BASES DE LA PROGRAMMATION

* Connaître la différence entre expression et instruction
* Connaître la convention pour les noms de variables et de fonctions, les espaces
* Connaître les instructions de base
* Connaître les différents usages de range
* Maîtriser la syntaxe (indentation et « : »)
* Maîtriser la syntaxe pour les importations de bibliothèques/modules
* Comprendre le fonctionnement des fonctions :
	+ ne surtout pas écrire de print à la place des return
	+ il peut y avoir 0, 1 ou plusieurs paramètres en entrée
	+ il peut y avoir 0, 1 ou plusieurs valeurs de sortie
* Savoir spécifier une fonction
	+ description de la finalité de la fonction
	+ description des arguments, type et (pré)conditions sur ceux-ci
	+ description des résultats, type et (post)conditions sur ceux-ci
* Connaître la différence entre interprétation et compilation

CODAGE ET TYPAGE

* Faire la différence entre coder (transformer des informations par un procédé connu et facilement réversible) et chiffrer (transformer des informations par une technique de cryptographie –code secret-, difficilement réversible, à l’aide d’une clé de chiffrement)
* Nombres entiers non signés :
	+ Maîtriser la conversion binaire vers décimal, et vice-versa
	+ Maîtriser la conversion binaire-hexadécimal et vice-versa
	+ Comprendre le fonctionnement général du passage décimal vers n’importe quelle base, et vice-versa, est un bonus non négligeable
* Nombres entiers signés :
	+ Maîtriser la méthode du complément à deux (et s’en rappeler le nom) pour la conversion décimal vers binaire et vice-versa
	+ Savoir que les erreurs d’arrondi, présentes sur toutes les machines, donnent des résultats mathématiquement faux ( en machine)

En conséquence, on ne teste pas l’égalité de deux flottants mais leur proximité.

* Nombres flottants :
	+ Maîtriser la méthode pour la conversion décimal vers binaire, et vice-versa, en 32 bits (IEEE 754 simple précision)
* Caractères :
	+ Connaître le nom et les différences entre ASCII, ISO 8859, les différentes possibilités utf-8, ut-16 et utf-32 d’Unicode.
	+ Connaître les fonctions ord(*caractère*) et chr(*nombre*) de Python.

le wwweb

* Faire la différence entre les langages HTML CSS JavaScript PHP
* HTML, CSS :
	+ Connaître la structure d’une page, le principe arborescent des balises (on ferme les balises dans l’ordre inverse de l’ouverture)
	+ HTML permet d’écrire le contenu de la page, CSS permet de mette en forme ce contenu
* Requêtes GET et POST
	+ Connaître la différence entre les requêtes GET et POST transmises par un formulaire en html, leurs avantages et inconvénients
* JavaScript :
	+ Étant donné les codes d’un script et d’une page html, pouvoir expliquer le fonctionnement et l’effet du script sur la page.
	+ Le JavaScript est utilisé (au lycée) pour modifier une page web sur le client
* PHP :
	+ Le PHP est utilisé pour construire une page web sur le serveur, qui sera ensuite envoyée au client.
	+ Le PHP est très utilisé pour gérer les bases de données

PROGRAMMATION : TYPES CONSTRUITS

* Listes (tableaux dynamiques)
	+ Connaître la fonction len(),la méthode tableau.sort()
	+ Savoir parcourir un tableau:
		- par indices

for i in range(len(tableau)) :

 *instruction avec* tableau[i]

* + - par valeur

for element in tableau :

 *instruction avec* element

* + Connaître par cœur les algorithmes :
		- calcul d’une moyenne d’un tableau
		- recherche du maximum dans un tableau
		- tableau ordonné ou non
		- appartenance d’un élément dans un tableau
		- nombre d’occurrences d’un élément dans un tableau
	+ Connaître les différentes façons de copier un tableau
	+ Quasi indispensable : compréhension des slices [a : b]
* Dictionnaires (tableaux associatifs)
	+ Savoir parcourir un dictionnaire par clé, par valeur, par item (keys(), values(), items())
	+ Instructions d’appartenance d’une clé, et renvoi éventuel de la valeur correspondante :
		- if cle in dico :
		- dico.get(cle)
		- dico[element] renvoie une erreur de cle si element n’est pas dans dico
	+ Connaître les algorithmes de recherche du maximum, de calcul de moyenne, de nombre d’occurrence d’une valeur
* Matrices
	+ maîtriser la notation matrice[i][j]
	+ parcourir une matrice par ligne, par colonne
* Tuples (p-uplets)
	+ notation
	+ un tuple n’est pas modifiable

Calcul booléen.

* Connaître les tables de vérité du ET, du OU, du NON, du XOR
* Connaître les symboles correspondants
* Plusieurs opérateurs logiques à la suite nécessitent des parenthèses, exemples  :
	+ NON (A OU B) ≠ (NON A) OU B
	+ A ET ( B OU C) ≠ (A ET B) OU C
	+ etc.
* Comprendre et savoir refaire le semi-additionneur à 1 bit
* Comprendre le principe de l’évaluation paresseuse :
	+ A et B est faux dès que A est faux, il est inutile de calculer B
	+ A ou B est vrai dès que A est vrai, il est inutile de calculer B

Algorithmique

* Algorithmes de tri
	+ Connaître l’algorithme du tri par sélection
	+ Connaître l’algorithme du tri par insertion
	+ Savoir compléter un programme implémentant le tri par sélection
	+ voir compléter un programme implémentant le tri par insertion
	+ au programme, difficile : donner l’invariant qui prouve la correction/justesse des tris par insertion et par sélection
* Recherche dichotomique
	+ Connaître l’algorithme de recherche dichotomique
	+ Savoir compléter un programme implémentant la recherche dichotomique
	+ au programme, difficile : variant de boucle permettant de prouver la terminaison de la recherche dichotomique
* Algorithmes gloutons
	+ Principe général :
		- parcours d’un tableau trié suivant un critère à déterminer
		- pas de remise en question des choix précédents
	+ Fonction sorted(tableau, key = cle), argument reverse = True/False facultatif
		- définition de la clé dans sorted par une fonction
* k plus proches voisins (*souvent repoussé en terminale*)
	+ Principe général
	+ Connaître l’algorithme, au moins son déroulement
	+ Savoir programmer le calcul de la distance euclidienne entre deux points dans un espace de dimension quelconque (2, 3, 10 ou 10000 coordonnées…)

INTERNET ET RESEAUX

* Connaître le modèle théorique TCP/IP à 4 couches
* Différencier l’architecture client-serveur de l’architecture peer to peer (pair à pair)
* Connaître l’usage de ces protocoles :
	+ http, https (communication client serveur), ftp (transfert de fichiers)
	+ smtp, pop, imap (protocoles mail)
	+ tcp, udp (protocoles de transmission de données par découpage en segments)
	+ dns (relie les IP aux url)
* Comprendre le protocole de détection d’erreur du bit alterné, ainsi que ses limites
* Différencier adresses MAC et adresses IP, savoir qu’IPv4 va disparaître progressivement au profit d’IPv6
* Maîtriser la forme des adresses IPv4
* Maîtriser la forme des masques IPv4
* Savoir trouver à partir d’une IPv4 et de son masque l’adresse de réseau, l’adresse de broadcast, la plage d’adresses machine et le nombre de celles-ci
* Savoir construire un réseau simple sous Filius, avec 3 sous-réseaux, un serveur DNS, un serveur web et un client

ARCHITECTURE DES ORDINATEURS

* Connaître quelques dates et personnalités fondamentales de l’histoire de l’informatique, y compris ce qui est vu dans le cous Internet.
* Retenir le modèle de Von Neumann

Entrée

Unité arithmétique

Sortie

Mémoire

Unité de commande

* Retenir le modèle de l’UAL
* Pouvoir comprendre/écrire un programme basique en assembleur simplifié

* Si le chapitre Arduino a été fait, savoir ce qu’est un capteur et un actionneur

INTRODUCTION A LINUX

* Comprendre ce que fait un système d’exploitation
* Comprendre ce que sont les permissions, connaître la commande Linux permettant de les modifier.
* Connaître par cœur les instructions Linux de base, y compris les options ls –a, ls -l et rm –r.

Introduction au traitement des donnees

* Comprendre le format csv
* Connaître le vocabulaire des bases de données
	+ table (= relation)
	+ enregistrement
	+ champ (= attribut, descripteur)
* Comprendre la manière dont un fichier csv est importé sous Python, ainsi que le format des données récupérées (liste de dictionnaires)
* Savoir :
	+ extraire les données d’une table suivant un ou plusieurs critères
	+ trier les données d’une table suivant un ou plusieurs critères (usage de la fonction sorted)
	+ fusionner deux ou plusieurs tables
	+ joindre deux tables comportant un champ commun