Choses à retenir pour l'examen

1 Ligne de commande : ce qu'il faut retenir

— Les commandes données tableau 1
— Savoir qu'il faut éviter les espaces dans les noms de fichiers

Commande	Utilisation
`pwd`	Emplacement dans l'arborescence pour donner le répertoire courant
`ls`	Lister les fichiers et les sous-dossiers, « -l » pour une version exhaustive, « -t » pour trier par date, « -r » pour inverser l'ordre, « -a » pour afficher les fichiers cachés
`cd`	Déplacement dans l'arborescence. «../» pour remonter d'un dossier, « ./ » dossier courant « / » racine, « ~ » home.
`mv`	déplacer ou renommer un fichier ou des répertoires
`cp`	copier un fichier ou des répertoires
`rm`	supprimer un fichier ou des répertoires
`mkdir`	créer un répertoire
`find`	trouver un fichier ou un dossier « -name » pour indiquer le nom du fichier
`chmod`	changer les droits sur un fichier, « +x » pour rendre exécutable
`grep`	trouver les lignes contenant une chaîne de caractère précise dans un fichier
`more/tail/head`	afficher un morceau de fichier
`sed`	faire des rechercher/remplacer dans un fichier
`zip` / `unzip`	compression et décompression au format zip
`df␣-h`	connaître l'espace disque
`du␣-h`	connaître la taille d'un dossier
`ssh`	connexion à une machine distante
`scp`	transfert de fichier d'un ordinateur à un autre
`&`	pour rendre le processus non bloquant
ctrl + C	arrêter le processus en cours d'exécution dans le terminal

TABLEAU 1 – Commandes de base à retenir.

2 Python, les bases : Ce qu'il faut retenir

— Savoir découper un problème complexe en superposition de problèmes plus simples. Il faut donc toujours prendre le temps de planifier son programme pour le découper en morceaux plus simple (en particulier avec des fonctions).
— Il est **indispensable de commenter son code**. En particulier les fonctions.
— Pour nommer une variable, elle doit être explicite, suivre une convention de nommage et être utilisable par le plus grand nombre (pas d'accents ni de noms français !)
— *Toutes les variables nécessaires à une fonction doivent être appelées lors de l'exécution de la fonction.*
— *Structure de la documentation d'une fonction : description, variables d'entrées (avec leur type si possible), variable de retour (avec leur type si possible).*
— *La structure générale d'un code python :*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Descriptif␣du␣fichier
"""

# Importation des librairies

# Definition des fonctions

# Programme principal
if __name__ == "__main__":
   pass
```

3 Python, opérations et type de base : Ce qu'il faut retenir

— Les types de bases utilisables sous python (`str`, `int`, `float`, `list`, `tuple`
— *Les options de formatage accessibles pour mettre en forme des variables en particulier pour mettre en forme un nombre flottant.* Ne pas utiliser de '+'.
— Différencier les objets mutables et les objets immutables.
— Comprendre la notion d'espace de nommage et les conséquences en terme de portée des variables.
— Savoir parcourir des listes, dictionnaires, etc. En sachant utiliser, en particulier les structures suivantes :
  — `for element in liste`,
  — `for indix,element in enumerate(liste)`
  — `for key,value in dict.items()`
— Utilisation des boucles `for` et `while` ainsi que des conditions `if`, `elif`, `else`.
— Savoir ouvrir (`with open():`) et fermer un fichier, le lire ou y écrire.
— Savoir importer une librairie locale (fichier .py) ou installée sur le système
— Savoir lire, comprendre et utiliser la documentation d'une librairie

4 Numpy : Ce qu'il faut retenir

— Pouvoir créer un tableau de type ndarray (fonctions `np.array`, `np.linspace` (avec l'option `retstep`), `np.arange`, `np.ones/np.zeros`, `np.ones_like/np.zeros_like`;
— Comprendre la notion de dimension, shape, size pour les ndarray et pouvoir y accéder;
— Savoir sélectionner un élément ou un groupe d'élément avec du slicing (syntaxe `start:stop:step` et dérivées, masking comme `arr[arr > a]`);
— Connaître et comprendre les règles de broadcasting.
— Savoir utiliser des opérations selon un axe ou un groupe d'axes (options `axis` pour toutes les fonctions numpy).
— Utilisation de la fonction `np.gradient` et `np.genfromtxt` avec leurs options courantes (espacement entre les points pour le gradient, `skip_header`, `delimiter`, `usecols` pour genfromtxt).
— Utilisation des fonctions mathématiques de base (sum, mean, median, log, sin, sqrt, tan, angle, abs, log, log10, tan, etc)
— Les fonctions `argmin` et `argmax` pour avoir les indices du maximum et du minimum.
— Modification des dimensions avec `expand_dims` et `squeeze`

— L'utilisation de boucles `for` doit se faire avec une extrême parcimonie, une utilisation abusive de boucles au lieu des options `axis` des fonctions numpy sera sanctionnée.

5 Matplotlib : Ce qu'il faut retenir

— Fonction `plt.plot`, `plt.show`, `set_title`, `set_xlabel`, `set_xlim` (et similaires sur y).
— Aucune option particulière de rendu n'est à retenir.

6 Scipy : Ce qu'il faut retenir

— Pour les fonctions qui agissent sur des fonctions (`solve_ivp`, `quad`, `brentq`, etc) : la possibilité d'utiliser l'option `args` pour fournir les paramètres fixés.
— L'utilisation des fonctions `integrate.solve_ivp` (équations différentielles) pour résoudre des équations différentielles du premier ou de second ordre.
— L'utilisation de la fonction `integrate.quad` pour faire de l'intégration ;
— L'utilisation de la fonction `optimize.brentq` pour faire de la recherche de zéro unidimensionnelle et `optimize.root` pour un problème à plusieurs dimensions ;
— L'utilisation de `optimize.minimize_scalar` et `optimize.minimize` pour trouver un minimum (à une dimension ou plusieurs dimensions respectivement)
— L'enchaînement `np.fft.fft`, `np.fft.fftfreq` pour faire une transformée de fourier.
— L'utilisation de `optimize.curve_fit` pour faire un ajustement de courbe.
Pour ces différentes fonctions, le prototype simplifié des fonctions sera donné en cas de besoin.

7 Contenu de l'examen

Il y aura à la fois des questions brèves, proches du cours et un ou plusieurs exercices plus complets qui nécessitent d'avoir recours à des fonctions et plusieurs portions de code pour être résolus.

7.1 L'examen portera sur :

— L'utilisation basique du terminal
— La connaissance des structures de base en python, de la forme attendue pour un script
— la production d'un code commenté et documenté. En particulier : un commentaire général en en-tête de fichier et des fonctions documentées avec le but de la fonction, les variables d'entrées (et leur type) ainsi que les valeurs de retour (et leur type)
— L'utilisation de tableaux multidimensionnels et les capacités de broadcasting, ainsi que les possibilités de calcul sur ceux-ci (fonctions mathématiques diverses)
— La lecture de fichiers de données
— L'utilisation d'une ou plusieurs fonctions complexes sous scipy/numpy (fonctions agissant sur une fonction)
— La capacité à produire un code complet, structuré, compartimenté, avec des fonctions

7.2 Ce qui sera pénalisé :

— Un code non documenté ni commenté
— Un script non structuré (section 4.6 du polycopié)
— Des noms de variables non explicites
— L'écriture de fonctions non documentées
— L'utilisation abusive de boucles `for` pour des opérations sur des tableaux numpy
— La non utilisation du slicing ou du masquage pour la sélection d'éléments d'un tableau numpy
— Utiliser des variables au sein d'une fonction qui ne sont déclarées comme argument de celle-ci

— La non-utilisation de fonctions vectorisées pour agir sur des tableaux
— Le non-respect de l'ordre des variables pour les fonctions complexes de scipy
— L'oubli du mot-clé `args` pour passer des paramètres à une fonction complexe de scipy
— Ne pas utiliser les options de formatage pour le texte.