COURS PI

L'école sur-mesure

de la Maternelle au Bac, Établissement d'enseignement privé à distance, déclaré auprès du Rectorat de Paris

Seconde - Module 5 - Algorithmique et programmation

Mathématiques

v.5.1

- ***Guide de méthodologie***
  pour appréhender notre pédagogie
- ***Leçons détaillées***
  pour apprendre les notions en jeu
- ***Exemples et illustrations***
  pour comprendre par soi-même
- ***Prolongement numérique***
  pour être acteur et aller + loin
- ***Exercices d'application***
  pour s'entraîner encore et encore
- ***Corrigés des exercices***
  pour vérifier ses acquis

www.cours-pi.com

Paris & Montpellier

EN ROUTE VERS LE BACCALAURÉAT

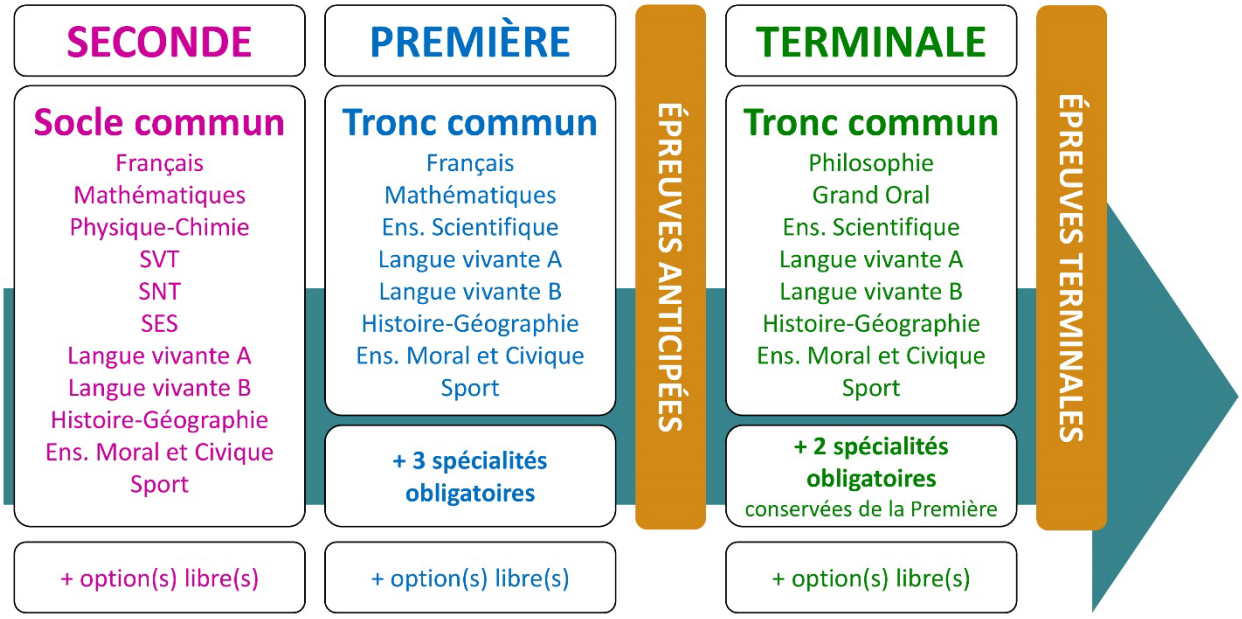
Comme vous le savez, **la réforme du Baccalauréat** est entrée en vigueur progressivement jusqu'à l'année 2021, date de délivrance des premiers diplômes de la nouvelle formule.

Dans le cadre de ce nouveau Baccalauréat, **notre Etablissement**, toujours attentif aux conséquences des réformes pour les élèves, s'est emparé de la question avec force **énergie** et **conviction** pendant plusieurs mois, animé par le souci constant de la réussite de nos lycéens dans leurs apprentissages d'une part, et par la **pérennité** de leur parcours d'autre part. Notre Etablissement a questionné la réforme, mobilisé l'ensemble de son atelier pédagogique, et déployé tout **son savoir-faire** afin de vous proposer un enseignement tourné continuellement vers l'**excellence**, ainsi qu'une scolarité tournée vers la **réussite**.

- Les **Cours Pi** s'engagent pour faire du parcours de chacun de ses élèves un **tremplin vers l'avenir**.
- Les **Cours Pi** s'engagent pour ne pas faire de ce nouveau Bac un diplôme au rabais.
- Les **Cours Pi** vous offrent **écoute** et **conseil** pour coconstruire une **scolarité sur-mesure**.

LE BAC DANS LES GRANDES LIGNES

Ce nouveau Lycée, c'est un enseignement à la carte organisé à partir d'un large tronc commun en classe de Seconde et évoluant vers un parcours des plus spécialisés année après année.



CE QUI A CHANGÉ

- Il n'y a plus de séries à proprement parler.
- Les élèves choisissent des spécialités : trois disciplines en classe de Première ; puis n'en conservent que deux en Terminale.
- Une nouvelle épreuve en fin de Terminale : le Grand Oral.
- Pour les lycéens en présentiel l'examen est un mix de contrôle continu et d'examen final laissant envisager un diplôme à plusieurs vitesses.
- Pour nos élèves, qui passeront les épreuves sur table, le Baccalauréat conserve sa valeur.

CE QUI N'A PAS CHANGÉ

- Le Bac reste un examen accessible aux candidats libres avec examen final.
- Le système actuel de mentions est maintenu.
- Les épreuves anticipées de français, écrit et oral, tout comme celle de spécialité abandonnée se dérouleront comme aujourd'hui en fin de Première.

A l'occasion de la réforme du Lycée, nos manuels ont été retravaillés dans notre atelier pédagogique pour un accompagnement optimal à la compréhension. Sur la base des programmes officiels, nous avons choisi de créer de nombreuses rubriques :

- **Suggestions de lecture** pour s'ouvrir à la découverte de livres de choix sur la matière ou le sujet.
- **Réfléchissons ensemble** pour guider l'élève dans la réflexion.
- **L'essentiel** et **Le temps du bilan** pour souligner les points de cours à mémoriser au cours de l'année
- **À vous de jouer** pour mettre en pratique le raisonnement vu dans le cours et s'accaparer les ressorts de l'analyse, de la logique, de l'argumentation, et de la justification.
- **Pour aller plus loin** pour visionner des sites ou des documentaires ludiques de qualité.
- Et enfin ... la rubrique **Les Clés du Bac by Cours Pi** qui vise à vous donner, et ce dès la seconde, toutes les cartes pour réussir votre examen : notions essentielles, méthodologie pas à pas, exercices types et fiches étape de résolution !

MATHÉMATIQUES SECONDE

Module 5 – Algorithmique et programmation

L'AUTEURE

Sylvie LAMY

« Faire des maths c'est jouer aux legos. Il s'agit d'assembler des briques pour solutionner des problèmes ». Diplômée de l'Ecole Polytechnique et agrégée de Mathématiques, elle poursuit aujourd'hui son parcours professionnel à l'Institut Géographique National et au Ministère des Transports comme chargée de mission sur les projets spatiaux. Passionnée par les sciences physiques, son approche pédagogique réside dans la transmission du raisonnement scientifique. Elle attend de ses élèves de comprendre et d'expliciter leur démarche dans la résolution des problèmes.

PRÉSENTATION

Ce **cours** est divisé en chapitres, chacun comprenant :

- Le **cours**, conforme aux programmes de l'Education Nationale
- Des **exercices d'application et d'entraînement**
- Les **corrigés** de ces exercices
- Des **devoirs** soumis à correction (et ***se trouvant hors manuel***). Votre professeur vous renverra le corrigé-type de chaque devoir après correction de ce dernier.

Pour une manipulation plus facile, les corrigés-types des exercices d'application et d'entraînement sont regroupés en fin de manuel.

CONSEILS A L'ÉLÈVE

Vous disposez d'un support de Cours complet : **prenez le temps** de bien le lire, de le comprendre mais surtout de l'**assimiler**. Vous disposez pour cela d'exemples donnés dans le cours et d'exercices types corrigés.
Vous pouvez rester un peu plus longtemps sur une unité mais travaillez régulièrement.

LES FOURNITURES

Vous devez posséder :

- une **calculatrice graphique pour l'enseignement scientifique au Lycée comportant un mode examen (requis pour l'épreuve du baccalauréat).**
- un **tableur** comme Excel de Microsoft (payant) ou Calc d'Open Office (gratuit et à télécharger sur http://fr.openoffice.org/). En effet, certains exercices seront faits de préférence en utilisant un de ces logiciels, mais vous pourrez également utiliser la calculatrice).

LES DEVOIRS

Les devoirs constituent le moyen d'évaluer l'acquisition de vos *savoirs* (« Ai-je assimilé les notions correspondantes ? ») et de vos *savoir-faire* (« Est-ce que je sais expliquer, justifier, conclure ? »).

Placés à des endroits clés des apprentissages, ils permettent la vérification de la bonne assimilation des enseignements.

Aux *Cours Pi*, vous serez accompagnés par un professeur selon chaque matière tout au long de votre année d'étude. Référez-vous à votre « Carnet de Route » pour l'identifier et découvrir son parcours.
Avant de vous lancer dans un devoir, assurez-vous d'avoir **bien compris les consignes**.
Si vous repérez des difficultés lors de sa réalisation, n'hésitez pas à le mettre de côté et à revenir sur les leçons posant problème. **Le devoir n'est pas un examen**, il a pour objectif de s'assurer que, même quelques jours ou semaines après son étude, une notion est toujours comprise.
Aux *Cours Pi*, chaque élève travaille à son rythme, parce que chaque élève est différent et que ce mode d'enseignement permet le « sur-mesure ».
Nous vous engageons à respecter le moment indiqué pour faire les devoirs. Vous les identifierez par le bandeau suivant :

Vous pouvez maintenant faire et envoyer le devoir n°1

Il est **important de tenir compte des remarques, appréciations et conseils du professeur-correcteur**. Pour cela, il est **très important d'envoyer les devoirs au fur et à mesure** et non groupés. **C'est ainsi que vous progresserez !**

Donc, dès qu'un devoir est rédigé, envoyez-le aux *Cours Pi* par le biais que vous avez choisi :

1) Par **soumission en ligne** via votre espace personnel sur **PoulPi**, pour un envoi **gratuit**, **sécurisé** et plus **rapide**.
2) Par **voie postale** à *Cours Pi*, 9 rue Rebuffy, 34 000 Montpellier
   *Vous prendrez alors soin de joindre une **grande enveloppe libellée à vos nom et adresse**, et **affranchie au tarif en vigueur** pour qu'il vous soit retourné par votre professeur.*

***N.B. :** quel que soit le mode d'envoi choisi, vous veillerez à **toujours joindre l'énoncé du devoir ;** plusieurs énoncés étant disponibles pour le même devoir.*
***N.B. :** si vous avez opté pour un envoi par voie postale et que vous avez à disposition un scanner, nous vous engageons à conserver une copie numérique du devoir envoyé. Les pertes de courrier par la Poste française sont très rares, mais sont toujours source de grand mécontentement pour l'élève voulant constater les fruits de son travail.*

SOUTIEN ET DISPONIBILITÉ

VOTRE RESPONSABLE PÉDAGOGIQUE

Professeur des écoles, professeur de français, professeur de maths, professeur de langues : notre Direction Pédagogique est constituée de spécialistes capables de dissiper toute incompréhension.

Au-delà de cet accompagnement ponctuel, notre Etablissement a positionné ses Responsables pédagogiques comme des « super profs » capables de co-construire avec vous une scolarité sur-mesure.
En somme, le Responsable pédagogique est votre premier point de contact identifié, à même de vous guider et de répondre à vos différents questionnements.

Votre Responsable pédagogique est la personne en charge du suivi de la scolarité des élèves.
Il est tout naturellement votre premier référent : une question, un doute, une incompréhension ? Votre Responsable pédagogique est là pour vous écouter et vous orienter. Autant que nécessaire et sans aucun surcoût.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.

QUEL EST SON RÔLE ?

Orienter les parents et les élèves.
Proposer la mise en place d'un accompagnement individualisé de l'élève.
Faire évoluer les outils pédagogiques.
Encadrer et coordonner les différents professeurs.

VOS PROFESSEURS CORRECTEURS

Notre Etablissement a choisi de s'entourer de professeurs diplômés et expérimentés, parce qu'eux seuls ont une parfaite connaissance de ce qu'est un élève et parce qu'eux seuls maîtrisent les attendus de leur discipline. En lien direct avec votre Responsable pédagogique, ils prendront en compte les spécificités de l'élève dans leur correction. Volontairement bienveillants, leur correction sera néanmoins juste, pour mieux progresser.

QUAND PUIS-JE LE JOINDRE ?

Une question sur sa correction ?

- faites un mail ou téléphonez à votre correcteur et demandez-lui d'être recontacté en lui laissant un message avec votre nom, celui de votre enfant et votre numéro.
- autrement pour une réponse en temps réel, appelez votre Responsable pédagogique.

LE BUREAU DE LA SCOLARITÉ

Placé sous la direction d'Elena COZZANI, le Bureau de la Scolarité vous orientera et vous guidera dans vos démarches administratives. En connaissance parfaite du fonctionnement de l'Etablissement, ces référents administratifs sauront solutionner vos problématiques et, au besoin, vous rediriger vers le bon interlocuteur.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.
04.67.34.03.00
scolarite@cours-pi.com

LE SOMMAIRE

OBJECTIFS

- Manipuler des séquences d'instructions.
- Manipuler les variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- Programmer, dans des cas simples, une boucle bornée, une boucle non bornée.
- Dans des cas plus complexes : lire, comprendre, modifier ou compléter un algorithme ou un programme.

COMPÉTENCES VISÉES

- Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- Lire et comprendre une fonction renvoyant une moyenne, un écart type. Aucune connaissance sur les listes n'est exigée.
- Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.
- Choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères).
- Concevoir et écrire une instruction d'affectation, une séquence d'instructions, une instruction conditionnelle.
- Écrire une formule permettant un calcul combinant des variables.

OBJECTIFS

- Manipuler des séquences d'instructions.
- Manipuler les variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- Programmer, dans des cas simples, une boucle bornée, une boucle non bornée.
- Dans des cas plus complexes : lire, comprendre, modifier ou compléter un algorithme ou un programme.

COMPÉTENCES VISÉES

- Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- Lire et comprendre une fonction renvoyant une moyenne, un écart type. Aucune connaissance sur les listes n'est exigée.
- Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.
- Choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères).
- Concevoir et écrire une instruction d'affectation, une séquence d'instructions, une instruction conditionnelle.
- Écrire une formule permettant un calcul combinant des variables.

SUGGESTIONS CULTURELLES

ESSAIS

- **Les maths c'est magique !** *Johnny Ball*
- **17 équations qui ont changé le monde** *Ian Stewart*
- **Alex au pays des chiffres** *Alex Bellos*
- **Le grand roman des maths :de la préhistoire à nos jours** *Mickael Launay*
- **La symphonie des nombres premiers** *Marcus du Sautoy*
- **Dans la jungle des nombres premiers.** *John Derbyshire*
- **Histoire universelle des chiffres : l'intelligence des hommes racontée par les nombres et le calcul** *Georges Ifrah*
- **Le démon des maths.** *Hans Magnus Enzensberger*
- **A propos de rien : une histoire du zéro** *Robert kaplan*

BANDES-DESSINÉES

- **Logicomix** *Doxiádis / Papadátos / Papadimitríou*
- **Les maths en BD 1 et 2** *Larry Gonick*
- **Les statistiques en BD** *Larry Gonick*

DOCUMENTAIRES AUDIOVISUELS

- **L'extraordinaire aventure du chiffre 1** *Terry Jones*
- **Le mystère des nombres premiers** *Marcus du Sautoy*

SITES INTERNET RESSOURCES

- **pyvideo.org**
- **planetpython.org**
- **dimensions-math.org**


Le Cours

INTRODUCTION

L'algorithme… un « terme » récurrent en cours de mathématique depuis le collège… Un terme courant dans la vie de tous les jours puisqu'on le retrouve dans les calculatrices, les calculateurs, l'informatique et même Parcoursup…

Le terme « algorithme » est issu d'un illustre mathématicien persan Al Khwarizmi, qui introduisit vers 820 en Occident la numération décimale (rapportée d'Inde) et enseigna les règles élémentaires des calculs s'y rapportant.

L'algorithme provient donc de la manipulation des chiffres mais s'est développé progressivement sous la notion de « méthode ». Cette « méthode » ou ce « process » permet de répondre à des questions comme « comment obtenir cette valeur ? », « calculer cela ? », « trouver telle information ? », « calculer tel nombre ? ».

Le concept développé pour un algorithme revient à répondre à une procédure et se caractérise par une application qui se fait mécaniquement.

OBJECTIFS

- Consolidation des acquis du cycle 4 autour de deux idées essentielles :
  - La notion de fonction
  - La programmation comme production d'un texte dans un langage informatique

COMPÉTENCES VISÉES

- Décrire des algorithmes en langage naturel ou dans un langage de programmation.
- En réaliser quelques-uns à l'aide d'un programme simple écrit dans un langage de programmation textuel.
- Interpréter, compléter ou modifier des algorithmes plus complexes.

PRÉSENTATION

PYTHON

On utilise en informatique de nombreux langages de programmation différents, chacun disposant de ses avantages et inconvénients.

Python est l'un d'entre eux, créé en 1991 et toujours populaire aujourd'hui grâce à sa polyvalence et sa simplicité.

Il est d'utilisation gratuite, placé sous licence libre et fonctionne sur presque toutes les machines d'aujourd'hui (PC sous Windows/Mac/Linux, ou smartphones, etc.).

EDUPYTHON

Edupython est une initiative française, lancée par des professeurs désirant faciliter l'utilisation du langage python lors de leurs cours.

L'installation du python standard est déjà simple. L'installation d'Edupython simplifie encore un peu plus les choses en pré-installant des modules utiles en cours (fonctions mathématiques ou musicales, par exemple) ainsi qu'un éditeur de texte (PyScripteur).

L'installateur principal est prévu pour être utilisé sous le système d'exploitation Windows (et c'est sur cette version que se base le présent document), mais il est possible d'installer EduPython sur Mac ou Linux (voir la section FAQ du site EduPython).
Le site d'EduPython : https://edupython.tuxfamily.org/

INSTALLATION

Rendez-vous sur le site d'EduPython et cliquez sur l'onglet « Téléchargement ».

Téléchargez la dernière version du package (à ce jour il s'agit de la 2.6). Attention : pour télécharger l'exécutable (solution la plus courante) ou le zip (si vous préférez) il faut cliquer sur les icônes, placées en plein milieu du texte.

Profitez-en pour récupérer aussi les différentes « fiches pour élèves » proposées plus bas concernant :

- les bases ;
- les chaînes de caractères ;
- la tortue ;
- graphisme ;
- les statistiques et probabilités.

Il s'agit à chaque fois d'une page rappelant l'essentiel de la syntaxe et des commandes à utiliser pour une notion donnée (la tortue est un programme permettant de dessiner à l'écran à l'aise de directives mathématiques).

Installation de l'exécutable (solution par défaut)

Si vous avez récupéré le fichier « Setup_EP26.exe » (ou une version plus récente), il vous suffit de cliquer dessus une fois le téléchargement terminé. Le programme va installer les outils, et vous aurez dans votre menu Démarrer un nouveau lien « Edupython » permettant de lancer l'éditeur « PyScripter » à partir duquel vous allez travailler.

Installation du zip

Ne récupérez la version zip que si vous savez organiser de manière personnalisée l'espace occupé par vos logiciels sur vos disques durs. Si c'est le cas, vous pourrez ainsi installer Edupython où vous le désirez sans créer automatiquement de lien dans le menu Démarrer ou d'entrée dans la base de registre.

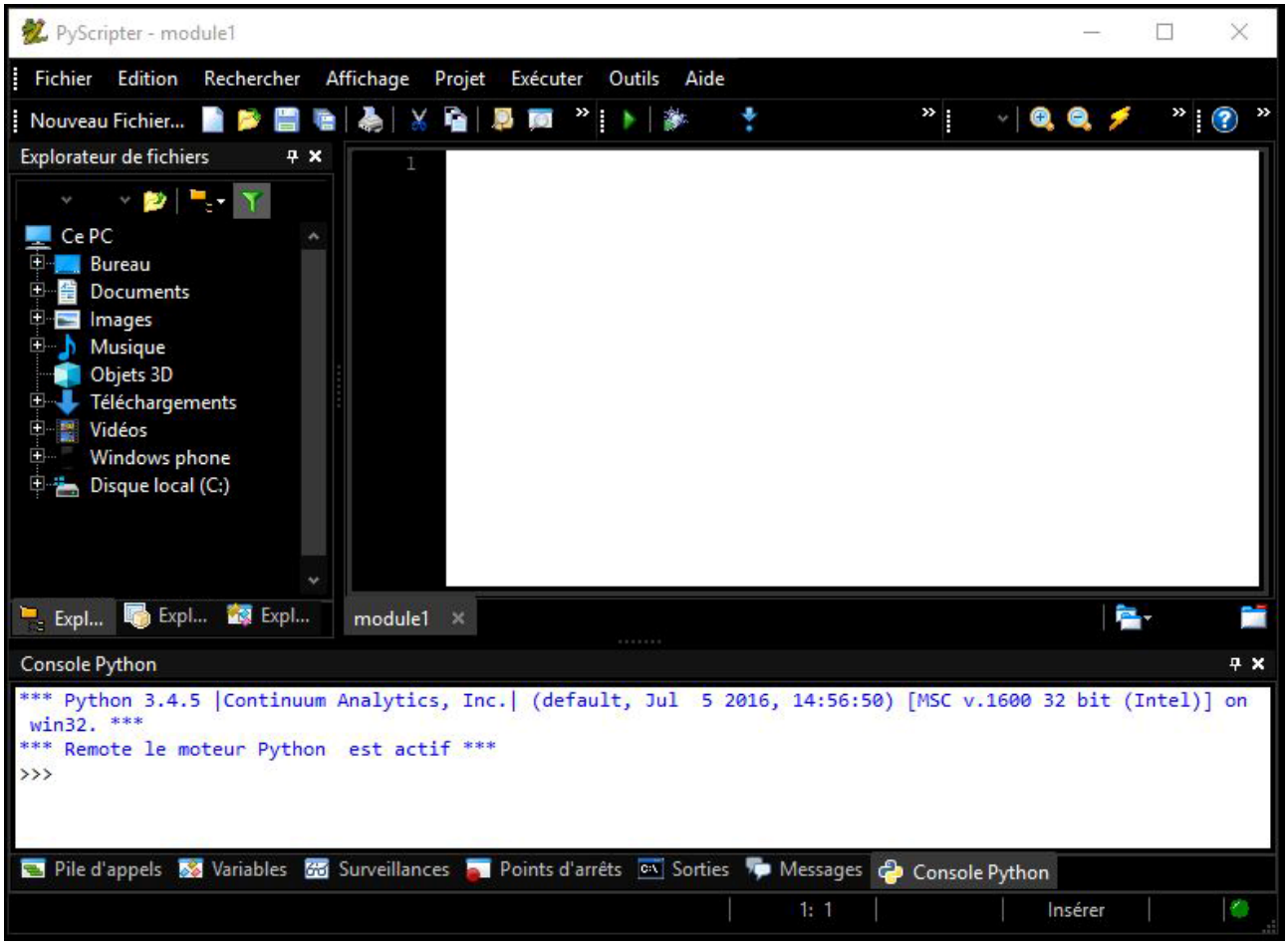
PREMIÈRES UTILISATIONS

SALUT LE MONDE !

Il est de coutume, dans les didacticiels destinés à l'apprentissage de langages informatiques, de commencer par l'action la plus simple qui soit : afficher un bout de texte. Et en général (ce n'est qu'une convention), le premier texte proposé est « Hello World! ».
C'est ce que nous allons faire (en français :-)).

Commençons par lancer PyScripter !
Pour cela, cherchez le lien Edupython dans votre menu démarrer (les programmes sont classés par ordre alphabétique, mais vous ne le voyez pas, vous pouvez commencer à taper le nom « Edupy... » au clavier et le lien devrait être mis en évidence).

L'affiche peut varier suivant l'OS (système d'exploitation) utilisé, mais le programme qui se lance devrait ressembler à ceci :



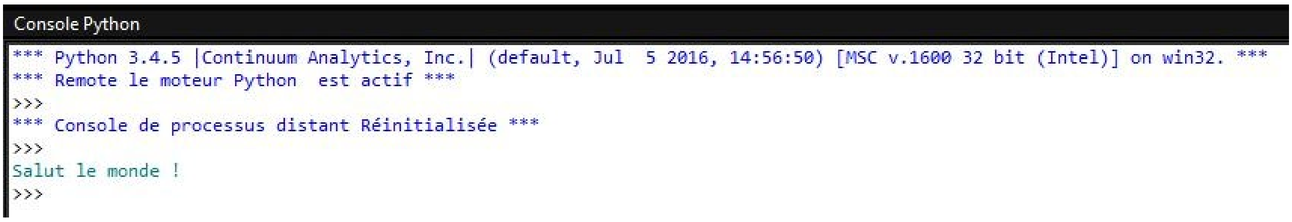
On notera que les menus (Fichier, Edition, etc.) sont en français ! Vous n'aurez pas besoin de toutes les fonctions, bien sûr, et les menus principaux sont assez standards et faciles à comprendre. Nous allons nous attarder principalement sur le menu « Exécuter » qui est plus inhabituel et qui va nous être utile. Pour ça, écrivons notre programme !

Dans la grande fenêtre d'édition sur fond blanc, tapez simplement la commande suivante :

```
print("Salut le monde !")
```

C'est tout !
Ensuite, dans le menu « Exécuter » descendez jusqu'à la ligne « Exécuter » et observez le résultat (dans la fenêtre « Console Python » en bas de l'écran).

Le résultat devrait ressembler à quelque chose comme ça (certaines lignes peuvent être cachées si la fenêtre est trop petite, mais on peut l'agrandir ou scroller dedans pour remonter) :



On voit trois phrases en bleu :
- « *** Python 3.4.5 |Continuum Analytics, Inc.| (default, Jul 5 2016, 14:56:50) [MSC v.1600 32 bit (Intel)] on win32. *** »
- « *** Remote le moteur Python est actif *** »
- « *** Console de processus distant Réinitialisée *** »

Il s'agit d'indications envoyées par python dont nous n'avons pas particulièrement besoin pour l'instant. Et ensuite En vert, on peut lire :
« Salut le monde ! »

C'est le texte que nous avons demandé à notre programme d'imprimer. En effet, la commande « print » sert à cela : afficher des caractères (textes, chiffres, etc.).

Raccourcis : vous avez peut-être remarqué dans le menu « Exécuter » que la ligne « Exécuter » profitait d'un raccourci clavier : Ctrl+F9.
Vous pouvez en effet aller plus vite ainsi, ou bien en utilisant le bouton de lecture (triangle vert) affiché dans la barre d'outils de la fenêtre principale.

LES FONCTIONS MATHÉMATIQUES D'EDUPYTHON

Nous l'avons évoqué plus haut, EduPython contient des fonctions mathématiques pré-intégrées. Python sait très bien travailler sur ce genre de fonctions, et on peut le paramétrer de manière très précise suivant le domaine étudié. Mais ici, EduPython propose tout simplement un ensemble de fonctions adapté au programme du lycée.

Ces fonctions sont regroupées dans une « bibliothèque » (un ensemble de fonctions) nommée « lycee » (sans accent).

Si l'on veut s'en servir, il faut l'indiquer à python en début de programme, à l'aide de la ligne suivante :

```
from lycee import *
```

Ce qui signifie : « va voir ce qui se trouve dans la librairie lycée, et récupère tout ce que tu y trouves ». En effet, suivant le contexte, l'astérisque, en informatique, peut signifier la multiplication ou bien constituer une sorte de « joker » qui ici signifie « tout ».

Exemple : calcul de puissance
Testons ça tout de suite en réalisant un mini-programme qui utilise la fonction puissance de la librairie lycée.
La fonction puissance est définie de cette façon dans lycée :

```
def puissance(a,n):
  """
  a,n sont des nombres
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Cette fonction renvoie le resultat de a^n
  """
  return a**n
```

Vous n'avez pas besoin de tout comprendre pour l'instant, notez juste que la forme qui doit être utilisée pour appeler cette fonction est : puissance (a,n).

On voit ici (comme on l'avait vu pour print), que les paramètres qu'on donne à une fonction (pour lui permettre de savoir ce qu'elle doit faire) sont à saisir entre des parenthèses.

Si l'on veut calculer 2 puissance 3, comment va-t-on faire ?
Écrivez le programme suivant :

```
from lycee import *

print(puissance(2,3))
```

Observez le résultat dans la console, vous devriez y lire que le module lycee a bien été activée, puis y voir le résultat de l'opération mathématique.

Et voilà pour vos premiers pas dans EduPython !

POUR ALLER PLUS LOIN

OBTENIR DE L'AIDE PERSONNALISÉE

Bien sûr, vous pouvez vous adresser à vos professeurs en cas de problème rencontré durant la réalisation des exercices.

Sachez toutefois qu'en cas de besoin, il existe un forum consacré à EduPython au sein duquel les créateurs et utilisateurs réguliers répondent aux questions qu'on leur pose :
https://edupython.tuxfamily.org/forum/index.php

D'une manière plus générale, de nombreuses communautés en ligne permettent d'obtenir rapidement de l'aide sur des questions de programmation.
En langue française, on peut citer des sites comme developpez.com, et en langue anglaise reddit.com ou stackoverflow.com !

EN SAVOIR PLUS SUR PYTHON

Python est un langage très populaire. On peut trouver à son sujet de nombreuses ressources en ligne gratuitement, que ce soit sous forme de vidéos, d'articles ou de livres électroniques à télécharger.

METTRE EN PRATIQUE DE MANIÈRE LUDIQUE

On peut utiliser le langage python pour réaliser des jeux vidéo mêlant graphisme et textes, à l'aide par exemple du programme (gratuit et open source) nommé Ren'py. Ce « moteur » permet de créer facilement des jeux du type « Visual Novel ».

Site principal : https://www.renpy.org/

Site de la communauté francophone : http://fr.renpy.org/

CHAPITRE 1
DÉMARRER AVEC PYTHON

Développé depuis 1989, Python est un langage portable (Unix/Linux, Mac OS X, Windows...), libre et gratuit. Il permet de développer de façon modulaire et orientée objet des applications de toutes tailles, notamment pour le Web, la plus connue d'entre elles étant le gestionnaire de contenu Zope. Python : de la syntaxe à l'optimisation. Python est tout indiqué pour le développement d'applications web : serveurs de contenu, moteurs de recherche, agents intelligents, objets distribués... Il est également performant pour réaliser des scripts d'administration système ou d'analyse de fichiers textuels, pour gérer l'accès à des bases de données, pour servir de langage glu entre plusieurs applications, réaliser des applications graphiques classiques, etc.

Ce chapitre est un chapitre de prise en main de Python de l'installation aux premiers pas de programmation.

OBJECTIFS

- Manipuler des séquences d'instructions.
- Manipuler les variables informatiques de type entier, booléen, flottant, chaîne de caractères.
- Programmer, dans des cas simples, une boucle bornée, une boucle non bornée.
- Dans des cas plus complexes : lire, comprendre, modifier ou compléter un algorithme ou un programme.

COMPÉTENCES VISÉES

- Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- Lire et comprendre une fonction renvoyant une moyenne, un écart type. Aucune connaissance sur les listes n'est exigée.
- Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.
- Choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères).
- Concevoir et écrire une instruction d'affectation, une séquence d'instructions, une instruction conditionnelle.
- Écrire une formule permettant un calcul combinant des variables.

PRÉ-REQUIS

- Notions de programmation sur Scratch vues au Collège.

DÉMARRER AVEC PYTHON

Installation

UN PEU D'HISTOIRE

À la fin des années 1980, le programmeur Guido van Rossum participe au développement du langage de programmation ABC au Centrum voor Wiskunde en Informatica (CWI) d'Amsterdam, aux Pays-Bas. Il travaillait alors dans l'équipe du système d'exploitation Amoeba dont les appels systèmes étaient difficilement interfaçables avec le Bourne shell utilisé comme interface utilisateur. Il estime alors qu'un langage de script inspiré d'ABC pourrait être intéressant comme interpréteur de commandes pour Amoeba.

En 1989, profitant d'une semaine de vacances durant les fêtes de Noël, il utilise son ordinateur personnel pour écrire la première version du langage. Fan de la série télévisée Monty Python's Flying Circus, il décide de baptiser ce projet Python.

Guido van Rossum est le principal auteur de Python, et son rôle de décideur central permanent de Python est reconnu avec humour par le titre de Benevolent Dictator for Life, BDFL (« Dictateur bienveillant à vie »).

Il est assisté d'une équipe de core developers qui ont un accès en écriture au dépôt de CPython et qui se coordonnent sur la liste de diffusion python-dev. Ils travaillent principalement sur le langage et la bibliothèque de base. Ils reçoivent ponctuellement les contributions d'autres développeurs Python via la plateforme de gestion de bug Roundup, qui a remplacé SourceForge.

Les allusions aux Monty Python sont assez fréquentes. Les didacticiels consacrés à Python utilisent souvent les mots spam et eggs comme variable métasyntaxique. Il s'agit d'une référence au sketch Spam des Monty Python, où deux clients tentent de commander un repas à l'aide d'une carte qui contient du jambon en conserve de marque SPAM dans pratiquement tous les plats. Ce sketch a été aussi pris pour référence pour désigner un courriel non sollicité.

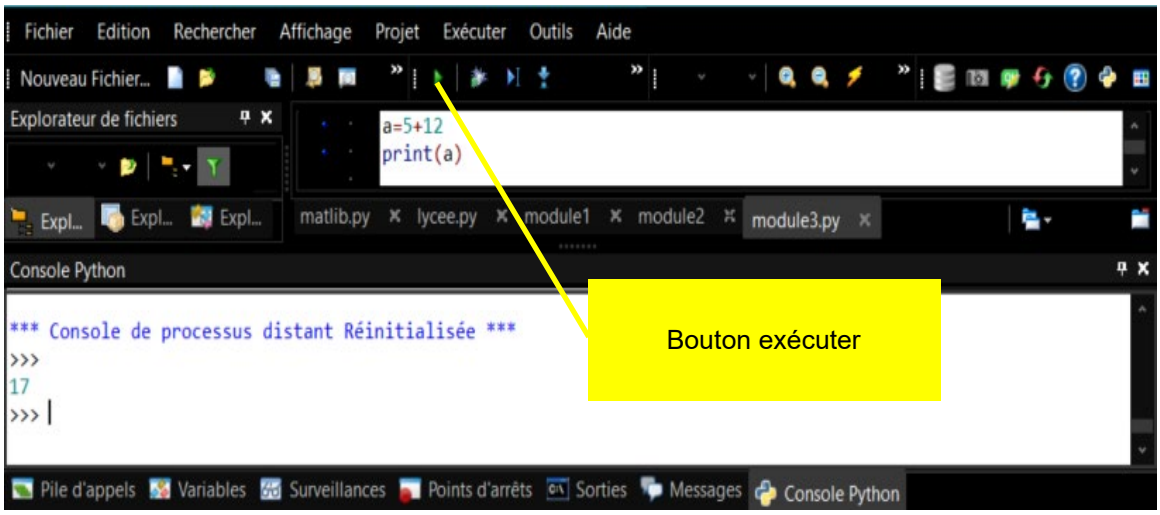
LES PRINCIPALES OPÉRATIONS

Exemple :

Dans la fenêtre principale de PyScripter taper :

```
a=5+12
print(a)
```

Puis exécuter le programme (flèche verte).



Le résultat 17 doit s'afficher dans la console Python.
Voici les différentes opérations qu'on peut faire sur des nombres

opérations	
addition	**+**
soustraction	**-**
multiplication	*****
puissance	****
quotient décimal	**/**
quotient entier	**//**
reste de la division euclidienne	**%**
Arrondir un nombre a avec n décimales.	**round(a,n)**

Les règles de priorité sont respectées.

Exemple

5**2−1=24

16%3=1

18//4+7=11

18/4+7=11,5

Pour afficher le résultat on a utilisé la fonction **print().**

À VOUS DE JOUER 1

Modifiez le programme.

```
a=5+12
print(a)
```

A. pour calculer $\frac{3\times5-6^2}{4}$: a=.....

B. pour déterminer le reste de la division euclidienne 1586÷18 : a=.....

C. pour déterminer le quotient de la division décimale 1586/18 , arrondi au millième :a=.....

On sépare 2 instructions en passant à la ligne.
On peut également les séparer par un point-virgule ; .

VARIABLES

- **Variables et affectations**

L'ESSENTIEL

Une **variable** est un emplacement donné sur une machine. Elle porte un nom. Lorsqu'on remplit cet emplacement par une valeur donnée, on parle **d'affectation** de la variable.

La valeur prise par une variable peut donc changer donc au cours du programme.
Le symbole d'affectation qui sera utilisé dans la suite est la flèche ←.

Exemple : on veut écrire un algorithme qui calcule 5^8 .

$N \leftarrow 5$	On affecte 5 à la variable *N*.	*N* contient 5.
$N \leftarrow N^4$	Le programme va calculer le contenu de la variable à la puissance 8, puis affecter le résultat à la variable *N*.	*N* contient 125.
Afficher(*N*)	La valeur 125 s'affiche.	

Traduction en Python.

Affectation A←B	A=B

Rappel : pour exécuter le programme dans Pyscripter, cliquer sur la flèche verte. Le résultat s'affiche dans la fenêtre du bas « Console Python ».

À VOUS DE JOUER 2

1. Complétez

$X \leftarrow 4$	X contient .....
$Y \leftarrow 3$	Y contient .....
$Y \leftarrow X^Y$	Y contient .....
$Affiche r(Y)$	

2. Ecrivez ce programme sous Python en incluant les commentaires.

- **Types de variables**

Toutes les variables possèdent un **type**. La manipulation d'une variable dépend de ce type. Par exemple, on ne peut pas calculer le carré d'un texte !
Voilà les 4 types principaux (on en verra d'autre par la suite) :

Types de variable	
nombre entier relatif	**int**
nombre décimal (flottants)	**float**
chaine de caractères (string)	**str**
booléen	**bool**

Exemples :
8 et -6 sont de type int.
5.3 est de type float.
'Chaise' est de type str
"L'arbre" est de type str
1 mais n'est pas divisible par 9.

Remarques
- En informatique, on ne fait pas la différence entre décimaux et réels non décimaux. Tous les nombres sont calculés avec des arrondis, comme sur une calculatrice donc sont des décimaux !
- La virgule est remplacée par un point.
- Les chaines de caractères se mettent entre guillemets simples ('...') ou entre guillemets ("..."). Il est préférable d'utiliser "..." car cela permet d'avoir des chaines avec apostrophe comme "L'arbre".

Le type booléen :
Une variable de type bool ne prend que 2 valeurs : True et False.
Nous y reviendrons dans le chapitre suivant.

À VOUS DE JOUER 3

Déterminez le type de ces variables.

$a=-4$ .....
b='Coucou' .....
$c=5.05$ .....
d="4" .....
e=True .....

Contrairement à d'autres langages de programmation, Python affecte en dynamique son type à une variable.

Exemple :

```
X=4   #X sera de type int
Y=1/X   #Y sera de type float
```

À VOUS DE JOUER 4

Déterminez le type de ces variables.

$a=3**2$ .....
$b=$"*Coucou*" .....
$c=3/2$ .....
$d=16\%3$ .....
e=False .....
$c=3//2$ .....

Pour connaître le type d'une variable : type(…)

Exemple :

```
print(type('coucou'))
X=5.6
print(type(X))
```

Ce script donnera :
<class 'str'>
<class 'float'>

- **Changer le type d'une variable**

Exemple préliminaire :

Reprenons l'exemple précédent

$N \leftarrow 5$
$N \leftarrow N^4$
Afficher(N)

qui a permis de calculer 5^4.

On veut compléter pour que la puissance 4 d'un nombre saisi par l'utilisateur s'affiche.

$N \leftarrow Saisir()$
$N \leftarrow N^4$
$Afficher(N)$

Traduction en Python.

La saisie se fait avec **input ()**. Le programme ouvre une fenêtre dans laquelle se fait la saisie.
Voilà ce qu'on est tenté d'écrire :

```
N=input()
N=N**4
print(N)
```

Si on exécute ce programme et qu'on entre 5, il programme génère une erreur !

```
N=input()
N=N**4
print(N)
```

Voici l'explication :
Le résultat de « input() » est toujours considéré comme une chaine de caractères (type str) et non un nombre. La puissance d'un texte ne peut pas être calculée!
On va convertir le résultat de input() en entier avec la fonction int().

```
N=input() #N est une chaine de caractères
N=int(N) #On convertit N en entier
N=N**4
print(N)
```

Ce programme est correct.

Remarque :

- **on peut écrire les 2 premières instructions en 1 seule.**

```
N=int(input())
N=N**4
print(N)
```

Voilà les conversions les plus utiles :

Conversions	
Chaine de caractères en nombre entier	**int (**chaine**)**
Chaine de caractères en nombre décimal	**float(**chaine**)**
Nombre en chaine de caractères	**str(**nombre**)**
Nombre (1 ou 0) en booléen	**bool(**nombre**)**

À VOUS DE JOUER 5

Ecrivez un script permettant d'afficher le triple d'un nombre entier entré par l'utilisateur.

- **Opération sur les chaînes de caractères**

La principale opération est la concaténation : on ajoute à une chaine de caractère une autre chaine de caractères.

opération	
Concaténation	+

Exemple :

```
A="Nombre de jours en mars: "
B="31"
Texte=A+B
print (Texte)
```

```
*** Console de processus distant Réinitialisée ***
>>>
Nombre de jours en mars: 31
>>>
```

Remarque :

➢ 31 est un nombre, mais on ne peut pas additionner un nombre et une chaine de caractères. On a donc écrit 31 sous forme d'une chaine de caractères.

On peut également convertir 31 en chaine de caractères avec str( ).

```
A="Nombre de jours en mars: "
B=31
Texte=A+str(B)
print (Texte)
```

À VOUS DE JOUER 6

1) Qu'affiche ce programme ?

```
A=52
B=" semaines. "
Texte=str(A)+B
print ("Dans une année, il y a "+Texte)
```

---

2) Pourquoi faut-il convertir A en str ?

AFFICHER ET SAISIR DES DONNÉES

Nous avons rencontré dans ce qui précède 2 instructions qui reviennent régulièrement print (…) et input (…) qui permettent à l'utilisateur de voir des résultats et de saisir des données. Nous allons voir plus en détails comment les utiliser.

- **Print (…)**

Comme on l'a vu print permet d'afficher dans la console Python.

- Si ce qu'on doit afficher est un nombre, on met le nombre.
- Si ce qu'on doit afficher est une variable, on met le nom de la variable et print affichera son contenu.
- Si ce qu'on doit afficher est une chaine de caractère, il faut la mettre entre guillemets (sinon print considère qu'il s'agit d'une variable.

Affichage	
Afficher(A,B,…)	**print(A,B,…)**

```
print(5)  #affiche le nombre 5
A=4
print(A) #affiche la valeur de la variable A donc 4
print("Vivement les vacances") #affiche le texte entre guillemets
```

```
*** Console de processus distant Réinitialisée ***
>>>
5
4
Vivement les vacances
```

On peut également utiliser print avec une expression (somme de 2 nombres, somme de variables,…).

```
print(5+7)  #affiche 12
A=4
print(A+2) #affiche 6
```

On remarque qu'à chaque utilisation de print(), l'affichage passe à la ligne.
On peut utiliser print (….) en séparant ce qu'on veut afficher par des virgules, ce qui permet de mélanger chaines de caractères et variables.

Exemple :

```
A=4
print("Vivement les vacances dans ", A,"jours !")
```

```
>>>
*** Console de processus distant Réinitialisée ***
>>>
Vivement les vacances dans  4 jours !
```

À VOUS DE JOUER 7

Complétez en utilisant les variables nom et prénom pour que le texte suivant s'affiche :

Bilbo Sacquet habite la Comté.

```
nom="Bilbo"
prenom="Sacquet"
print (……)
```

Remarque :

- \n dans un texte fait passer à la ligne.

On peut ajouter un message (appelé **message d'invite**) pour indiquer à l'utilisateur ce qu'on attend.

Exemple

```
N=input("entrer un entier positif")
N=int(N)
print(N**2)
```

Saisir	
Saisir A	A=**input()**
Saisir A avec message d'invite	A=**input(**"message"**)**

À VOUS DE JOUER 8

Ecrivez un script qui demande son année de naissance à l'utilisateur, et l'année courante puis qui affiche son âge :
J'ai <âge> ans.

LA BIBLIOTHÈQUE MATH

Pour l'instant, nous n'avons utilisé que des nombres et des opérations. Très vite, vous aurez besoin d'utiliser la racine carrée des nombres, ou les fonctions trigonométriques. Contrairement à Excel, Python « basique » ne connait pas ces fonctions. Pour y accéder, il faut importer le module **math** dans lequel ces fonctions sont définies.
On doit écrire la ligne de code suivante de préférence au début du programme u(en tout cas avant d'appeler des fonctions mathématiques) : **from math import ***
Exemple

```
from math import *
print(sqrt(64))
```

Ce script donnera : 8

Remarque :

- math est le nom d'un fichier ; * signifie qu'on importe toutes les fonctions et variables définies dans ce fichier.

Principales fonctions mathématiques (nécessite : from math import *)	
Racine carrée de a	**sqrt(a)**
Puissance quelconques a^b	**pow(a,b)**
Sinus, cos, tangente d'un angle en **radian**	**sin(a) ,cos(a),tan(a)**
Nombre π	**pi**
$\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$ (pour déterminer un angle en radian à partir du sinus, cosinus ou de la tangente)	**asin(a) ,acos(a),atan(a)**
Conversion radians→ degrés	**degrees(a)**
Conversion degrés→ radians	**radians(a)**
Partie entière d'un nombre (entier immédiatement inférieur à un nombre). Le résultat donné est un float.	**floor(a)**

Remarque :

- sqrt renvoie un flottant, même si on entre un carré parfait.

À VOUS DE JOUER 9

Écrivez un script qui affiche la racine carrée d'un nombre entré par l'utilisateur.

LES COMMENTAIRES

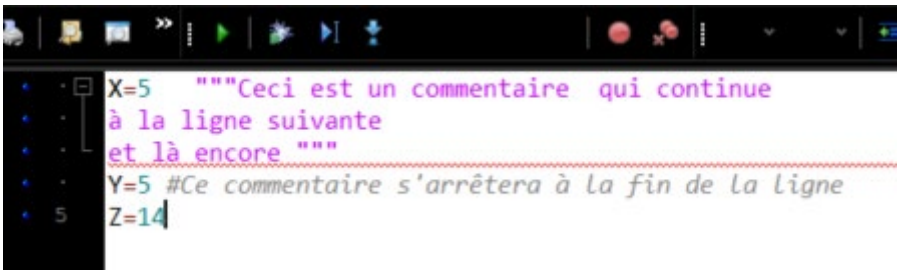
Un commentaire est un morceau de code dont Python ne tient pas compte.
On a déjà rencontré les commentaires précédemment avec le signe #.

```
X=4   #On affecte 4 à la variable X
Y=3   #On affecte 3 à la variable Y
Y=X**Y
print (Y)
```

Le # indique à Python qu'à partir de ce signe **et jusqu'à la fin de la ligne**, il s'agit d'un commentaire.
Dans l'exemple précédent, Python lit :

```
X=4
Y=3
Y=X**Y
print (Y)
```

Si on veut écrire plusieurs ligne de commentaires, ou parfois annihilé des lignes de code, on peut utilisé """
On met """. Tout ce qui suit sera du commentaire, jusqu'à l'insertion d'un autre """.



Commentaire dans une ligne	# commentaire
Commentaire plusieurs lignes	""" commentaire """

Abordons maintenant une série d'exercices, afin de vérifier vos connaissances.
Les exercices ont été classés dans un ordre d'approfondissement croissant.
Les réponses aux exercices se trouvent en fin de manuel.

EXERCICE 01

Que contiennent les variables ? Quel est leur type à la fin du programme ?

```
a=2;b=a+2
c=1/b*3
c=2*c-1
```

EXERCICE 02

Que va afficher le programme suivant ?

```
a=2;b=3
s=a+1/a
s=s+1/b
print (s)
```

EXERCICE 03

1) Ecrivez un script en Python permettant de calculer l'aire et le volume d'un pavé droit dont les dimensions sont entrées par l'utilisateur.

2) Testez ce programme pour un pavé de longueur 3,5, de largeur 2,1 et de hauteur 4.
Contrainte : n'utilisez qu'une seule instruction print ; le résultat doit s'afficher ainsi :

```
*** Console de processus distant Réinitialisée ***
>>>
Aire= 59.5
Volume= 29.4
>>>
```

EXERCICE 04

Voici un programme permettant de calculer l'hypoténuse d'un triangle rectangle connaissant 2 côtés adjacents ainsi que les angles aigus en degrés.

1. Saisir a
2. Lire a
3. Saisir b
4. Lire b
5. c←racine (a^2+b^2)
6. Afficher : "l'hypoténuse vaut :"
7. Afficher c
8. angle←$\tan^{-1}$(a/b)
9. angle←angle*pi/180
10. angle2←90-angle
11. Afficher "les angles valent:"
12. Afficher angle
13. Afficher "et"
14. Afficher angle2

Réécrivez le programme en Python et testez-le.

DÉMARRER AVEC PYTHON

Structures conditionnelles et boucles

ACTIVITÉ 1

Une piscine fait payer € 3,50 € l'entrée avec un tarif réduit de 2,50 si on a moins de 14 ans.

1) Qu'affiche le programme si on entre 16 ?

2) Comment s'appelle la structure si ...alors ?

3) Complétez le programme :

```
age=int(input('Quel est votre âge ?'))
if (age>=14):
   prix=3.5
else :
   prix=2.5
```

STRUCTURES CONDITIONNELLES

- **Structure conditionnelle if… else**

L'ESSENTIEL

Une **structure conditionnelle** permet d'exécuter certaines instructions si une condition est remplie et d'autres instructions si la condition n'est pas remplie.
Elle a cette structure :

→ *Début de la structure :* **Si** *Condition*
Alors
Instructions
Sinon
Instructions

→ *Fin de la structure :* **Fin Si**

Exemple :

On considère l'algorithme qui indique le nombre le plus grand entre 2 nombres décimaux entrés par l'utilisateur.

Entrée des données **:**
A ←Saisir()
B ←Saisir()
Traitement des données
Si A<B
Alors
*Afficher (*B est plus grand*)*
Sinon
*Afficher (*A est plus grand*)*
Fin Si

Traduction en Python :

Structure conditionnelle
if (condition) : **ligne 1** **ligne 2** **…** **else :** **igne 1** **ligne 2** **…**

Remarques :

- Il n'y a pas d'instruction de fin de boucle. Ce qui joue ce rôle est **l'indentation**, c'est-à-dire le décalage d'une ligne à l'autre (par défaut 4 espaces). C'est un élément fondamental en Python.
- **Le début d'un bloc est marqué par les 2 points : .**
- La fin d'un bloc est marquée par un décalage vers la gauche (*Retour arrière* sur le clavier).

```
A=input()
A=float(A)
B=input()
B=float(B)
if (A<B) :
   print("A est plus grand")
else :
   print("B est plus grand")
```

À VOUS DE JOUER 10

2) Programmez en Python :

Entrée des données **:**
X ←Saisir()
Traitement des données
Si X pair
Alors
X←X/2
Sinon
X←X+1
Fin Si

À VOUS DE JOUER 10

5) Que contient X si on entre : 12 ? ……

6) Que contient X si on entre : 13 ? ……

- **Expressions de test**

L'expression à tester était A>B. On a également les opérateurs suivants :
L'expression du test en un booléen.

opérateurs de comparaison	s
Egal : A=B	A==B
Différent : A≠B	A !=B
Inférieur ou égal : A≤B	A<=B
Inférieur : A<B	A<B
Supérieur ou égal	A>=B
Supérieur : A>B	A>B

Il faut bien faire attention au signe =.
En python, A=B est une affectation. A==B est un test.

Un exemple un peu plus compliqué. On veut tester si la somme de 2 nombres saisis appartient à l'intervalle]0 ;100[.

Entrée des données :
A ←Saisir()
B ←Saisir()
Traitement des données
S=A+B
Si 0<S<100
Alors
*Afficher (*OUI*)*
Sinon
*Afficher (*NON*)*
Fin Si

On va traduire la double inégalité 0<S<100 en 2 inégalités simples. $0<S<100 \Leftrightarrow S>0$ et $S<100$. Cela se fait en Python avec le **connecteur logique** and.

```
A=float(input())
B=float(input())
S=A+B
if(S>0 and S<100):
   print('OUI')
else :
   print('NON')
```

Remarque : on a mis une seule ligne d'instruction pour la saisie et la conversion en flotttant des nombres A et B.

opérateurs logiques	
P et Q	P **and** Q
P ou Q	P **and** Q
non P	**not** P

À VOUS DE JOUER 11

Ecrivez la condition en Python

	Python
$x \leq 12$	
$s \in [-2;3[$	
$T \in]-\infty;0[\cup[-2;+\infty[$	
$y \neq -1$	

- **Autres structures conditionnelles**

Si le cas « sinon » ne nécessite pas d'instructions, l'instruction conditionnelle peut prendre la forme :

→ *Début de la structure :* **Si** *Condition*
Instructions

→ *Fin de la structure :* **Fin Si** .

En Python, il suffit de ne pas écrire le bloc **else**.

Exemple

Programme permettant d'afficher la distance à 0 d'un nombre.

```
A=float(input())
if(A<0):
   A=-A
print(A)
```

Il faut remarquer que le print est aligné avec le if. Donc que A soit positif ou négatif, on affiche sa distance à 0.

Structure conditionnelle simple
if (condition) : **ligne 1** **ligne 2** **…**

A contrario, on peut avoir des cas où il y a plus que 2 alternatives.
On utilise alors l'instruction **elif**.

Structure conditionnelle
if (condition) : **ligne 1** **ligne 2** **…** **elif (condition):** **ligne 1** **ligne 2** **…** **elif (condition):** **ligne 1** **ligne 2** **…** **else :** **ligne 1** **ligne 2** **…**

Exemple :
Programme permettant d'afficher les mentions à un examen en fonction de la moyenne.

```
M=float(input())
if(M>=16):
   print("reçu avec Mention Très Bien")
elif (M>=14) :
   print("Reçu avec Mention Bien")
elif (M>=12):
   print("Reçu avec Mention Assez Bien")
elif (M>=10):
   print("Reçu avec Mention très Bien")
else:
   print("Non Reçu")
```

Remarque : les elif sont exécutés dans l'ordre : si la condition est rempli , on exécute le bloc et on sort de la structure. Sinon on passe au elif (ou else) suivant.

BOUCLES INTÉRACTIVES

- **La boucle for…**

Boucle for
for i in range (n,p) : **instruction 1** **instruction 2** **…**

Remarques :
- Ne pas oublier les 2 points et l'indentation.
- **range(n,p) est constitué de tous les entiers entre n et p-1.**
- **range(n) est constitué de tous les entiers entre 0 et n-1**

Attention en particulier à l'indice final !

range (3,8) crée la liste [3,4,5,6,7].
range (8) crée la liste [0,1,2,3,4,5,6,7].
range (0,8) et range (8) correspondent à la même liste.

Liste « de 2 en 2 », de « 3 en 3 »,....
range (premier terme, dernier terme, pas)
range (5,12,3) crée la liste [5,8,11].

Exemple
Programme qui affiche la table de 3 (entre 1 et 10).

```
for i in range(1,11):
    print("3x",i,"=",3*i)
```

À VOUS DE JOUER 12

1) A quels entiers correspond :

range(2,8) : .....

range(0,5) : .....

range(3) : .....

range(2,12,4) : .....

2)

```
M=-2
for m in range(1,4):
    M=M+m
print(M)
```

Le bloc M=M+m est exécuté ..... fois.
Complétez.

		M =-2
étape 1	m=1	M =-1
étape 2	m=...	M =...
étape 3	m=...	M =...

Que va afficher le programme ? .....

3) Ecrivez cet algorithme en Python.

Pour k variant de 2 à 5
    Afficher k^2
Fin pour

.....

.....

.....

.....

- **La boucle while…**

Boucle while
while (condition) : **instruction 1** **instruction 2** **…**

Exemple :
On veut afficher les puissances de 2 inférieures à 40000.

```
texte=""
p=1
while (p<=40000):
  texte=texte + " "+str(p)
  p=2*p
print(texte)
```

À VOUS DE JOUER 13

```
x=1
n=0
while (x<11):
  x=x+3
  n=n+1
print(n)
```

4) Complétez le tableau :

	n	*x*	
	0	1	On entre dans la boucle car x<11
Passage 1	…	…	On (entre/sort) ..... dans la boucle car .....
Passage 2	…	….	On (entre/sort) ..... dans la boucle car .....
Passage 3	…	….	On (entre/sort) ..... dans la boucle car .....
Passage 4	…	….	On (entre/sort) ..... dans la boucle car .....

5) En déduire la valeur affichée.....

6) Ce script permet de déterminer le premier entier n tel que .....

- **Pour aller plus loin : interrompre une boucle**

On peut interrompre une boucle avec la commande **break**.
On détermine les multiples de 3
On considère la suite des nombres 2,5,8,11,….. inférieurs à 100 et on cherche à savoir si parmi ces nombre figure un certain nombre N.

```
test=False
N=51
x=2

while (x<=100):
   if(x==N):
      test=True
      break
   x=x+3
print (test)
```

Interrompre la boucle permet de gagner du temps d'exécution.

EXERCICE 05

Corrigez ce script.

```
a=input()
for i in range(6)
s=a**2+5
sqrt(s)
print (s)
```

EXERCICE 06

1) Que fait ce script ?

```
n=int(input())
s=0
for i in range(n) :
    s=s+3**i
print (s)
```

2) Quel sera le résultat si on entre 3 ?

EXERCICE 07

Qu'affiche ce script ?

```
s=0
for i in range (3):
    for j in range (5):
        s=s+i+j
print (s)
```

EXERCICE 08

$f(x)=x^3-x+4$

Déterminez le minimum et la maximum de la fonction sur [-1;1]. (on fera un balayage avec un pas de 0,01).

Indication : on pourra poser $x=\frac{i}{100}$ ou i est un entier.

EXERCICE 09

$f(x) = 2x^2 + 3x - 9$

Ecrivez un programme testant si l'image d'un nombre x entré par l'utilisateur est strictement négative ou positive.

EXERCICE 10

Console Python

```
>>>
*** Console de processus distant Réinitialisée ***
>>>
 0 1 4 9 16 25 36 49 64 81 100 121 144 169 196
>>>
```

Ecrivez un script Python correspondant à cet affichage.

EXERCICE 11

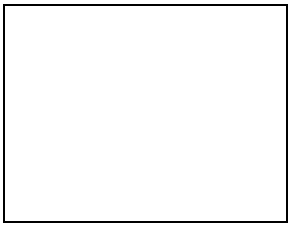
Une loterie propose des billets numérotés de 1 à 1000. Les billets gagnants ont des numéros supérieurs ou égaux à 200 et divisibles par 9.

Ecrivez un programme permettant de savoir si un billet dont on entre le numéro est gagnant.

EXERCICE 12

1) Ecrivez un script pour calculer $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{50}$.

2) Ecrivez un script pour savoir à partir de quelle valeur *n*, $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}>3$



On doit trouver N=11.

EXERCICE 13

Ecrivez un script pour déterminer la première puissance d'un nombre positif donné supérieure ou inférieure à une certaine valeur. Les nombres sont donnés par l'utilisateur.
La demande de saisie et la réponse doivent être claires.

EXERCICE 14

Ecrivez un script qui permet de savoir si un nombre entier entré par l'utilisateur est premier.
On utilisera la propriété suivante d'un nombre premier :
Un nombre N est premier si tous les restes de la division euclidienne de N par les entiers d (avec d compris entre 2 et $\sqrt{N}$) sont non nuls.

DÉMARRER AVEC PYTHON

Fonctions

ACTIVITÉ 2

Voilà un programme Scratch.

1) Que fait ce programme ? Dessinez le résultat.

2) Quel est l'intérêt d'avoir écrit à l'extérieur du bloc principal le bloc carré ?
En Python, un bloc est appelé fonction.

3) Citez 2 fonctions de base de Python.

La plupart du temps, les exercices vous demanderont l'écriture de fonctions.

NOTION DE FONCTION

La notion de fonction en programmation n'a pas grand-chose à voir avec les fonctions vues en mathématiques. Il s'agit d'une "boite noire". Vous en avez déjà rencontré, sans que le mot soit nommé : par exemple print(…) est la fonction qui permet d'afficher; int(…) est la fonction qui permet de convertir un texte en un entier; input(…) est la fonction qui permet de saisir des données, cos(...) calcule le cosinus d'un angle.

On remarque qu'une fonction porte un **nom** : (print, input,…) suivi de parenthèses dont l'intérieur peut être vide ou non. Ce qui figure entre () est utilisé par la fonction pour accomplir sa tâche. Ce sont les **arguments**.

Certaines fonctions ont besoin d'argument(s) : exemple print a besoin de savoir ce qui doit être affiché. D'autres fonctions comme input peuvent être utilisées sans argument.

On a également vu que input renvoyait une chaine de caractères que l'on pouvait affecter à une variable.
On verra que certaines fonctions renvoient un **résultat**, alors que d'autres ne le font pas.

On a utilisé pour l'instant des fonctions de Python préprogrammées. Mais on peut être amené à programmer ses propres fonctions. Cela permet :

- d'avoir des programmes beaucoup plus clairs ;
- d'éviter de réécrire les mêmes lignes de code.

LES FONCTIONS SOUS PYTHON

Définir une fonction

Exemple 1 :

On veut écrire une fonction qui renvoie la table de multiplication de 3 et 6.
On avait écrit précédemment le script pour la table de 3 :

```
for i in range(1,10):
    print("3x",i,"=",3*i)
```

On peut récrire 2 fois la boucle, pour 3 et 6.

```
for i in range(1,10):
    print("3x",i,"=",3*i)
for i in range(1,10):
    print("6x",i,"=",6*i)
```

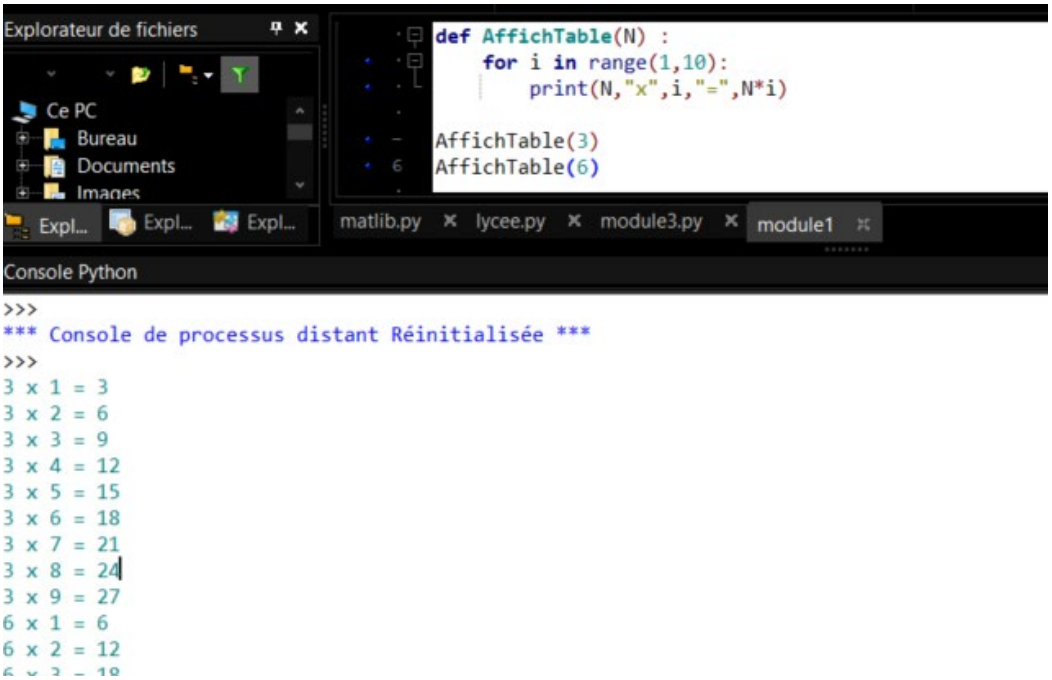
On va maintenant **définir la fonction** AffichTable qui pour N donné va afficher la table de N, puis utiliser cette fonction pour 3 puis pour 6.

```
def AffichTable(N) :
        for i in range(1,10):
            print(N,"x",i,"=",N*i)
```

```
AffichTable(3)
AffichTable(6)
```

La fonction AffichTable nécessite 1 argument :

Voilà la copie écran de ce script et le résultat à l'exécution.



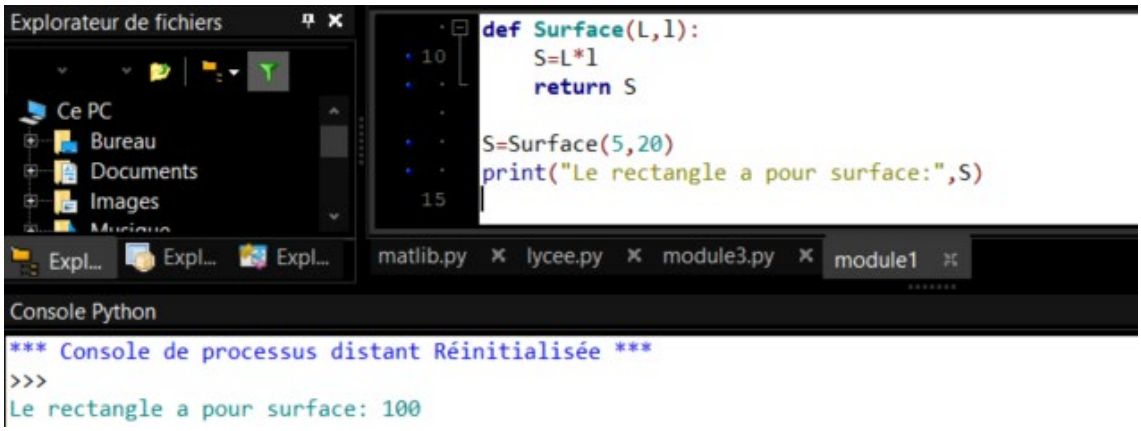
Exemple 2 :

On veut écrire une fonction donnant la surface d'un rectangle en fonction de sa longueur et de sa largeur. Contrairement à ce qui précède, la fonction doit donner un résultat qu'on pourra ensuite utiliser. Cela se fait en Python avec l'instruction **return**.

```
def Surface(L,l) :
    S=L*l
    return S

S=Surface(5,20)
print("Le rectangle a pour surface:",S)
```

Voilà ce que cela donne à l'écran :



Fonction simple sans renvoi
def nom() : **bloc d'instructions** **return *(facultatif)***
Fonction avec arguments sans renvoi
def nom(arg1,arg2,…) : **bloc d'instructions** **return *(facultatif)***
Fonction avec arguments avec renvoi d'un résultat
def nom(arg1,arg2,…) : **bloc d'instructions** **return (resultat=**

Remarque :

- Si la fonction ne renvoie pas de résultat, il est conseillé d'écrire l'instruction **return**. Cela permet de marquer la fin de la fonction et automatiquement, l'indentation retour se fait. Par ailleurs, beaucoup de langages imposent le **return** à la fin d'une fonction.

On peut utiliser return avec une variable, une opération, du texte,…

Exemple :

```
def Surface(L,l) :
        return (L*l)

S=Surface(5,20)
print("Le rectangle a pour surface:",S)
```

Ne pas oublier

- **les 2 points à la fin de la ligne def ;**
- **l'indentation.**

À VOUS DE JOUER 14

```
def chat (m):
    if (m>0) :
        texte="perché"
    else :
        texte="attrapé"
    return (texte)
```

1) Quel est le nom de cette fonction ? .....

2) Utilise-t-elle un argument ? Si ou, lequel ? .....

3) Renvoie-t-elle un résultat ? Si ou, lequel ? .....

- **Appeler une fonction**

On appelle simplement une fonction en écrivant son **nom** et en mettant ses **arguments** sous forme de valeur ou de variable.

```
def Surface(L,l) :
        return L*l

S=Surface(5,20)
print("Le rectangle a pour surface:",S)
```

On a appelé la fonction en entrant les valeurs de longueur et de largeur.
Mais on aurait pu également calculer la surface avec des valeurs saisies par l'utilisateur.

```
def Surface(L,l):
    S=L*l
    return S

Longueur=float(input("Longueur ?"))
Largeur=float(input("Largeur ?"))
S=Surface(Longueur,Largeur)
print (S)
```

Remarque :

- On ne peut pas appeler une fonction avant de l'avoir définie.

À VOUS DE JOUER 15

```
def chat (m):
   if (m>0) :
      texte="perché"
   else :
      texte="attrapé"
   return texte
A=chat(6)
```

Que contient la variable A ? .....

- **Gestion des variables**

Reprenons l'un des exemples précédents :

```
def Surface(L,l):
   S=L*l
   return S

Longueur=float(input("Longueur ?"))
Largeur=float(input("Largeur ?"))
S=Surface(Longueur,Largeur)
print (S)
```

Les variables utilisées par l'appel peuvent être différentes des variables utilisées comme arguments dans la définition.

Dans l'exemple précédent, L et l sont les arguments utilisés dans la définition ; dans l'appel, on a Longueur et Largeur.

Les variables utilisées dans la fonction ne sont pas connues à l'extérieur de la fonction. On dit que ce sont des variables locales.

Ce programme génère une erreur car N n'est pas connu à l'extérieur.

```
def test() :
    N=3
    return
print (N)
```

```
Message
Traceback
  <module>
NameError: name 'N' is not defined
```

Ce programme s'exécute correctement.

```
def test():
    N=3
    return N
print(test())
```

De même les variables utilisées à l'extérieur de la fonction ne sont pas connues à l'intérieur de la fonction. Il faut les passer en argument.

UTILISER UNE FONCTION A L'EXTÉRIEUR D'UN FICHIER

Dans le chapitre précédent, on a défini et utilisé une fonction à l'intérieur d'un même fichier Python appelé **script** qui a pour extension .py. Mais, quand les programmes sont longs, il est parfois judicieux de mettre les définitions de fonctions dans des fichiers séparés qui sont appelés dans ce cas **modules**, qui ont également comme extension .py.

On peut ensuite organiser les modules par dossier qui sont appelés **packages**.

Python propose des packages de base très utiles en particulier, par exemple :

- numpy qui contient tous les modules mathématiques, dont les 2 modules utiles au lycée : math et random. .
- pyplot qui contient les modules nécessaires aux représentations graphiques.

Pour utiliser une fonction maFonction() définie dans un module monModule.py , il y a plusieurs possibilités.

Possibilité 1 :

```
import monModule
monModule.maFonction ()
```

Cela permet d'accéder au module, mais on doit préciser le nom du module quand on utilise la fonction.

Exemple :

```
import math
print(math.cos(2))
```

Possibilité 2:

```
from monModule import *
maFonction ()
```

On importe toutes les fonctions du module. C'est ce qu'on a utilisé jusqu'à présent pour importer le module math. Cela revient à inclure toutes les fonctions dans son programme.

Exemple :

```
from math import *
print(cos(2))
```

Possibilité 3 :

```
from monModule import …..
```

On importe uniquement la ou les fonctions utiles du module.

Exemple :

```
from math import cos,sin
print(cos(2))
```

EXERCICE 15

```
def mystere1(N):
    s=0
    for i in range(2,N+1,2):
        if(N%i==0):
            s=s+i
    return (s)

def mystere2(N,S):
    C=N
    while (C<=s):
        C=C+N
    return (C)

def mystere3(N,S):
    C=N
    n=1
    while (C<=s):
        C=C+N
        n=n+1
    return (n)

def mystere4(N):
    a=N//100
    b=N//10-a*10
    c=N-10*b-100*a
    return c*100+b*10+a

A=mystere1(20)
B=mystere2(20,131)
C=mystere3(20,131)
D=mystere4(241)
```

1) Donnez les valeurs des variables A, B, C et D.

2) Expliquez avec une phrase ce que fait chacune des fonctions mystères.

EXERCICE 16

1) Ecrivez une fonction Trinome (a,b,c,x) qui donne la valeur de $P(x) = ax^2 + bx + c$.

2) $P(x) = 6x^2 - 2x + 4$

Utilisez cette fonction pour calculer $P(-3)$.

```
def Trinome(a,b,c,x) :
    return a*x**2+b*x+c

print(Trinome(6,-2,4,-3))
```

EXERCICE 17

Écrivez en Python une fonction **convert (v,a)** permettant de convertir une vitesse : soit de m/s en km/h si a est nul , soit de km/h en m/s si a vaut 1.

EXERCICE 18

1) Ecrire en Python une fonction **hypotenuse** permettant de calculer l'hypoténuse d'un triangle rectangle connaissant ses 2 côtés adjacents a et b.

2) Ecrire en Python une fonction **perimetre** permettant de calculer l'hypoténuse d'un triangle rectangle connaissant ses 2 côtés adjacents a et b (on utilisera la fonction précédente hypotenuse).

Vous pouvez maintenant faire et envoyer le devoir n°1