponse (*Cellule Texte brut*).
- Les exploitations pédagogiques d'une *cellule Code* sont nombreuses (aide à faire découvrir une instruction, l'utiliser sous forme de Shell ou de programmation, permet de demander de corriger un script faux ou de compléter un script incomplet…).

2) Installer Anaconda ou Miniconda. Comment créer un Jupyter Notebook ?

Etape 1 : installer Jupyter Notebook sur votre ordinateur personnel

Le plus facile est d'installer Anaconda qui contient Jupyter Notebook. Choisir la bonne version (sous Windows, généralement la version 64 bits mais cela dépend de l'architecture de l'ordinateur) de l'exécutable au lien suivant : https://www.anaconda.com/products/individual



Il est possible d'installer Jupyter Notebook par l'intermédiaire de Miniconda, mais c'est beaucoup plus compliqué.

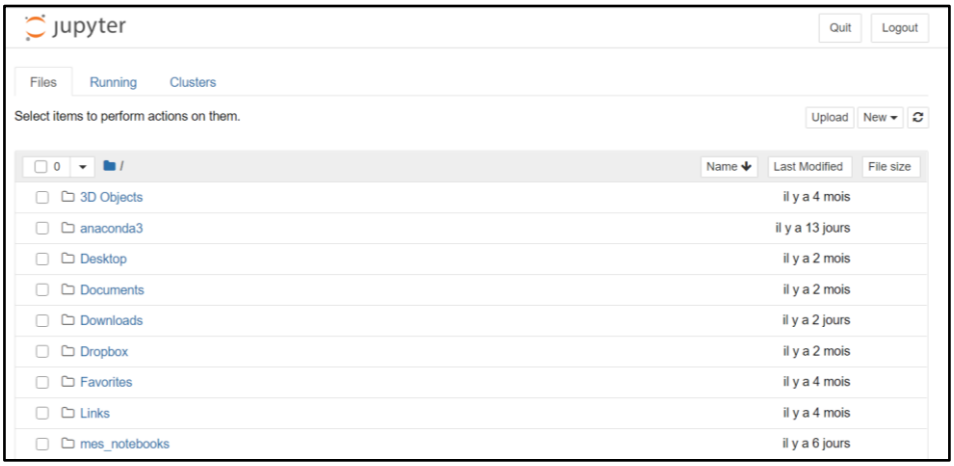
Voir ***annexe 2 : installation de Jupyter Notebook avec les distributions Anaconda ou Miniconda sous Windows.***

Etape 2 : Créer un Jupyter Notebook

Vous pourrez alors lancer Jupyter Notebook en cliquant sur *Launch* de l'icône 

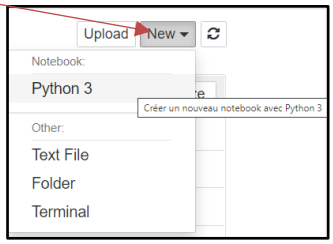
.

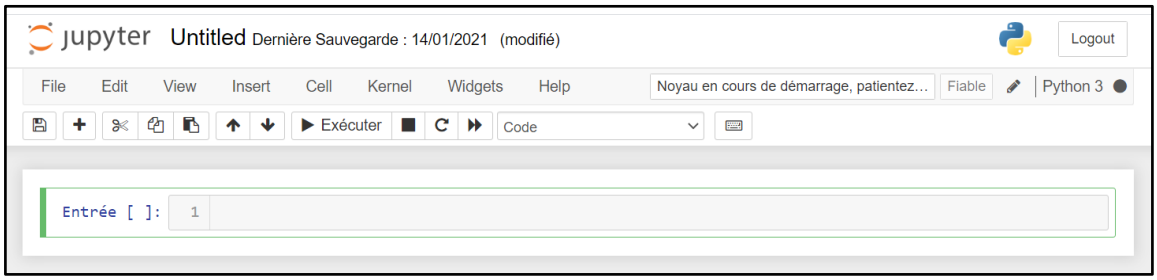
Jupyter Notebook s'ouvre dans votre navigateur par défaut à l'adresse *http://localhost:8888/tree#notebooks* :



Nous vous conseillons de créer un dossier *mes_notebooks* dans « *C : \Utilisateurs\ votre_nom* » pour stocker tous vos notebooks. Pour créer ce dossier il faudra aller dans le répertoire qui porte votre nom *votre_nom* car il ne sera pas possible de le faire avec l'interface de Jupyter. Il est aussi possible de le créer à l'aide de Jupyter Notebook par *New/Folder*.

Cliquer alors sur *New/Python 3* pour créer un nouveau Jupyter Notebook :





La première cellule qui apparaît est une *cellule Code*.

Vous pouvez ajouter  ou  supprimer autant de cellules que vous souhaitez aux trois « formats » suivants :

- *cellule Code* : Shell ou zone de programmation
- *cellule Texte brut* : zone où les élèves pourront taper leurs réponses sans format de texte (ni éditeur d'équations).
- *cellule Markdown* : zone où vous allez écrire votre texte en langage Markdown, et les formules mathématiques en Latex. Le langage html est aussi supporté mais ne sera pas conservé si vous souhaitez convertir votre Jupyter Notebook au format pdf (via un fichier Latex ou directement). Vous pouvez aussi insérer dans ce type de cellule des liens hypertextes et des images.

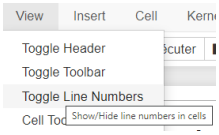
Voici quelques liens pour apprendre à compléter votre *cellule Markdown* à l'aide :

- du langage de balisage Markdown :
  - https://github.com/capytale/capytale/blob/master/PRES%20-%20Bases%20pour%20rediger%20un%20notebook.ipynb
  - https://seps.flibuste.net/markdown_help
  - https://wprock.fr/blog/markdown-syntaxe/

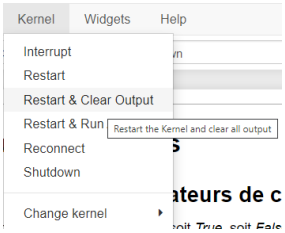
- de formules mathématiques écrites en Latex :
  - https://fr.wikipedia.org/wiki/Aide:Formules_TeX
  - Online Latex Equation Editor - Sciweavers

Quelques remarques utiles :

- Il est possible de numéroter les lignes de code des cellules Code par *View/Toggle Line Number*



Avant de déposer votre notebook sur GitHub, penser à bien effacer les *Output* de vos scripts par *Kernel/Restart & Clear Output* (ou *Cell/All Output/Clear*).

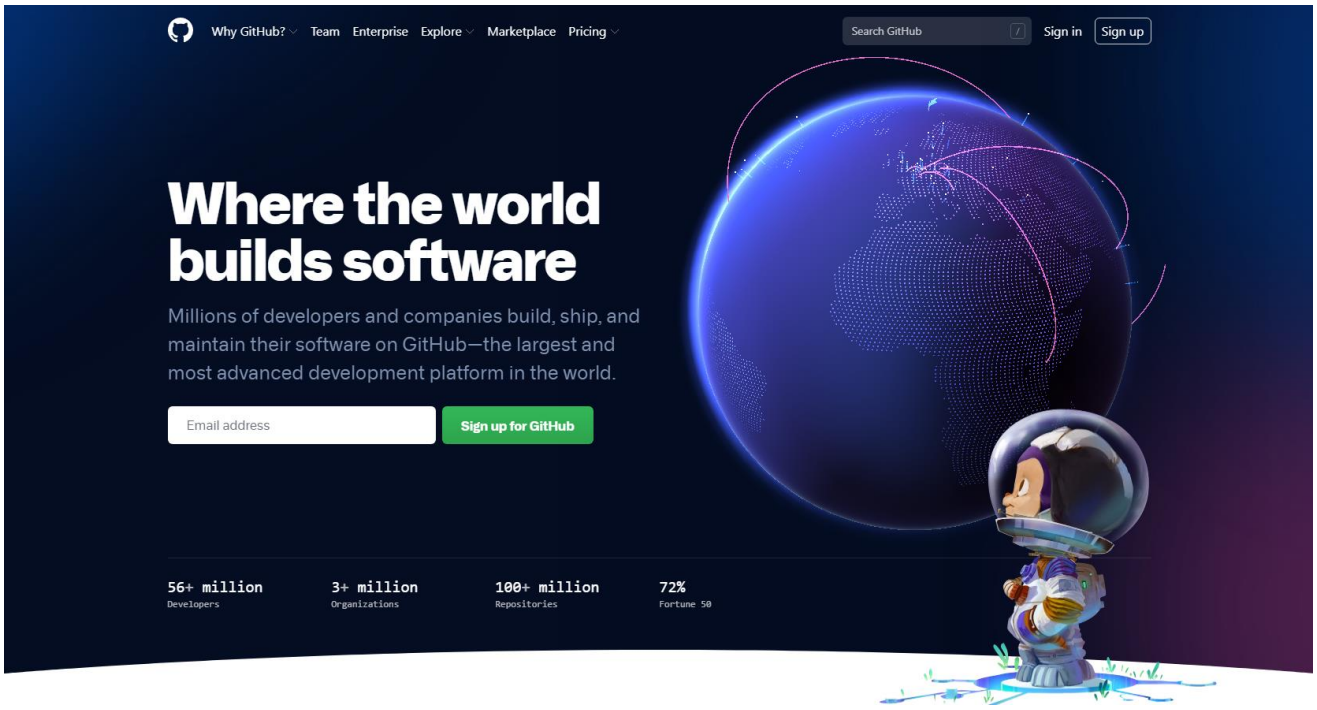


3) Créer un compte GitHub et un environnement en ligne avec Binder

Etape 1 : ouvrir un compte GitHub

Créer un compte gratuit sur la plateforme GitHub où nous allons stocker nos Jupyter Notebooks.

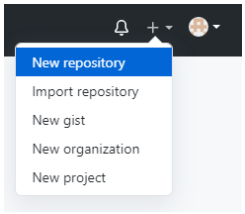
Suivre le lien suivant : https://github.com/ et cliquer sur le bouton Sign up for GitHub ou Sign up.

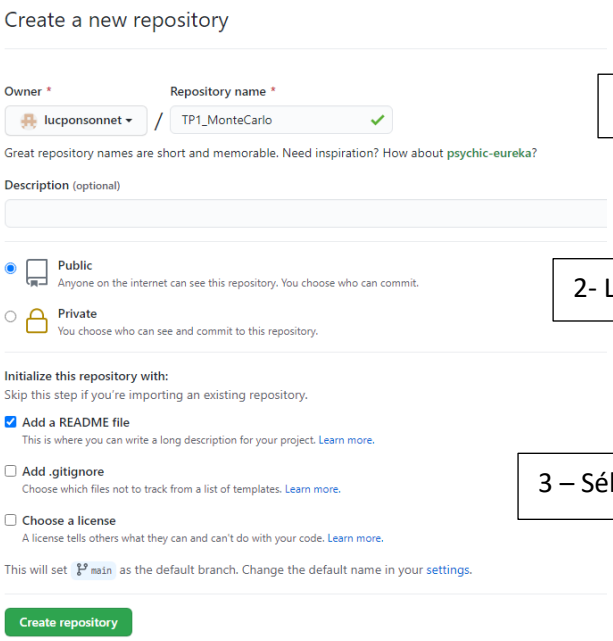


Au besoin, vous pouvez lire la page suivante : https://openclassrooms.com/fr/courses/5641721-utilisez-git-et-github-pour-vos-projets-de-developpement/6113011-demarrez-votre-projet-avec-github

Etape 2 : créer un *New repository* et y déposer votre TP Jupyter Notebook

Une fois connecté à votre compte GitHub par Sign in, cliquer sur New ou sur +, et sélectionner *New repository* :



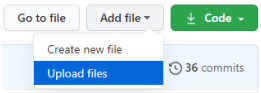


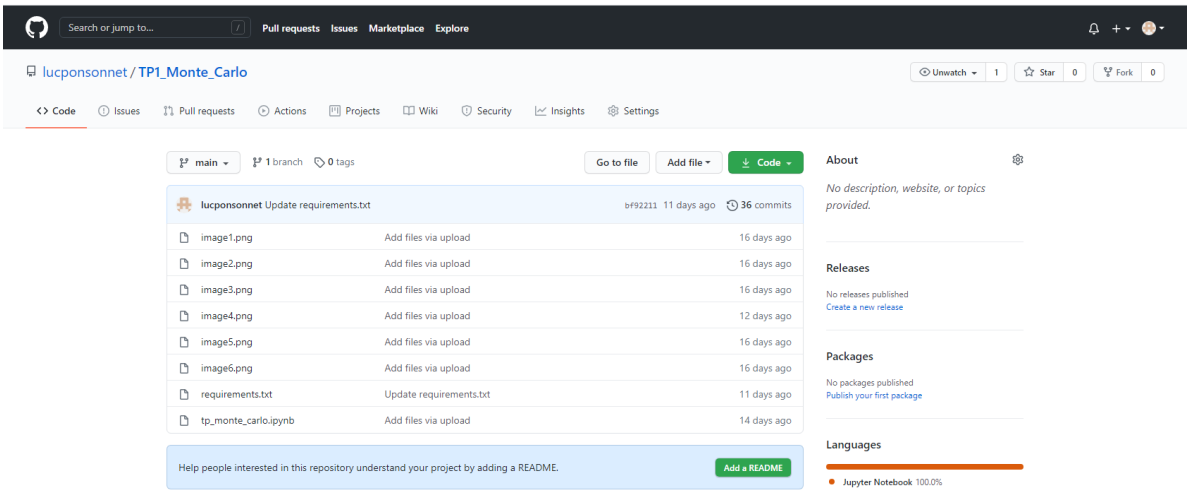
1- Ecrire le nom de votre TP

2- Laisser par défaut : *Public*

3 – Sélectionner *Add a README file*

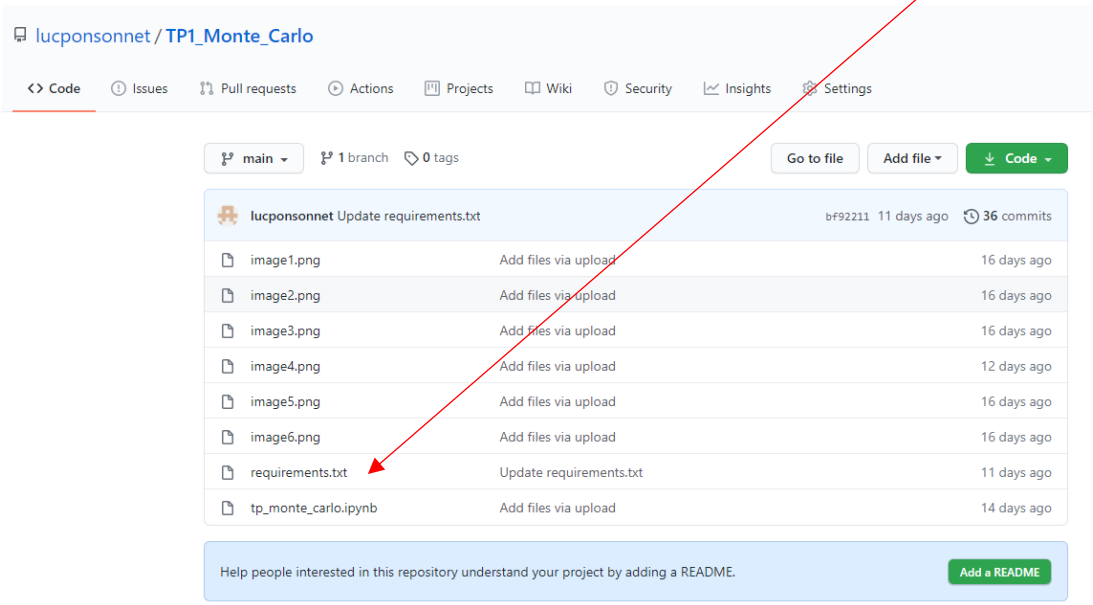
Vous pouvez alors glisser-déposer tous vos éléments (généralement votre fichier au format ipynb, des images et éventuellement un fichier *requirements.txt* (voir remarque ci-après) qui composent votre TP ou bien sélectionner *Add file/Upload files* par :



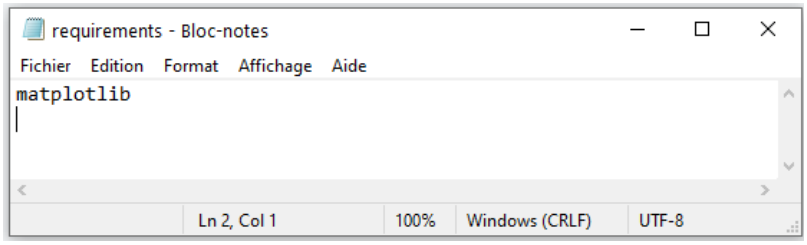


Une remarque très importante :

Il sera parfois nécessaire d'ajouter dans le dépôt GitHub un fichier texte qui sera nommé obligatoirement *requirements.txt* (que l'on pourra créer par exemple avec *Bloc-note* ou à l'aide GitHub même par *Add file/Create new file*) pour que l'environnement généré par Binder (voir étape suivante) puisse accéder aux librairies *Matplotlib* ou *Numpy*. Il n'est pas nécessaire de le faire pour les modules *Math* et *Random*.

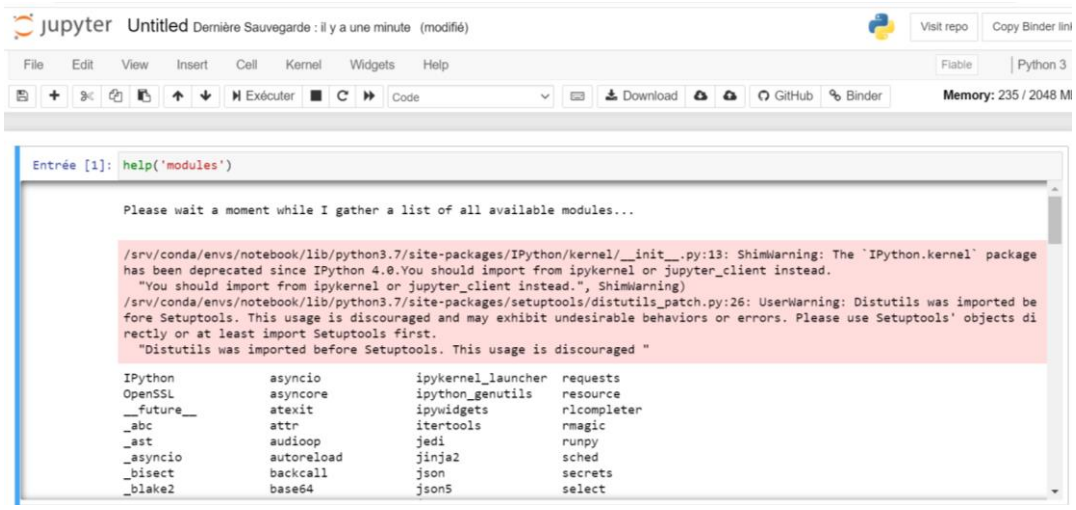


Voici par exemple le contenu du fichier *requirements.txt* du *TP1_Monte_Carlo* :



Remarque :

Avec la commande *help('modules')*, il est possible d'obtenir les modules de Python « natifs » dans Jupyter Notebook.

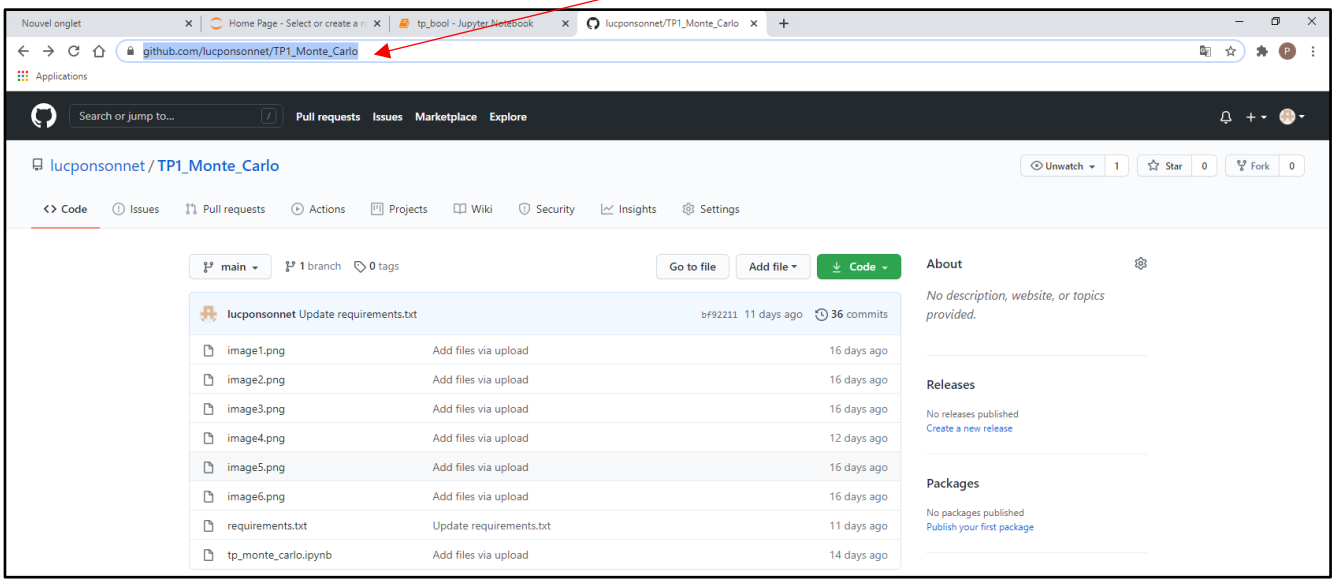


Etape 3 : créer l'environnement en ligne grâce à Binder

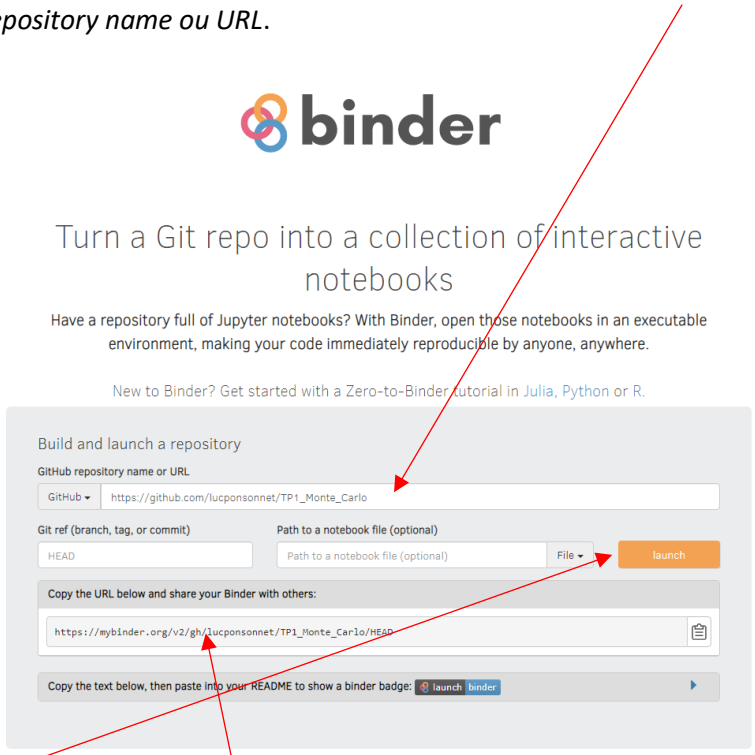
Binder va déployer un environnement Jupyter en ligne à partir du TP notebook qui a été déposé sur GitHub. Il pourra être utilisé par les élèves sans aucune installation préalable par un simple clic sur un lien Binder.

Lorsque vous êtes dans le répertoire de votre TP sur la plateforme GitHub. Copier l'adresse GitHub de ce TP.

Ici : https://github.com/lucponsonnet/TP1_Monte_Carlo



Puis aller sur Binder en suivant ce lien : https://mybinder.org/ et copier l'adresse GitHub de votre TP dans la barre intitulée *Github repository name ou URL*.



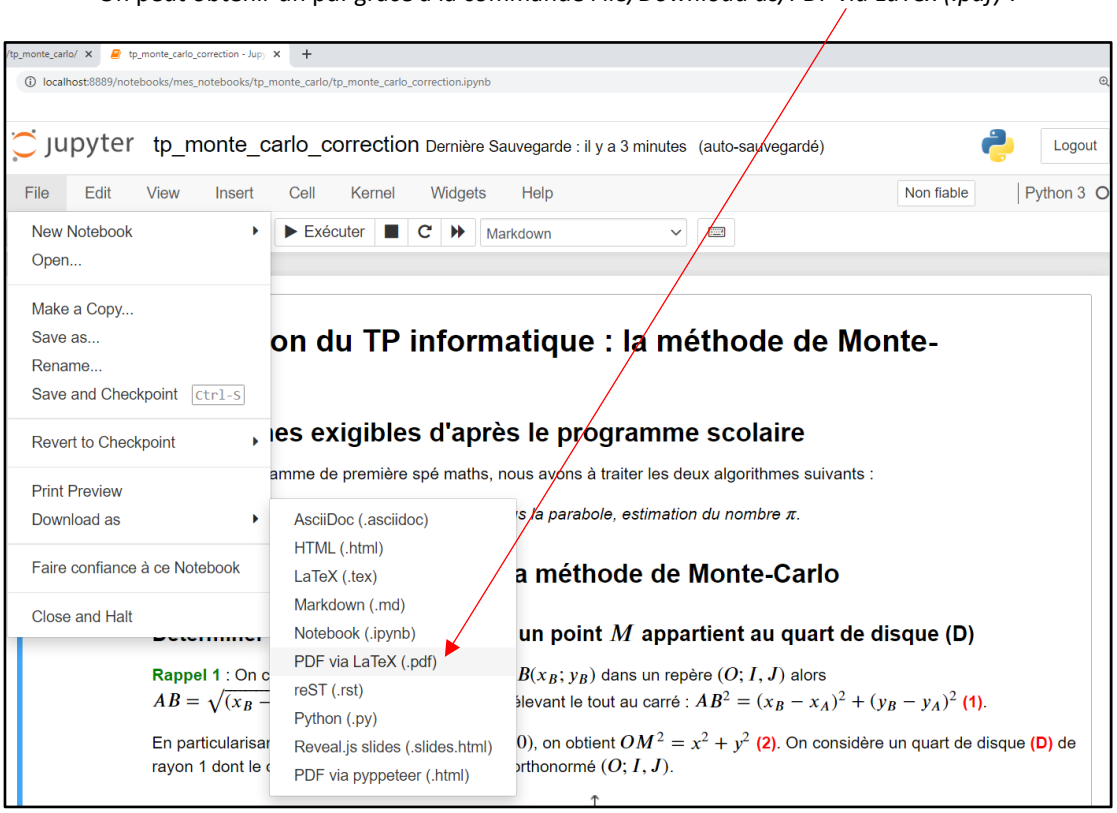
Cliquer sur *launch*.

Le lien suivant pourra être donné à vos élèves !

4) Comment créer un *pdf* à partir d'un Jupyter Notebook ?

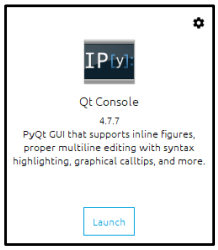
a) Sur votre ordinateur personnel

- Vous devez avoir préalablement installé MikTex : https://miktex.org/download .
  Mettre à jour MiKTek : par *MiKTek Console /Updates / Check for Updates*
- On peut obtenir un pdf grâce à la commande *File/Download as/PDF via LaTex (.pdf)* :

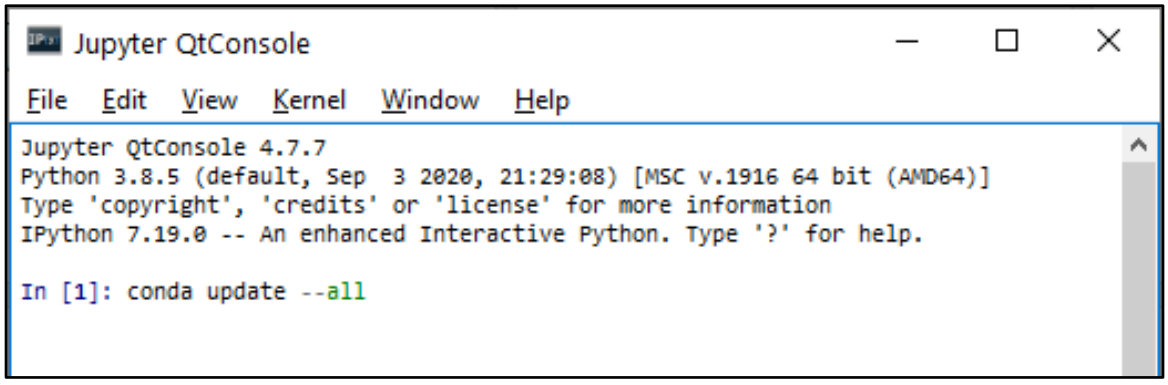


Il sera parfois nécessaire d'installer la dernière version de Jupyter en suivant la démarche suivante :

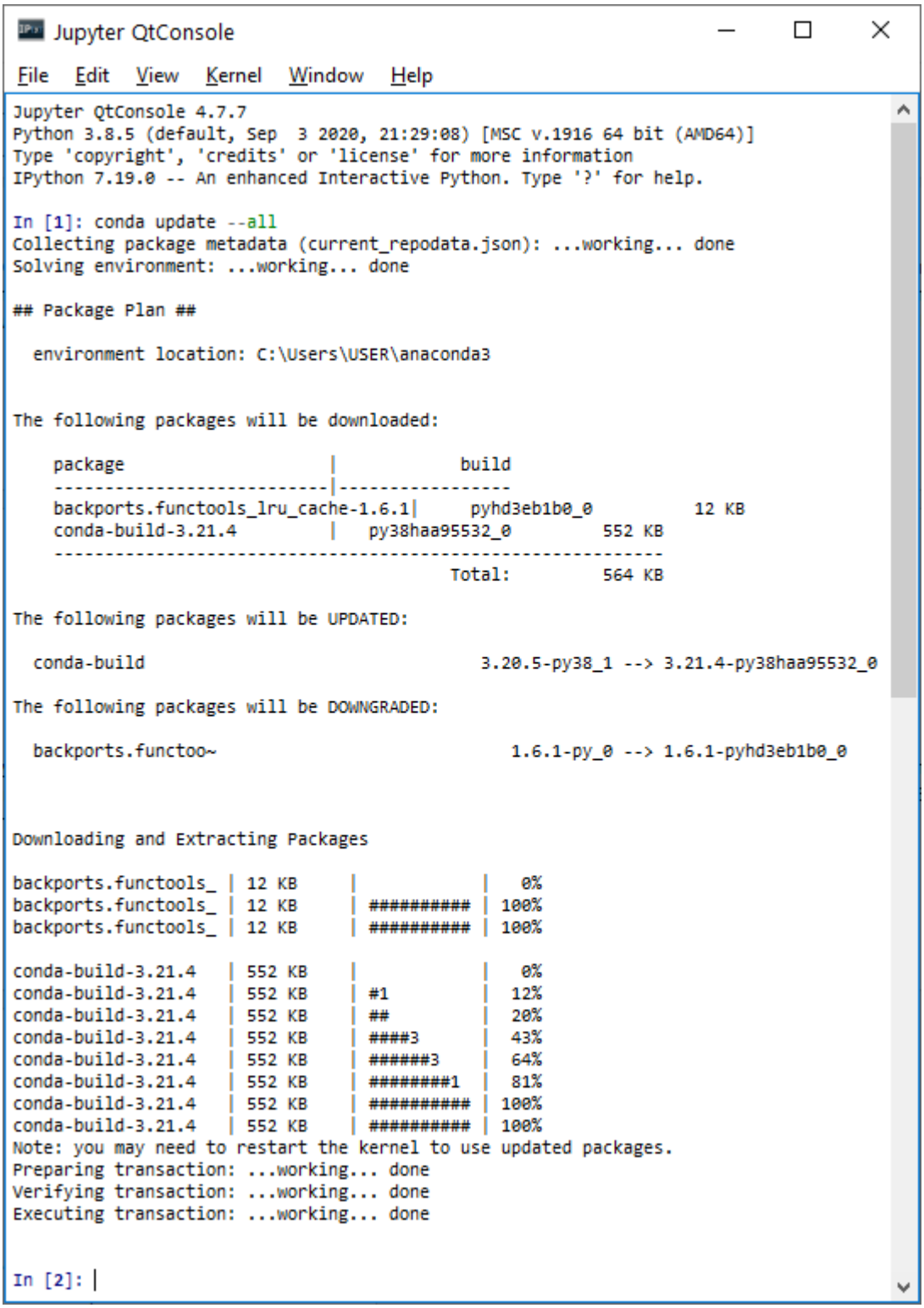
- Dans Qt Console d'Anaconda, taper : *conda update --all*.



- Valider en appuyant sur la touche Entrée.



- Attendre un moment, il apparaît alors la fenêtre suivante :



- Redémarrer Anaconda par *Ctrl + R.*

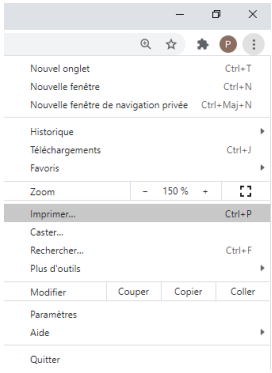
Remarques importantes :

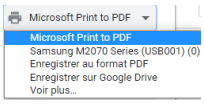
- Pour ceux qui s'y connaissent un peu en Latex, il sera parfois préférable d'obtenir à partir du Notebook de Jupyter le fichier *.tex* intermédiaire (*File/Download as/Latex (.tex)*) avant de générer le pdf. Ce fichier .tex nous permettra de modifier :

    - la taille des images en remplaçant l'instruction **\includegraphics** {image1.png} par **\includegraphics**[width=5cm]{ image1.png}. Cela qui permettra de redimensionner la taille de l'image comme on le souhaite.
    - Concernant les titres #, ##, ##..., si vous avez ajouté des numérotations (exemple : ## 1-Introduction), il faudra les enlever car lors de la compilation en pdf, une numérotation propre au fichier Latex sera mise en lien avec l'écriture Markdowon des titres.
    - Seuls le langage Markdown et les formules Latex seront correctement convertis en pdf donc pas le langage en html.

b) Sur l'ordinateur d'un élève

- Créer un fichier au format html par *File/Download as/HTML (.html)*.
- Déposer ce fichier .html dans un dossier qui contient toutes les images du TP (ils pourront les télécharger de l'environnement Binder).
- Puis imprimer au format pdf avec le navigateur cette page web. Par exemple avec Chrome :





S'ils n'ont pas de créateur de pdf, ils pourront installer pdf creator : https://pdfcreator.fr/

c) Remarque

Il est possible d'obtenir un pdf à partir d'un notebook via un fichier html. Mais l'enseignant aura intérêt à installer MikTex car ainsi il pourra faire des modifications via ce fichier Tex avant la compilation en un pdf de qualité (voir un exemple, **annexe 3 : *exemple de pdf créé en passant par un fichier .tex***).

5) Annexes

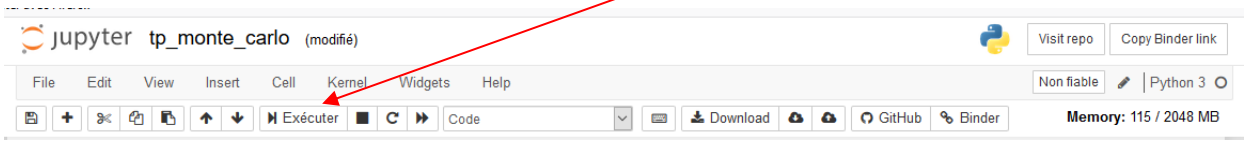
a) Annexe 1 : consignes élèves concernant les TP Jupyter Notebooks

Ouvrir le TP

- Cliquer sur le lien suivant et attendre que l'environnement de travail soit créé par Binder : https://mybinder.org/v2/gh/lucponsonnet/TP1_Monte_Carlo.git/HEAD
- Puis cliquer sur le fichier d'extension *.ipynb* : *tp_monte_carlo.ipynb*.

1) Pour lancer une *cellule code* et compléter une *cellule texte*

- ***Cellule Code* (où il est marqué Entrée [ ]) :**
  Cliquer dans la *cellule Code*, puis cliquer sur le bouton *Exécuter* (vous pouvez aussi utiliser le raccourci clavier CTRL + Entrée) :



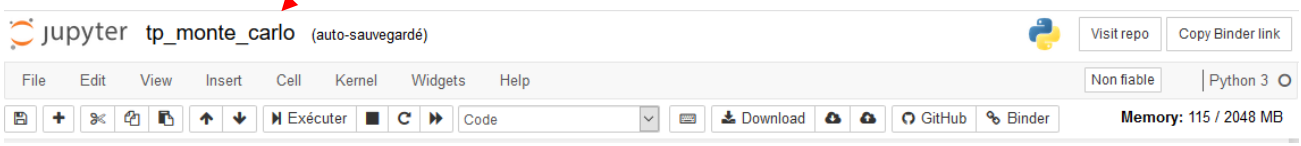
- ***Cellule Texte brut* :**
  Vous devrez saisir dans la *cellule Texte* la réponse à la question posée. Parfois, il ne sera pas possible d'écrire un exposant. A vous d'être astucieux et d'écrire par exemple OA^2 au lieu de OA^2 qui n'est pas possible d'écrire dans le texte brut.

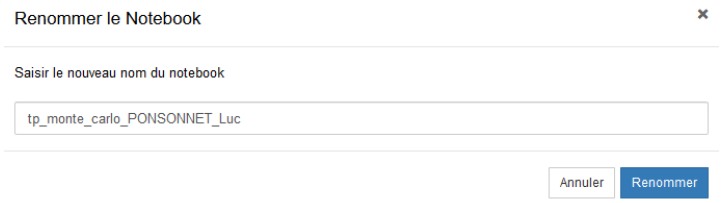
De temps en temps :

- A l'aide de l'icône disquette, il sera nécessaire d'enregistrer votre travail.
- D'actualiser la page si la connexion avec le serveur s'est interrompue (touche F5 ou l'icône ↻ du navigateur).

2) Pour enregistrer le TP une fois terminé au format html

- **Pour enregistrer votre TP au format html**
  - Cliquer sur le titre du TP, et renommer le TP en y ajoutant votre NOM et prénom :

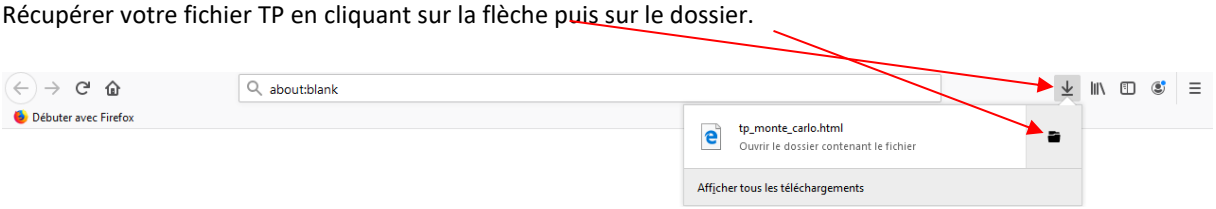




  - Enregistrer votre TP par *File/Download as/HTML (.html).* Il sera enregistré par défaut dans le dossier *Téléchargements*. Si on le souhaite, il est possible de faire choisir le lieu de téléchargement.

- **Pour déposer le fichier sur Pronote ou Moodle**

  Récupérer votre fichier TP en cliquant sur la flèche puis sur le dossier.

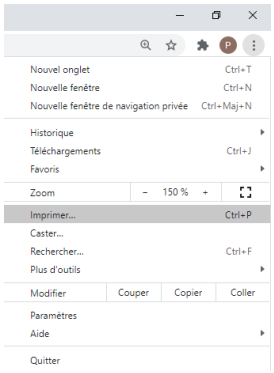


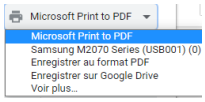
  Vous pouvez déposer votre fichier *tp_monte_carlo.html* dans un dépôt Moodle ou Pronote selon les consignes de votre enseignant.

- **Remarque :** il se peut que les images n'apparaissent plus dans le fichier .html car le fichier html n'est plus dans le répertoire qui contient les images du TP.

3) Pour enregistrer le TP au format pdf

- Créer un fichier au format html par *File/Download as/HTML (.html)*.
- Déposer ce fichier .html dans un dossier qui contient toutes les images du TP (vous pourrez les télécharger de l'environnement Binder).
- Puis imprimer au format pdf avec le navigateur cette page web. Par exemple avec Chrome :

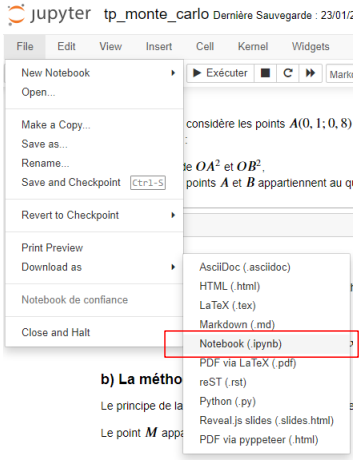




S'ils n'ont pas de créateur de pdf, ils pourront installer pdf creator : https://pdfcreator.fr/

4) Pour enregistrer le TP au format ipynb

- Enregistrez votre travail sous la forme *TP_NOM_Prenom*,
- Vous pouvez aussi remettre votre TP à votre professeur au format *.ipynb* par :



b) Annexe 2 : installation de Jupyter Notebook avec les distributions Anaconda ou Miniconda sous Windows.

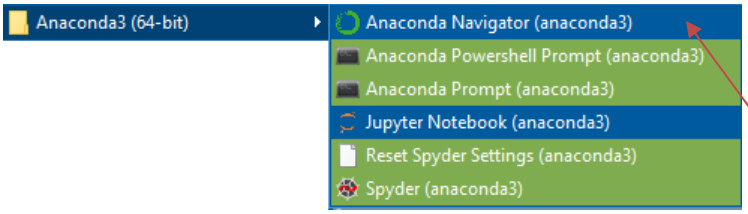
A) Anonconda VS Miniconda ?

Le plus simple pour installer *Jupyterlab* et *Jupyter Notebook* est d'installer la distribution *Anaconda* (qui contient le gestionnaire de paquets *Conda*, plus les bibliothèques scientifiques, plus un environnement de développement…), seulement *Anaconda* nécessite un espace disque de plusieurs giga-octets.
La distribution *Miniconda* est une version allégée d'*Anaconda* qui contient aussi le gestionnaire de paquets *Conda*. Elle occupe moins d'espace disque mais comme vous le verrez, l'installation de *Jupyterlab* et *Jupyter Notebook* est beaucoup plus difficile par Miniconda.

B) Avec Anaconda

1) Télécharger le fichier qui va vous permettre d'installer *Anaconda*

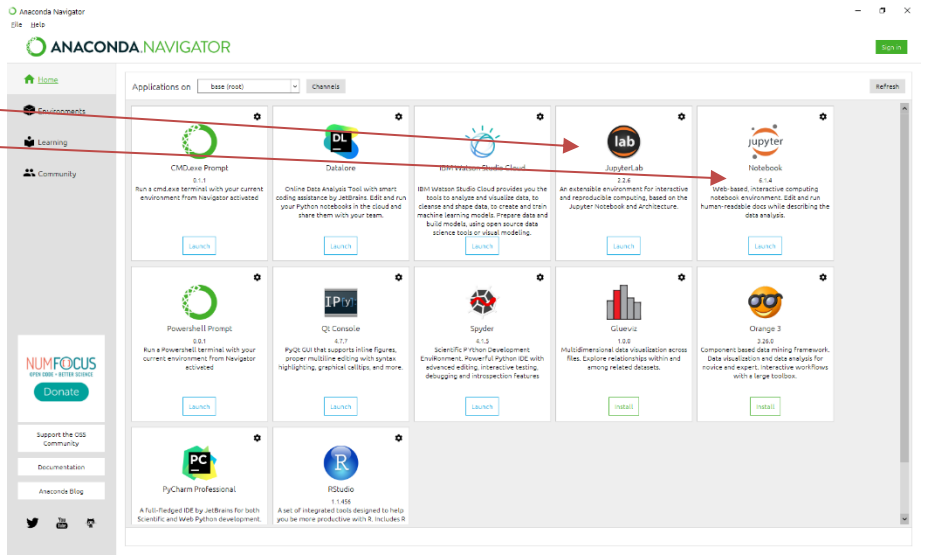
- Télécharger le fichier *Anaconda3-2020.11-Windows-x86_64.exe* (ou 32 bits selon votre ordinateur) au lien suivant : https://www.anaconda.com/products/individual#windows
- Lancer l'installation de Anaconda en double cliquant sur ce fichier, et suivre les indications de chacune des fenêtres d'installation.
- Vous constaterez dans le menu *Démarrer* la nouvelle rubrique suivante :



2) Lancer *JupyterLab* et *Jupyter Notebook* dans le navigateur web par défaut

Le plus facile est de démarrer le navigateur d'Anaconda en cliquant sur *Anaconda Navigator (anaconda3)*. On obtient alors :

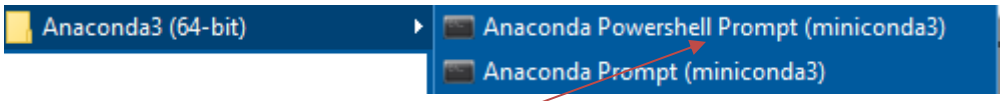
Pour lancer *JupyterLab* ou *Jupyter Notebook*, cliquer sur les icônes correspondantes.



C) Avec Miniconda

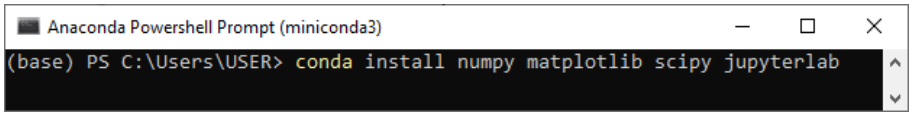
1) Télécharger le fichier qui va vous permettre d'installer *Miniconda*

- Télécharger le fichier *Miniconda3-latest-Windows-x86_64.exe* (ou la version 32 bits selon votre ordinateur) au lien suivant : https://docs.conda.io/en/latest/miniconda.html
- Lancer l'installation de Miniconda en double cliquant sur ce fichier, et suivre les indications de chacune des fenêtres d'installation.
- Vous constaterez dans le menu *Démarrer* la nouvelle rubrique suivante :



2) Utilisation de Conda pour installer des modules complémentaires dont Jupyterlab

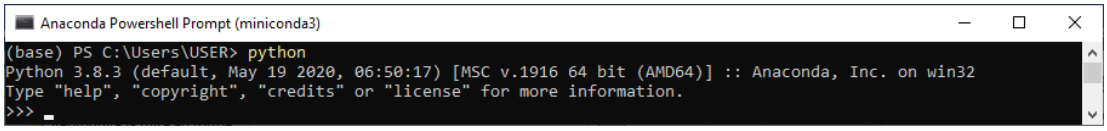
- Lancer le Shell ci-dessus en cliquant sur *Anaconda Powershell prompt (miniconda3)*, écrire dans le Shell la ligne de commande suivante, et appuyer sur la touche *Entrée* :



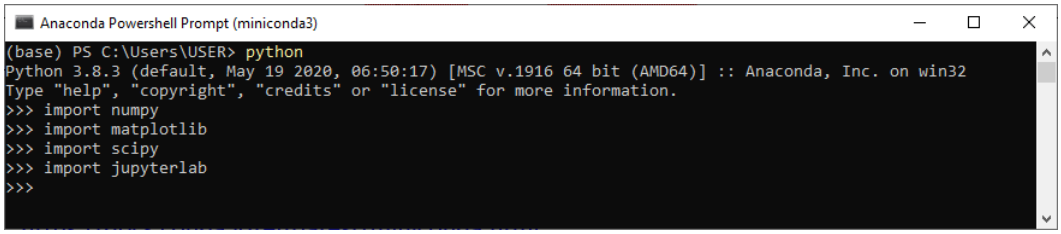
- A la question **Proceed ([y]/n) ?**, répondre **y**.

3) Vérifier la bonne installation des modules complémentaires

- Dans le Shell, lancer l'interpréteur de Python en entrant *python*, on obtient alors :



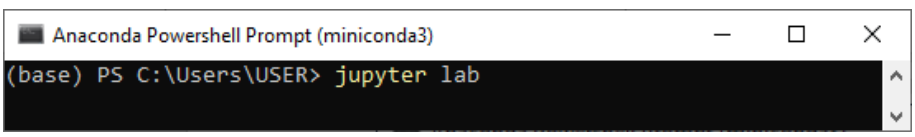
- Puis dans l'interpréteur Python, vérifier la bonne importation des modules par :



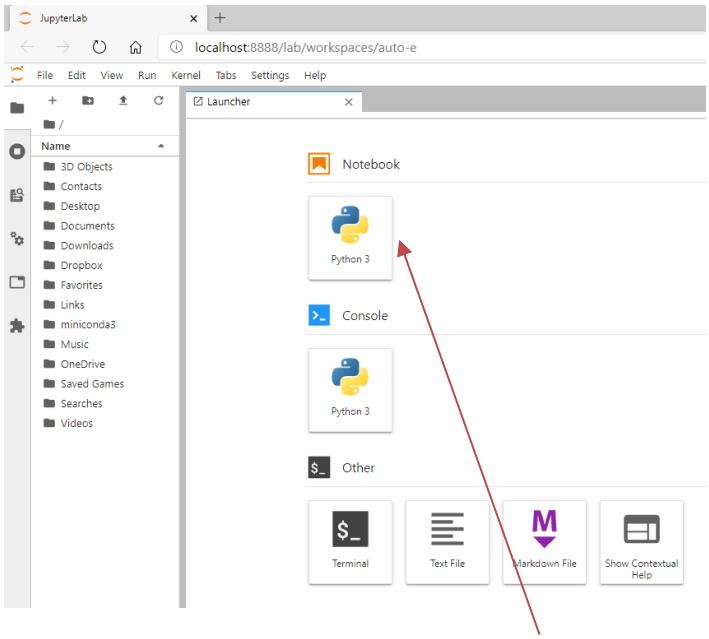
S'il n'apparaît aucune erreur comme ici, cela signifie que les modules ont bien été installés. Pour quitter l'interpréteur de Python et revenir au Shell, taper *exit()*.

4) Lancer *JupyterLab* dans le navigateur web par défaut

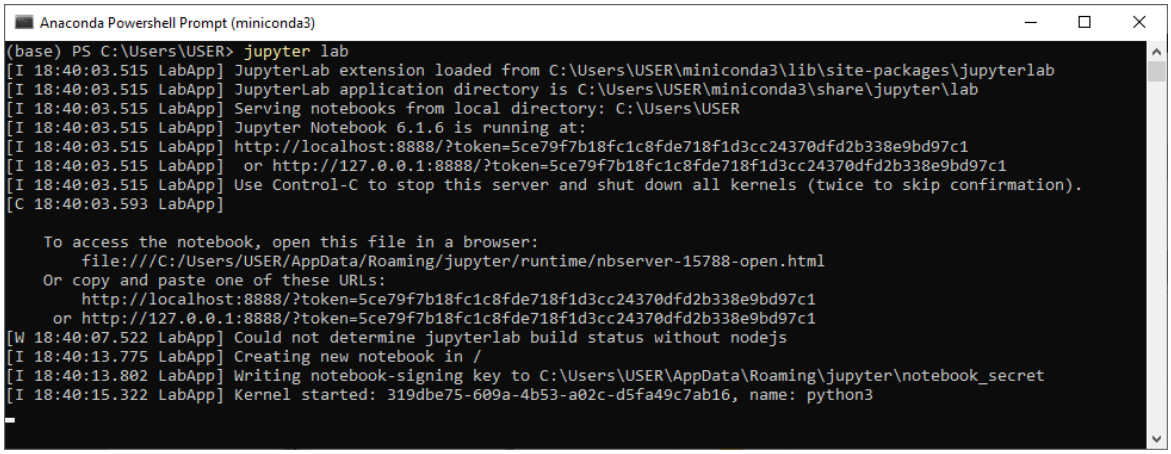
- Une fois dans le Shell entrer (attention à bien respecter l'espace) *jupyter lab* comme ci-dessous :



- Cette commande ouvre votre navigateur web par défaut et démarre *JupyterLab*.

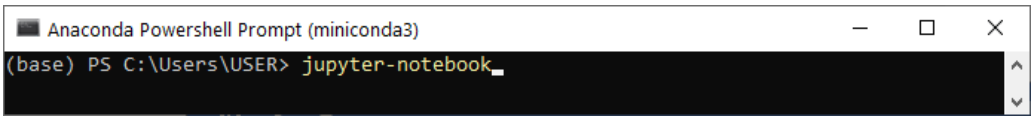


- On peut obtenir un *Notebook* de *Jupyter* en cliquant sur l'icône *Python 3*.
- Attention de ne pas fermer la fenêtre suivante du Shell :

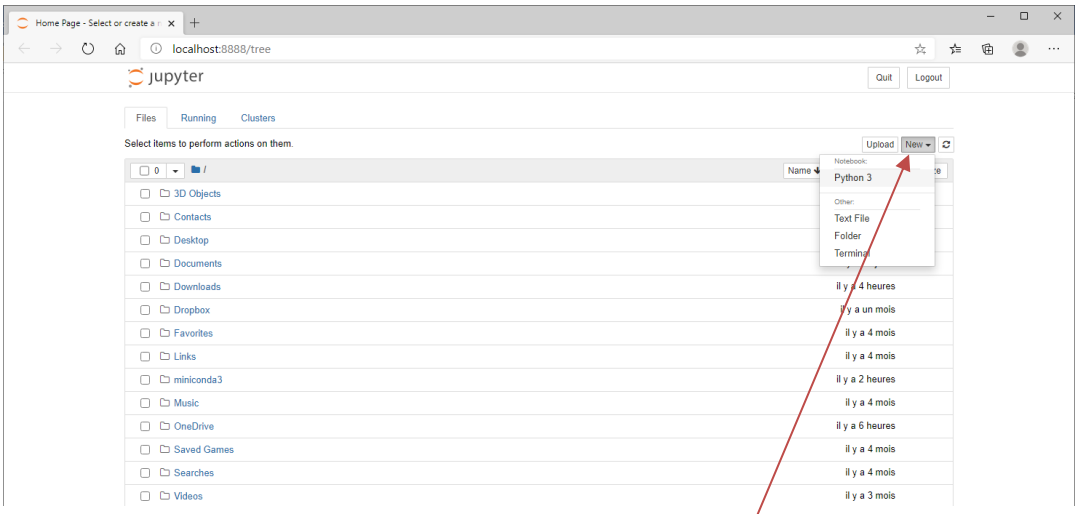


5) <u>Lancer *Jupyter Notebook* dans le navigateur web par défaut</u>

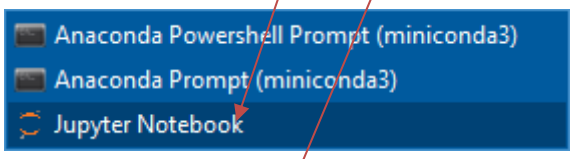
- Taper et valider la commande ci-dessous dans le Shell :



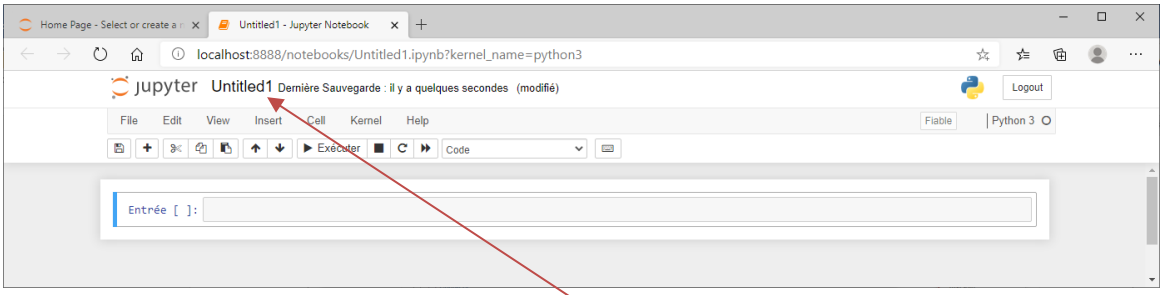
- On obtient l'interface de *Jupyter Notebook* dans le navigateur web par défaut :



- On obtient par la suite un lien direct vers *Jupyter Notebook* dans le menu *Démarrer* :



- Pour créer un *Notebook de Jupyter*, cliquer sur *New*, puis sélectionner *Python 3*. On obtient alors le *Notebook* vierge avec une seule cellule et dans un autre onglet du navigateur :



- Donner un nom à votre *Notebook* en cliquant sur *Untitled*. Si le nom est *mon_notebook*, le fichier aura pour extension *.ipynb*, et il sera enregistré dans le répertoire qui contient le dossier *miniconda3*.

c) Annexe 3 : exemple de pdf créé en passant par un fichier .tex

Le logiciel MikTek et Anaconda ont été installés et mis à jour sur l'ordinateur personnel (voir pages 11 et 12).

tp_monte_carlo_correction

January 23, 2021

1 Correction du TP informatique : la méthode de Monte-Carlo

1.1 Algorithmes exigibles d'après le programme scolaire

D'après le programme de première spé maths, nous avons à traiter les deux algorithmes suivants :

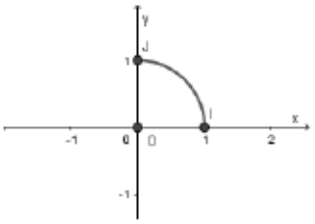
Méthode de Monte-Carlo : estimation de l'aire sous la parabole, estimation du nombre π.

1.2 Estimation du nombre π par la méthode de Monte-Carlo

1.2.1 Déterminer un critère pour savoir si un point M appartient au quart de disque (D)

Rappel 1 : On considère les points $A(x_A;y_A)$ et $B(x_B;y_B)$ dans un repère $(O;I,J)$ alors $AB = \sqrt{(x_B-x_A)^2+(y_B-y_A)^2}$ et donc en élevant le tout au carré : $AB^2 = (x_B-x_A)^2+(y_B-y_A)^2$ **(1)**.

En particularisant **(1)** aux points $M(x;y)$ et $O(0,0)$, on obtient $OM^2 = x^2+y^2$ **(2)**. On considère un quart de disque **(D)** de rayon 1 dont le centre O est l'origine d'un repère orthonormé $(O;I,J)$.



Question 1 : On considère les points $A(0,1;0,8)$ et $B(0,7;0,8)$ du repère $(O;I,J)$. A l'aide de la formule **(2)** et de votre calculatrice, taper dans la *cellule texte* ci-dessous : * les valeurs de OA^2 et OB^2, * et dire si les points A et B appartiennent au quart de disque **(D)** de centre O et de rayon 1.

$OA^2 = 0,1^2+0,8^2 = 0,65$ donc $OA^2 \leq 1$, et on déduit que $A \in$**(D)**.

$OB^2 = 0,7^2+0,8^2 = 1,13$ donc $OB^2 > 1$, et on déduit que $B \notin$**(D)**.

En résumé :

Soit $M(x;y)$ un point du plan doté d'un repère orthonormé $(O;I,J)$.

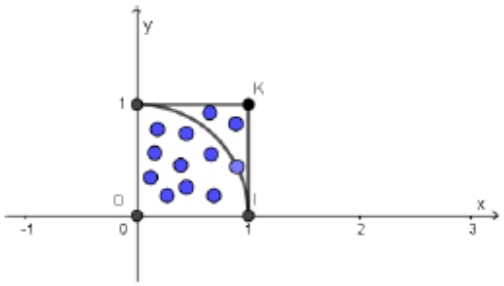
$M \in$ **(D)** $\Leftrightarrow OM \leq 1 \Leftrightarrow OM^2 \leq 1 \Leftrightarrow x^2+y^2 \leq 1$.

Autrement dit : si $x^2 + y^2 \leq 1$ alors le point $M(x;y)$ appartient au quart de disque **(D)**, frontière comprise.

1.2.2 La méthode de Monte-Carlo

Le principe de la méthode de Monte-Carlo consiste à tirer au hasard les coordonnées (x, y) d'un point M avec $x \in [0;1]$ et $y \in [0;1]$.

Le point M appartient soit au quart de disque **(D)**, soit au carré $OIKJ$.



$M(x;y) \in$ **(D)** $\Leftrightarrow x^2 + y^2 \leq 1$ d'après le résumé précédent.

Soit N est le nombre de points $M(x, y)$ tirés au hasard donc avec x et y des réels aléatoires de $[0;1]$.

Soit n le nombre de points $M(x, y)$ qui appartiennent au quart de disque **(D)**.

On admettra l'affirmation suivante :

Le rapport $\frac{n}{N}$ donne une approximation du quotient $\frac{\text{Aire de (D)}}{\text{Aire du carré } OIKJ}$.

Or $\frac{\text{Aire de (D)}}{\text{Aire du carré } OIKJ} = \frac{\frac{\pi \times 1^2}{4}}{1^2}$ car Aire de (D) $= \frac{\pi \times 1^2}{4}$ et Aire du carré $= 1^2$.

En simplifiant, on obtient que $\frac{\text{Aire de (D)}}{\text{Aire du carré } OIKJ} = \frac{\pi}{4}$.

En invoquant l'affirmation **(3)**, on déduit que $\frac{n}{N} \approx \frac{\pi}{4}$. Ainsi $\pi \approx \frac{4n}{N}$.

Question 2 : On suppose que sur 1000 tirages de points $M(x, y)$ de manière aléatoire avec $x \in [0;1]$ et $y \in [0;1]$, 800 d'entre eux appartiennent au quart de disque **(D)**. En déduire une valeur approchée de π que vous saisirez dans la *cellule texte* ci-après :

Nous avons $N = 1000$ et $n = 800$ donc $\frac{4 \times 800}{1000} = 3,2$. Une approximation de π par la méthode de Monte-Carlo est 3, 2.

1.2.3 Implémentation en langage Python de la méthode de Monte-Carlo pour donner une approximation du réel π

Exécuter plusieurs fois la *cellule code* suivante :

[1]:
```
from random import *
random()
```

[1]:
```
0.23980463858600443
```

Question 3 : A l'aide du mémento Python (cliquer sur le lien) dire ce que renvoie précisément la fonction *random()* ? Taper votre réponse dans la *cellule texte* ci-après :

Lorsque nous exécutons plusieurs fois la *cellule code*, nous constatons que la fonction *random()* renvoie une valeur aléatoire comprise entre 0 et 1. Le *Mémento Python* précise que la fonction *random()* renvoie une valeur pseudo-aléatoire qui appartient à $[0;1[$. On peut remarquer que la borne 1 n'est pas prise.

[2]:
```
from random import *

def Monte_Carlo(N):
    n=0
    for i in range(N):
        x=random()
        y=random()
        if x**2+y**2<=1:
            n=n+1
    return round((4*n)/N,5)

Monte_Carlo(1000000)
```

[2]:
```
3.13782
```

Donner une valeur approchée de π à l'aide de l'appel de fonction *Monte_Carlo(1000000)*. Saisir votre résultat dans la *cellule texte* ci-après :

Le script précédent donne par exemple $\pi \approx 3,13782$.

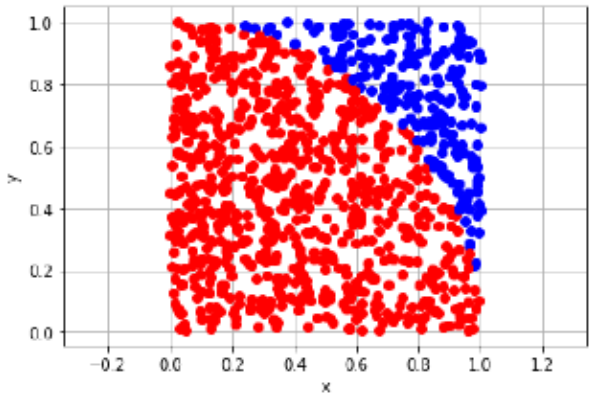
Exécuter le programme suivant qui : - affiche 1000 points aléatoires. - donne une approximation de π par la méthode de Monte-Carlo.

[3]:
```
from random import *
import matplotlib.pyplot as plt

def Monte_Carlo(N):
    plt.axis('equal')
    plt.xlabel("x")
    plt.ylabel("y")
    n=0
    for i in range(N):
        x=random()
        y=random()
        if x**2+y**2<=1:
            plt.plot(x,y,"or")
            n=n+1
```

```
        else :
            plt.plot(x,y,"ob")
    plt.grid()
    plt.show()
    return round((4*n)/N,5)

Monte_Carlo(1000)
```



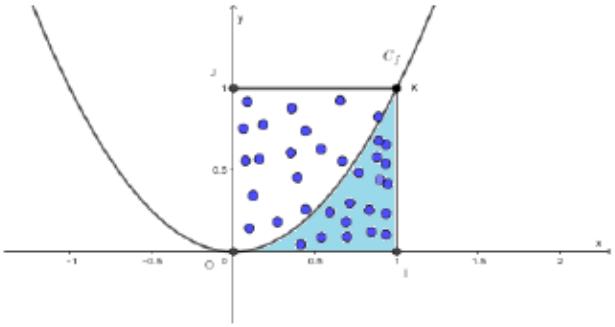
[3]: 3.108

1.3 Estimation de l'aire de la surface sous la courbe d'une parabole par la méthode de Monte-Carlo

1.3.1 La problématique

Soit C_f la parabole représentative de la fonction f définie sur $\mathbb{R}$ par $f(x) = x^2$.

On considère la surface bleutée **(S)** de la figure ci-dessous, définie par $\{M(x,y) \ / \ 0 \leq x \leq 1$ et $0 \leq x \leq x^2\}$.



On souhaite obtenir une approximation par la méthode de Monte-Carlo, de l'aire de la surface bleutée **(S)** délimitée par la parabole C_f, l'axe des abcisses et la droite d'équation $x = 1$. ### L'implémentation en langage Python d'une fonction aire_parabole(N) Saisir le script ci-contre dans la *cellule code* ci-dessous :

```
1  from random import *
2
3  def aire_parabole(N):
4      n=0
5      for i in range(N):
6          x=random()
7          y=random()
8          ...
9          ...
10         ...
11 aire_parabole(1000000)
```

Compléter les lignes 8), 9) et 10) pour que le script affiche une approximation de l'aire de la surface **(S)** avec deux chiffres après la virgule :

[5]:
```
from random import *

def aire_parabole(N):
    n=0
    for i in range(N):
        x=random()
        y=random()
        if y<=x**2:
            n=n+1
    return round(n/N,2)
aire_parabole(1000000)
```

[5]: `0.33`

Tapper dans la *cellule texte* suivante la valeur que vous obtenez :

Une approximation de l'aire de la surface **(S)** avec deux chiffres après la virgule est par exemple de : 0,33.

2 Quelques remarques en lien avec ce TP informatique

2.0.1 Qui a inventé la méthode de Monte-Carlo ?

"Le terme méthode de Monte-Carlo, ou méthode Monte-Carlo, désigne une famille de méthodes algorithmiques visant à calculer une valeur numérique approchée en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. Le nom de ces méthodes, qui fait allusion aux jeux de hasard pratiqués au casino de Monte-Carlo, a été inventé en 1947 par Nicholas Metropolis, et publié pour la première fois en 1949 dans un article coécrit avec Stanislaw Ulam". D'après Wikipédia

- Nicholas Metropolis (1915-1999) : physicien gréco-américain

- Stanislaw Ulam (1909-1984) : mathéméticien polonais

2.0.2 Une correction vidéo du 3. b)

Voici une correction vidéo de la question 3. b) pour récompenser ceux qui sont allés jusqu'au bout de ce TP informatique !

Lien vers la vidéo