

- Noms : **Le nom des variables et des fonctions est sensible à la casse** (majuscules et minuscules sont traitées comme des lettres différentes) et ne doit pas contenir d'espace, de caractères spéciaux (sauf le sous-tiret '_') ou accentués.
- Commentaires : A partir du caractère '#' et jusqu'à la fin de la ligne, tous les caractères sont ignorés par l'interpréteur Python (le logiciel qui exécute les instructions du programme). Ceci sert à mettre des informations destinées uniquement aux lecteurs humains.

Les types de base :	Exemple
Entier : int	n = 25
Nombre à virgule (pseudo-réels ou nombres flottant) : float	x = 1.41
Chaîne de caractère (suite de lettres, chiffres ou symboles) : str	message = 'Coucou' <small>On peut aussi utiliser les double cotes (") ex : t = "test"</small>
Booléen : boolean variable logique qui ne peut valoir que True ou False	test = 't' in 'Galilée' <small>Teste si la lettre t est dans le mot 'Galilée'</small> ici test vaut False
On peut convertir un type en un autre quand c'est possible (ex : le texte '42' en nombre 42) en utilisant le nom du type comme instruction Ex : n = int ('156') ou texte = str (5.6 + 2.4)	

Les opérateurs de base :	Exemple
Opérations classique : + ; - ; * (la multiplication est notée *) ; /	n = 2 * 3 + 5 # n vaut 11
Mise à la puissance : **	c = 4**3 # c vaut 4 à la puissance 3 donc 64
Division entière : //	b = 21 // 4 # b vaut 5
Reste de la division (très utile pour les modulus ou pour la parité)	t = 21 % 4 # t vaut 1 (pas un multiple de 4)

Instruction	Exemple
Importer un module (pour pouvoir utiliser ses fonctions) import nom du module <small>A mettre toujours en début de fichier</small>	import math # fonctions usuelles de math import matplotlib.pyplot as plt
Afficher une variable print (variable)	h = 6.626e-34 print (h) <small>Affiche 6.626e-34</small>
Demander une information à l'utilisateur (stockée dans une chaîne) reponse = input (variable)	reponse = input ('prix au kilo ?') prix = float (reponse) <small>La réponse est une chaîne, on la converti en nombre</small>
Générer un nombre aléatoire entre a et b random.randint (a, b) <small>Il faut importer le module 'random' avant</small>	de_6 = random.randint (1, 6) print ('Le dé a fait : ', de 6)

Listes	
Les listes numérotées (type list) : Une liste est une structure permettant de stocker plusieurs valeurs de tous types en les rangeant selon leur numéro	t = [] # crée une liste vide t = [42, 'abc'] # crée une liste de deux valeurs (le nombre 42 et la chaîne 'abc') t = [0]*25 # crée une liste de 25 éléments ne contenant que des zéros
Ajouter un élément à la fin d'une liste : append ()	t. append (b) # ajout l'élément b à la fin de la liste t
Déterminer la taille de la liste : len ()	t = [4, 'kilo', 9, -1, 0] longueurListe = len (t) <small>longueurListe vaut 5</small>
Accéder à un élément de la liste par son numéro t[i] <small>i+1^{ème} élément de la liste t (les indices vont de 0 pour le 1^{er} terme à la taille de la liste - 1)</small>	t = ['x', 3.5, 42, 'Ampère'] print (t[2]) <small>Affiche 42</small> t[0] = 7 <small>t vaut maintenant [7, 3.5, 42, 'Ampère']</small>
Supprimer un élément de la liste : remove ()	t = [12, 5, -3, 'abc'] t. remove (-3) <small>t vaut maintenant [12, 5, 'abc'] et t[2] = 'abc'</small>
Supprimer un élément au rang i : pop ()	t = ['a', 2.1, 'b', 42, 'abc', 11, 'def'] t. pop (3) # On peut aussi utiliser del t[i] <small>t vaut maintenant ['a', 2.1, 'b', 'abc', 11, 'def'] et t[4] = 11</small>

Boucles	
Permet de répéter N fois un bloc de code for i in range(N) : → bloc	for i in range(N) : a = a + i print (i**3) <i>i va prendre toutes les valeurs entières de 0 à N-1</i>
Permet de parcourir une liste en examinant chaque éléments for element in liste : → total = total + element	liste = [15 , 3, 8, -4] total = 0 for element in liste : total = total + element print (total) <i>A la fin total vaut 22</i>
Répéter un bloc de code tant qu'une condition est remplie : while condition : <i>code de la boucle</i>	nb = random.randint(1,20) reponse = input ('Quel est le nombre ?') while nb != int(reponse) : reponse = input ('Essaye encore')

Les opérateurs logiques et de test :	Exemple
Test d'égalité : == Test de différence : !=	2 == (1+1) vaut True et 8 != (9-1) vaut False
Supérieur à : > supérieur ou égal : >= <i>idem pour inférieur</i>	7 < 3 vaut False et 24 >= (22+2) vaut True
Et logique : and a and b est vrai si a et b sont tous les deux vrais	(4 > 8) and (5 == (2+3)) vaut False (7 != 9) and (8 > 6) vaut True
Ou logique : or a or b est vrai si a ou b est vrais (ou qu'ils sont tous 2 vrais)	(4 > 8) or (5 == (2+3)) vaut True (7 == 9) or (8 <= 6) vaut False
Négation : not (permet d'inverser une condition) not a est <i>vrai</i> si a est faux et <i>faux</i> si a est vrai	not (2 == (1+1)) vaut False not (6 < 3) vaut True

Instructions conditionnelles
Exécuter un bloc d'instruction seulement si une condition est vraie : if condition : ... <i>instructions si condition est vraie</i> ... elif condition2 : # Sinon si (else if) ... <i>instructions si condition est faux et condition2 est vraie...</i> else : # Sinon ... <i>instructions si toutes les conditions sont fausses...</i>
if age <= 18 : → etat = 'Enfant' → if statut == 'Collégien' : → DNB = False → elif statut == 'Lycéen' : → DNB = True elif age > 65 : → etat = 'Retraité' else : → etat = 'Actif'

Fonctions
Définir une fonction = bout de programme qui prend (éventuellement) un ou plusieurs arguments en entrée et renvoie (éventuellement) un résultat : def nomFonction(arg1, arg2, ...) : ... Bloc d'instructions de la fonction ... return resultat NB : Une fonction doit toujours être définie avant de l'appeler. Du coup on place en général en début de code les définitions de fonction.
def refracte(n1, i1, n2) : sinus_i2 = (n1*math.sin(i1/180*math.pi))/n2 if sinus_i2 > 1 : print ('Reflexion totale') return None i2 = math.asin(sinus_i2) return i2*180/math.pi #Programme principal print ("L'angle de réfraction pour une incidence de 30° est : ", refracte(1,30,1.33))