Séquence 1 – Notions transversales de programmation

Objectifs

1. Fonctions et appels de fonction
2. Séquences, affectation, conditionnelles
3. Boucles bornées, boucles non bornées
4. Spécification d' une fonction (décrire les préconditions, décrire des postconditions)
5. Mise au point de programmes (utiliser des jeux de tests)
6. Utilisation de bibliothèques

0 Première approche de la pensée algorithmique

Pour s' exercer à la pensée algorithmique et apprendre à travailler la programmation avec Python, nous allons utiliser le site France IOI :

http://www.france-ioi.org/

0.0 S'inscrire :

A faire vous même 1.

- Allez sur France IOI
- Créez votre identifiant et votre mot de passe :
  - Id : prenomclassecordeliers (ex : jean2gcordeliers, tout en minuscules)
  - Mot de passe : nom_JJMMAAAA (nom de famille_date de naissance)
- Notez votre identifiant et votre mot de passe :
  - Id :
  - mdp :
- Connectez-vous
- Cliquez sur `Mon profil` (en haut à gauche)
- Décochez `Afficher seulement les champs requis ou recommandés` et de nouveaux champs apparaissent
- Complétez les champs `Prénom` et `Nom de famille`

0.1 S' enregistrer dans le groupe des Cordeliers

A faire vous même 2.

- Dans l'onglet `Enseigner`, cliquer sur `Groupes et classes`
- Pour rattacher votre compte à votre groupe classe :
  - Cherchez le nom de votre groupe : `Cordeliers1NSI_2023`
  - Cliquez sur le nom du groupe
  - Mot de passe : `Cordeliers1NSI_2023`

Ainsi l'enseignant peut suivre ta progression.

0.2 Progresser :

A faire vous même 3.

- Dans l'onglet `Progresser`, cliquez sur `Cours et Problèmes`
- Choisissez le langage `Python` (en haut à droite)
- Puis le `parcours lycée`
- C' est parti !

Attention, effectue les exercices dans l'ordre !

1 Le langage informatique Python

1.0 Utiliser Python en ligne

Il est possible de travailler en ligne à partir d'un site web :

- basthon : https://console.basthon.fr/
- repl.it : https://repl.it/languages/Python3
- Sofuspy : https://irem.univ-reunion.fr/blockly/plurialgo/blockly/extensions/sofuspy/run.html
- ...

Pratique pour débuter sans rien installer mais cela montre vite ses limites (impossible d' installer des modules supplémentaires, plus difficile de créer/lire/modifier des fichiers, …).

A faire vous même 4.

- Allez sur Basthon
- Sur la page web, il y a deux cases qui s' ouvrent
- Cherchez celle ou il y a le signe : >>>
- Cliquez juste à côté de ce sigle
- Écrivez ceci et tapez sur Entrée :

```
>>> print("Hello world !")
```

- Que se passe-t-il ?
- Vous venez de saisir votre première commande !

IMPORTANT : Dans toute la documentation python, vous allez retrouver ce signe **>>>** Cela correspond à l' invite de commande python.

1.1 Utiliser python avec un IDE (Integrated Development Environment)

Pour gagner en productivité, il est possible d' utiliser un IDE. Voici une liste d' IDE python :

- Spyder
- Eclipse avec extension python
- Visual Studio Code avec extension
- …

Il est possible d'installer et de lancer un « éditeur Python ». Dans notre lycée, nous allons d'abord utiliser Edupython ou Spyder.

A faire vous même 5.

- Allez dans le menu Démarrer
- Cherchez et lancez Edupython
- Un fenêtre s'ouvre. Elle est partagée en 3 sous-fenêtres
- Cherchez celle ou il y a le signe : >>>
- Cliquez juste à côté de ce sigle
- Ecrivez ceci et tapez sur Entrée :

```
>>> print("Hello world !")
```

- Que se passe-t-il ? Vous venez de saisir votre première commande !
- Montrez au professeur

1.2 Deux commandes de base Python

1.2.1 La commande print()

Cette commande permet d'afficher (imprimer) un message.

ATTENTION ! Ce message doit être mis entre guillemets : " … "

A faire vous même 6.

- Écrivez ceci :

```
>>> print(Voici un premier message)
```

- Que se passe-t-il ? Quel est le problème ? Comment le résoudre ? Écrivez la bonne commande
- Écrivez ceci :

```
>>> print("Affichage du deuxième message")
```

- Tapez la commande pour afficher : Python c'est super !
- Essayez les touches flèches haut / flèches bas. Que se passe-t-il ?
- Imprimez les messages suivants :
  - `Vive la rentrée !`

- SNT c’ est génial.
- Les Cordeliers : Meilleur lycée de France !!!

1.2.2 La commande input()

Cette commande permet de demander à un « utilisateur » de saisir un texte.
Attention ! A partir de maintenant, même si vous n’êtes pas schizophrène, vous devez jouer 2 rôles, le rôle du développeur python et le rôle de l’utilisateur python.

A faire vous même 7.

- Écrivez ceci :
  ```
  >>> input("Comment t’appelle-tu ?")
  ```
- Que se passe-t-il ? Que faire ?
- Écrivez ceci :
  ```
  >>> input("Quel âge as-tu ?")
  ```
- Tapez la commande pour afficher la question : Quelle matière préfère-tu ?
- Essayez les touches flèches haut / flèches bas. Que se passe-t-il ?
- Appelez le professeur.

1.3 Calculer avec Python

Il est possible de calculer avec Python. C’est ce que fait la calculatrice Numworks.

- Les classiques : addition, soustraction, multiplication, division
  ```
  >>>2+3
  5
  >>>11-2
  9
  >>>46*5
  230
  ```
- Quotient seuil ou quotient entier de division euclidienne :
  ```
  >>>27 // 7
  3
  ```
- Modulo ou reste de division euclidienne :
  ```
  >>>27 % 7
  6
  ```
- Puissance :
  ```
  >>>5 ** 3
  125
  ```

A faire vous même 8.

- Effectuez le calcul suivant
  - $\frac{4^3+65*13}{27-7}$

2 Notion de variable en Python

2.0 Convention de nommage

Guido Van Rossum, le créateur de Python, a fait des choix forts dès le début. Il a préféré donner un maximum de libertés techniques aux développeurs et, en parallèle, conseiller des bonnes pratiques et des conventions. Ensuite, libre à chacun de s’ y conformer.

Les conventions officielles et notamment les conventions de nommage se trouvent dans la PEP 8 : https://peps.python.org/pep-0008/#naming-conventions

PEP : Python Enhancement Proposal

A savoir : il est conseillé de nommer une variable tout en minuscule avec possibilité d’ utiliser les underscores : `ma_variable, une_autre_variable, ce_que_vous_voulez`. Pas de caractères accentués (é, è, à, …) et pas de caractères spéciaux (&, #, @, …).

A savoir : les constantes (ou variable que ne sont pas destinées à évoluer) doivent être nommées

tout en majuscules et underscores : `PI, NOMBRE_D_OR, CHARGE_ELECTRON`.

2.1 Affectation de variable

Comme dans Scratch, Python manipule des variables (étiquettes dans lesquelles on stocke des valeurs).
Pas besoin de les créer ! Elles se créent automatiquement à la 1ère affectation de variable.

A faire vous même 9.

- Écrivez ceci :
  ```
  >>> toto
  ```
- Que se passe-t-il ?
- Écrivez ceci :
  ```
  >>> toto=2005
  >>> toto
  ```
- La variable `toto` est créé et contient la valeur 2005
- Écrivez ceci :
  ```
  >>>toto=1998
  >>>toto
  ```
- La variable `toto` contient une autre valeur
- Créez une variable `date_coupe_du_monde` en lui affectant `2018`
- Créez une variable `poids_de_mon_chat` en lui affectant `6.125`
- Créez une variable `nom_de_mon_chien` en lui affectant `Médor` (Attention ! Il faut encadrer la valeur avec des guillemets !)
- Choisissez un nom de variable et affectez-lui une valeur
- En vous aidant de ce que nous avons déjà vu, trouvez une commande Python qui affiche/imprime une variable déjà créé
- Trouvez une commande Python qui affiche/imprime plusieurs variables les unes derrière les autres.

A faire vous même 10.

- Allez sur la page : https://www.python-lycee.com/parcours-apprentissage-pl1
- Regardez la vidéo et reproduisez les commandes dans la console en ligne
- Faites les exercices du pdf
- Appelez le professeur.

2.2 Variable et commande input()

Nous pouvons combiner l'affectation de variable et la commande input.

A faire vous même 11.

- Écrivez ceci :
  ```
  >>> prenom=input("Quel est votre prénom")
  >>> prenom
  ```
- Que se passe-t-il ?
- Écrivez une commande permettant de demander la couleur préférée et de la mettre dans une variable couleur
- Écrivez une commande permettant de demander l'âge et de la mettre dans une variable age
- Testez les commandes suivantes :
  ```
  >>> metier="DevOp"
  >>> print("Mon futur métier c'est ", metier)
  ```
- De la même façon, trouvez une commande qui permette d'afficher une phrase (que vous inventerez) qui intègre les 3 variables (prenom, couleur et age)
- Appelez le professeur.

2.3 Le typage dynamique dans python

Chaque variable créée à un type. Les tout premiers sont :

- `str` : Pour les chaînes de caractères
- `int` : Pour les entiers
- `float` : Pour les nombres décimaux

Pas besoin de les définir à l' avance, les variables sont typées en fonction des valeurs qu' elles accueillent.

A faire vous même 12.

- Écrivez ceci :

```
>>> phrase = "Il fait beau aujourd' hui"
>>> type(phrase)
>>> hauteur = 5
>>> type(hauteur)
>>> poids = 70.5
>>> type(poids)
```

Il est parfois possible de passer d' un type à un autre.

A faire vous même 13.

- Conversion en chaîne de caractères :

```
>>> poids_str = str(poids)
>>> type(poids_str)
>>> taille_str = str(taille)
>>> type(taille_str)
```

- Conversion en types numériques

```
>>> hauteur2 = int(hauteur_str)
>>> type(hauteur2)
>>> poids2 = float(poids_str)
>>> type(poids2)
>>> poids3 = int(poids_str)
```

- Que se passe-t-il ?

ATTENTION ! La variable créée par une commande input sera systématiquement une chaîne de caractères. Il faudra veiller à la convertir si nécessaire.

2.4 Opérations sur les types numériques

- Différentes représentations numériques :
  - classique : `5`
  - binaire : `0b1100`
  - hexadécimale : `0xab`
  - octale : `0o75`
  - A noter : bin permet de générer une chaîne de caractères correspondant au nombre binaire : `bin(55)`
- Opérations arithmétiques :
  - Les classiques : `+ - * /`
  - Quotient de division euclidienne : `3.0 // 2`
  - Reste de division euclidienne : `5 % 2`
  - Négation : `-x`
  - Valeur absolue : `abs(-5)`
  - Puissance : `5**2`
- Opération bit à bit :
  - or : `0b01 | 0b10`
  - and : `0b01 & 0b10`
  - xor : `0b01 ^ 0b10`
  - not : `~0b10`
  - décalage de bit vers la droite : `0b10010 >> 2`
  - décalage de bit vers la gauche : `0b10010 << 3`

A faire vous même 14.

- Testez toutes ces possibilités dans une console python

2.5 Opérations sur les chaînes de caractères

- Important : Tout ce qui se trouve derrière le caractère `#` n' est pas interprété par python ce qui permet de commenter le code
- Une chaîne de caractères peut être définie au choix par :
  - des guillemets : `"Quel est votre prénom ?" "Comment va-tu aujourd'hui"`
  - des apostrophes : `'a' 'document' 'il me dit : "Où es-tu ?"'`
  - des triples guillemets ou triple apostrophes :

```
"""Les triples guillemets
ou triple apostrophes
```

```
permettent d’ écrire sur plusieurs lignes"""
```

- On peut concaténer deux chaînes

```
>>> "super !" + "On s’ éclate ici !"
"super !On s’ éclate ici !"
```

- On peut multiplier par un nombre entier

```
>>> "super !" * 3
"super !super!super !"
```

- Il est possible de ne récupérer qu’ une partie de la chaîne, c’ est le slicing
  - Attention ! Le premier caractère a l’ indice 0

```
>>> ma_chaine = "Tout ceci est formidable."
>>> ma_chaine[3] #On part de l’ indice 0
‘t’
>>> ma_chaine[:6] #Tous les caractères jusqu’ à l’ indice 6 EXCLU
‘Tout c’
>>> ma_chaine[6 :] #Tous les caractères depuis l’ indice 6 INCLU
‘eci est formidable.’
>>> ma_chaine[6 : 12]
‘eci es’
>>> ma_chaine[6 : -1] #jusqu’ au dernier exclu
‘eci est formidable’
>>> ma_chaine[6 : -2] #jusqu’ à l’ avant-dernier exclu
‘eci est formidabl’
```

- Testez toutes ces possibilités dans une console python

3 Mes premiers scripts

La fenêtre avec les >>> s’appelle la *console*.
La console est intéressante quand on veut tester des commandes simples.
Dès qu’il faut enchaîner plusieurs commandes, il vaut mieux écrire un script. Un script est un simple fichier texte avec une extension en .py.
En ligne ou dans un IDE, il faut changer dans la sous-fenêtre *script*. Il faudra d’ abord enregistrer son son fichier et ensuite l’ exécuter.
En ligne de commande, il nous faut :

- générer un fichier texte (avec un éditeur comme Vim ou Nano) que l’ on peut nommer mon_script.py.
- Exécuter ce fichier en tapant :

```
$ python mon_script.py
```

Important ! Il vaut mieux tout de suite mettre les bons entêtes dans le script permettant d’ indiquer à l’ OS hôte que c’ est bien un script python et qui permette d’ utiliser tous les caractères accentués :

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

A faire vous même 15.

- Créez votre premier script avec le bon entête
- Développez un petit programme qui dialogue avec un utilisateur. Il devra :
  - Demander son nom
  - Demander sa date de naissance
  - Demander la date d’ aujourd’hui
  - Afficher une phrase avec tous ses renseignement et aussi l’ âge de la personne.
  - Pour les plus rapides, vous pouvez demander d’ autres renseignements (poids, taille, …) et faire d’ autres affichages (calcul de l’ IMC par exemple).
  - Pour les plus rapides, testez ceci dans la console :

    ```
    >>> print('\033[31m' + 'ce texte est rouge')
    ```

    Soignez votre affichage, utilisez des couleurs et des effets
    Plus d’info :
    https://chamilo.univ-grenoble-alpes.fr/courses/IUT1RT1M2109/document/1718-Sokoban/build/sequences_ansi.html

A faire vous même 16.

- Écrivez un programme qui calcule le volume d'un parallélépipède rectangle dont on demande au départ la largeur, la hauteur et la profondeur.

A faire vous même 17.

- Écrivez un programme qui convertisse un nombre entier de secondes (qui est demandé à l' utilisateur au départ), en un nombre d'années, de mois, de jours, de minutes et de secondes. (Utilisez l'opérateur modulo : %).
- Faites un bel affichage final avec toutes ces informations

A faire vous même 18.

Turtle est un module Python qui permet de tracer des traits dans une fenêtre.
Il va nous permettre d'illustrer une partie de ce que nous avons déjà étudié.
Quelques commandes :

`reset()`	effacer le dessin
`goto(x, y)`	aller à l'endroit de coordonnées x, y
`forward(distance)`	avancer d'une distance donnée
`backward(distance)`	reculer d'une distance donnée
`up()`	relever le crayon (pour ne plus dessiner)
`down()`	abaisser le crayon (pour dessiner)
`color(`*couleur*`)`	<*couleur*> est une chaîne : 'red', 'blue', 'green',...
`left(angle)`	tourner à gauche d'un angle exprimé en degrés
`right(angle)`	tourner à droite d'un angle exprimé en degrés

Attention : pour que l'utilisateur puisse fermer la fenêtre de la tortue suivant le processus usuel il faut achever le programme par l'instruction :

```
mainloop()
```

- Testez le script suivant :

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

from turtle import *
color('red')
forward(200)
left(90)
forward(50)
done()
```

4 Les fonctions

4.0 Une première approche simple avec Turtle

Il est possible de mettre des instructions que l'on utilise plusieurs fois dans une fonction. Une fonction est définie avec le mot-clé : **def**

A faire vous même 19.

- Dans la fenêtre script écrivez :

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
from turtle import *
def dessineUnCarre() :
    fillcolor('orange')
    begin_fill()
    forward ( 75 )
    left(90)
    forward ( 75 )
    left(90)
    forward ( 75 )
    left(90)
    forward ( 75 )
    left(90)
    end_fill ()
```

- Exécutez-le … Il ne se passe rien !
- Mais l'ordinateur a maintenant en tête les instructions pour faire un carré.

- Dans la console, vous pouvez saisir (n'oubliez pas les parenthèses à la fin) :
```
>>> dessineUnCarre()
>>> done()
```
- On vient de faire appel à la fonction `dessineUnCarre`

4.1 IMPORTANT ! L'indentation dans Python

L'indentation, c'est-à-dire le décalage vers la droite, des lignes est fondamental en Python. C'est cela qui détermine les blocs de commandes qui s'applique à telle ou telle fonction, telle ou telle instruction conditionnelle ou qui appartient à telle ou telle boucle.
Voyez dans l' exemple ci-dessus, tout ce qui est indenté appartient à la fonction.
Voir la PEP8, il est préconisé une indentation de 4 (quatre) espaces.

4.2 Notion de paramètre

Il est possible de passer un ou plusieurs paramètres à la fonction. On les renseigne dans les parenthèses de la fonction.

A faire vous même 20.

- Dans la fenêtre script écrivez :
```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
from turtle import *
def dessineUnCarreParametre(cote) :
    fillcolor('orange')
    begin_fill()
    forward ( cote )
    left(90)
    forward ( cote )
    left(90)
    forward ( cote )
    left(90)
    forward ( cote )
    left(90)
    end_fill ()
```
- Important : Ne pas oublier le caractère `:` en fin de ligne et avant l' indentation
- Dans la console, vous pouvez saisir :
```
>>> dessineUnCarreParametre()
```
- Que se passe-t-il ? Pourquoi un message d' erreur ?
- Photographiez-bien ce message, vous le reverrez régulièrement sur vous faites du développement python
- Testez maintenant :
```
>>> dessineUnCarreParametre(50)
>>> dessineUnCarreParametre(200)
```

4.3 Les fonctions mathématiques

Les fonctions python sont définies grâce au mot clé *def*.
Chaque fonction prend des paramètres (que l'on retrouve dans les parenthèses et retourne des valeur (juste après le mot-clé *return*).

A faire vous même 21.

F est la fonction définie sur $\mathbb{R}$ par :

$$\begin{cases} si\ x \leqslant 0\ alors\ f(x) = x^2 \\ si\ 0 < x \leqslant 1\ alors\ f(x) = x \\ si\ 1 < x\ alors\ f(x) = -2x + 3 \end{cases}$$

- Écrivez le programme python d'une fonction f qui donne pour résultat f(x) suivant la valeur de x et enregistrez-le dans premiere_fonction.py
```
def f(x) :
    if x<=0 :
        resultat=x**2
    elif x<=1 :
        resultat=x
```

```
    else :
        resultat=-2*x+3
    return resultat
```

- Puis testez

```
>>> from premiere_fonction import f
>>> toto=f(5)
>>> toto
-7
>>> toto=f(0.5)
>>> toto
0.5
```

A faire vous même 22.

- Sur une droite graduée, A et B sont deux points d'abscisses a et b.
  - Exprimez la distance AB en fonction de a et b
  - Écrivez la fonction python d'une fonction dictanceDeuxPoints(a,b) qui donne pour résultat la distance AB

A faire vous même 23.

- Allez à l' adresse : https://www.python-lycee.com/parcours-apprentissage-pl2
- Regardez la vidéo et pratiquez avec les outils de la page
- Répondez aux 4 exercices de fin de page

4.4 Commentaires et docstrings

Pour rappel : Tout ce qui se situe derrière un # n' est pas pris en compte et sert à commenter son code python.
Pour chaque fonction, il est possible de rédiger une docsctring, ou une petite notice explicative de la fonction. C' est une bonne pratique qui permet aux autres de facilement utiliser vos fonctions. Les trois choses principales à indiquer :

- Explication de ce que fait la fonction
- Les paramètres : Lesquels, signification de chaque, quel(s) type(s), valeurs par défaut, …
- La(les) sortie(s) de la fonction : Lesquels, signification de chaque, quel(s) type(s)

A faire vous même 24.

- Reprenez la fonction f ci-dessus et complétez-la comme suit :

```
def f(x) :
    """Math function with 3 different definitions
    param : x, float
    return : resultat, float
    """
    if x<=0 :
        resultat=x**2
    elif x<=1 :
        resultat=x
    else :
        resultat=-2*x+3
    return resultat
```

- Exécutez votre script afin de mettre la fonction en mémoire
- Tapez :

```
>>> help(f)
```

- La docstring s' affiche.
- Dans certains IDE, laisser la souris sur la fonction permet l' ouverture automatique d' une info-bulle avec la docstring

A faire vous même 25.

- Pour les plus rapides : Sphinx est un module python qui compile toutes les docstring et permet de faire automatiquement une belle documentation web en ligne.
- Commencez à lire, installer et essayer sphinx : https://pypi.org/project/Sphinx/

5 Les séquences - for … in range():

Quand on veut répéter une commande un certain nombre de fois (3 fois 7 fois, 632 fois, …), on utilise une séquence `for … in … .`

A faire vous même 26.

- Boucle sur tous les caractères d' une chaîne :

```
une_chaine = "uvwxyz"
for i in une_chaine :
    print(i)
```

- Important : Ne pas oublier le caractère `:` en fin d' instruction `for … in` et avant l' indentation
- Création d' un compteur numérique :

```
for cpt in range(15) : # de 0 à 15 exclu
    print(cpt) # saut à la ligne à chaque print
```

- Création d' un compteur numérique II :

```
for cpt in range(7, 15) : # de 7 à 15 exclu
    print(cpt, end=" ") # pas de saut de ligne
```

- Création d' un compteur numérique III :

```
for cpt in range(7, 15, 3) : # pas de 3
    print(cpt, end="-") # séparés par des tirets
```

A faire vous même 27.

- Allez à l' adresse : https://www.python-lycee.com/parcours-apprentissage-pl4
- Regardez la vidéo et pratiquez avec les outils de la page
- Répondez aux 4 exercices de fin de page

A faire vous même 28.

- Écrivez un script Python qui réalise les choses suivantes :
  - Afficher un message : Début de programme
  - Demander à un utilisateur : Jusqu'à combien dois-je compter ?
  - Afficher le comptage (Et de 0. Et de 1. Et de 2 ...)
  - Afficher un message : Fin de programme
- Pour les plus rapides : Faites une compte-à-rebours dont voici la sortie :
  - `Attention ! Prêt au décollage !`
  - `5 !`
  - `4 !`
  - `3 !`
  - `2 !`
  - `1 !`
  - `0 !`
  - `Décollage …`
- Pour les plus rapides : Rappelez-vous les caractères de couleurs au A faire vous-même 12 ...

A faire vous même 29.

- Écrivez un programme qui affiche les 20 premiers termes de la table de multiplication par 7.
- Écrivez un programme qui affiche les `n` premiers termes de la table de multiplication par `m` avec `n` et `m` des valeurs demandées à l' utilisateur.
- Écrivez une fonction qui affiche les `n` premiers termes de la table de multiplication par `m` avec `n` et `m` paramètres de la fonction.

A faire vous même 30.

- Avec turtle, écrivez une fonction dessineUnTriangleEquilateral qui utilisera 2 paramètres : couleur, longueur
- Pour les plus rapides : Modifiez le script ci-dessus et ajoutez une fonction dessineUnPolygone qui utilisera 3 paramètres : couleur, longueur, nombre_de_cotes

A faire vous même 31. Facultatif

- Pour les plus rapides et les plus motivés, manipulation d' images numériques : http://ninoo.fr/LC21_22/NSI_1ere/seq1_le_langage_informatique_python/1_activite_modification_image_avec_python.pdf

A faire vous même 32.

- Écrivez un programme qui affiche les 20 premiers termes de la table de multiplication par 7.

A faire vous même 33.

- Écrivez un programme qui calcule le volume d'un parallélépipède rectangle dont on demande au départ la largeur, la hauteur et la profondeur.
  A noter : la fonction int() convertit une donnée saisie en entier :

```
>>>int('21')
21
```

A faire vous même 34. Pour les costauds

Écrivez un programme qui convertisse un nombre entier de secondes fourni au départ, en un nombre d'années, de mois, de jours, de minutes et de secondes. (Utilisez l'opérateur modulo : %).

6 Comparez deux valeurs

- Inférieur

```
>>> a = 1
>>> b = 2
>>> a < b
True
```

- Inférieur ou égal

```
>>> a = 1
>>> b = 2
>>> 2*a <= b
True
```

- Egalité de valeur

```
>>> a = 1
>>> b = 2
>>> 2*a == b
True
```

- Supérieur

```
>>> a = 1
>>> b = 2
>>> a > b
False
```

- Supérieur ou égal

```
>>> a = 1
>>> b = 2
>>> 2*a >= b
True
```

- Différence de valeur

```
>>> a = 1
>>> b = 2
>>> 2*a != b
False
```

A faire vous même 35.

- Avec python, testez si les 2 valeurs suivantes sont égales (valeurs True/False)

$$\frac{3}{5}+\frac{2}{7}$$

$$\frac{31}{35}$$

7 Instructions conditionnelles

Dans un programme, si une condition est vraie alors on exécute des commandes et si elle est fausse on exécute d'autres commandes.

A faire vous même 36.

- Dans la fenêtre script écrivez :

```
1.  # -*- coding: UTF-8 -*-
2.  nombre1 = 55
3.  nombre2=input("Quel est votre nombre ?")
4.  nombre2=int(nombre2)
5.  if nombre2 > nombre1 :
6.      print("Votre nombre est plus grand que le nombre du programme")
7.  elif nombre2 < nombre1 :
8.      print("Votre nombre est plus petit que le nombre du programme")
9.  else :
10.     print("Bravo ! Vous avez trouvé !")
```

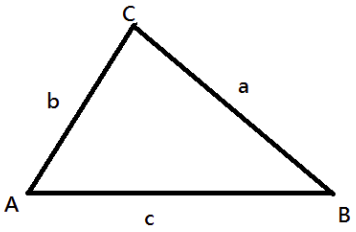
- Attention ! Les lignes 6, 8 et 10 ont 4 espaces en début de ligne
- Testez ce script
- Si vous avez compris, vous passez à la suite. Sinon vous appelez le professeur.

A faire vous même 37. Pour les costauds

- Écrivez un programme qui calcule les 50 premiers termes de la table de multiplication par 13, mais n' affiche que ceux qui sont des multiples de 7.

A faire vous même 38. Pour les costauds

ABC est un triangle dont on connaît les longueurs en cm, des côtés.
Écrire le script python d'une fonction rectangleOuPas(a, b, c) qui donne pour résultat True (vrai) si le triangle est rectangle ou False (faux) sinon.



8 Affectation augmentée

- Exemple simple :

  « a += 1 » est équivalent à « a = a + 1 »

```
>>>a = 15
>>>a += 1
>>>a
16
```

- Opérateurs compatibles : + - * /

```
>>>a = 15
>>>a *= 2
>>>a
30
```

9 Boucles non bornées - while

Une boucle non bornée ne s'arrête que quand une condition est remplie.

A faire vous même 39.

- Dans la fenêtre script écrivez

```
# -*- coding: UTF-8 -*-
choix = "o"
nombre=1
while choix=="o" :
    nombre = nombre *2
    print("Puissance de 2 : ", nombre)
    choix = input("Tapez o pour continuer ")
print("AU REVOIR !")
```

- A noter : Se trouvent dans la boucle les 3 commandes qui sont décalés de 4 vers la droite

10 Mes premiers scripts

Vous voici dans Edupython ou Spyder.
Pour Edupython, la fenêtre avec les >>> s'appelle la console.
Pour Spyder, la fenêtre avec les In [1] s'appelle la console.
La console est intéressante quand on veut tester des commandes simples.
Dès qu'il faut enchaîner plusieurs commandes, il vaut mieux changer de sous-fenêtre et aller dans l'autre fenêtre pour écrire un script.

A faire vous même 40.

- Dans la fenêtre script écrivez une deuxième fois :

```
# -*- coding: UTF-8 -*-
print("Début du script")
for i in range(5) :
    print("A la ", i)
print("Fin du script")
```

- ATTENTION ! METTEZ 4 ESPACES AU DEBUT DE LA 4e LIGNE !
- Enregistrez ce script quelque part
- Exécutez-le (bouton avec une flèche verte)
- Retrouvez-le avec votre explorateur de fichiers
- Quel extension ont les scripts Python ?
  ……
- A noter : La 1ère ligne permet simplement d'utiliser les caractères accentués (é, è, ë, ê, à, …). Elle est à ajouter à chaque début de script.

- Appelez le professeur.

A faire vous même 41.

- Écrivez un script Python qui réalise les choses suivantes :
  - Afficher un message : Début de programme
  - Demander à un utilisateur : Jusqu'à combien dois-je compter ?
  - Afficher le comptage (Et de 1. Et de 2. Et de 3 ...)
  - Afficher un message : Fin de programme

11 IMPORTANT ! L'indentation dans Python

L'indentation, c'est-à-dire le décalage vers la droite, des lignes est fondamental en Python. C'est cela qui détermine les blocs de commandes qui s'applique à tel ou tel instruction conditionnelle ou qui appartient à telle ou telle boucle.

A faire vous même 42.

- Dans la fenêtre script écrivez

```
# -*- coding: UTF-8 -*-
choix = input("Choisissez une couleur : rouge ou bleu")
if choix=="rouge" :
    print("Vous avez choisi : ", choix)
    print("Vous avez bien choisi")
print("AU REVOIR !")
```

- Dans la fenêtre script modifiez comme ceci :

```
# -*- coding: UTF-8 -*-
choix = input("Choisissez une couleur : rouge ou bleu")
if choix=="rouge" :
    print("Vous avez choisi : ", choix)
print("Vous avez bien choisi")
print("AU REVOIR !")
```

- Quelle est la différence entre les deux scripts ?

  .....

12 Consolidation des connaissances avec Turtle

Turtle va nous permettre d'illustrer une bonne partie de ce que nous avons déjà étudié.
Rappel : Vous avez un bref descriptif de quelques commandes un peu plus haut dans le cours.

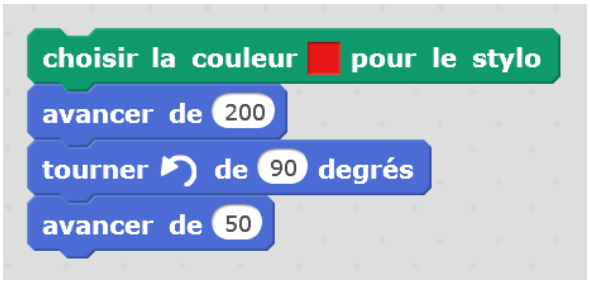
A faire vous même 43.

- Voici un 1er exemple à saisir :

Python	Scratch

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

from turtle import *
color('red')
forward(200)
left(90)
forward(50)
done()
```



- Exécutez-le
- A noter : La 1ère ligne permet de bien préciser à l'ordinateur qu'il s'agit d'un programme python.

A faire vous même 44.

- Voici un 2e exemple avec une boucle while :

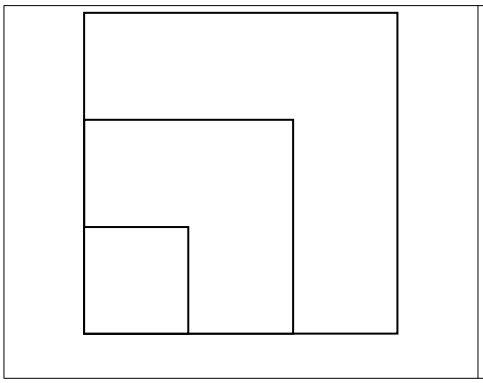
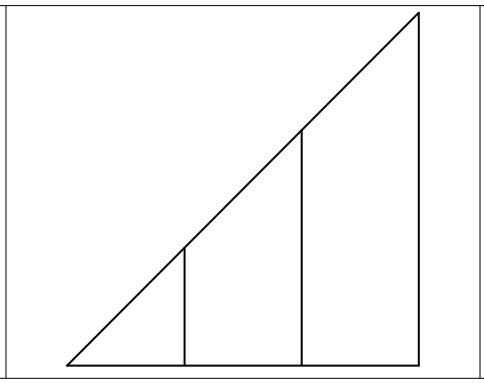
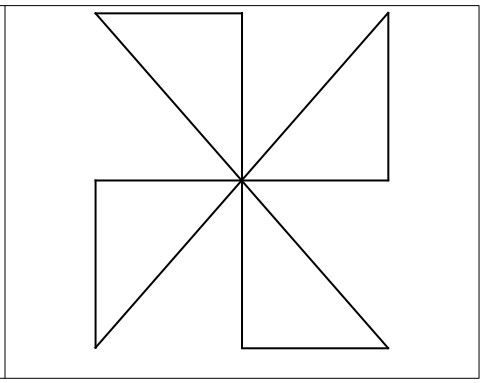
```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

from turtle import *
color('red', 'yellow')
begin_fill()
while abs(pos()) > 1:
    forward(200)
```

```
        left(170)
    end_fill()
    done()
```

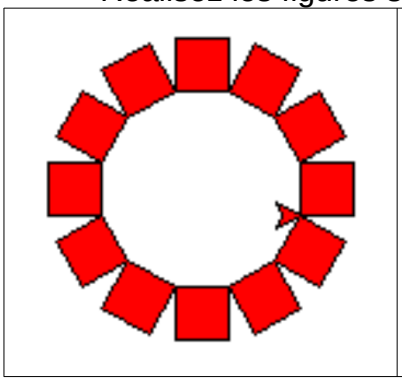
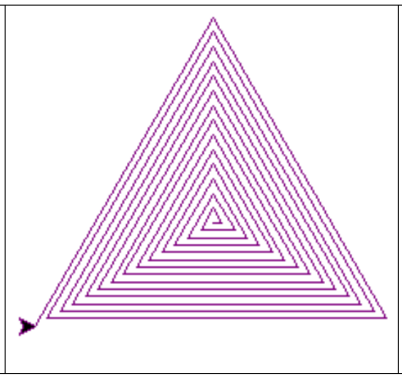
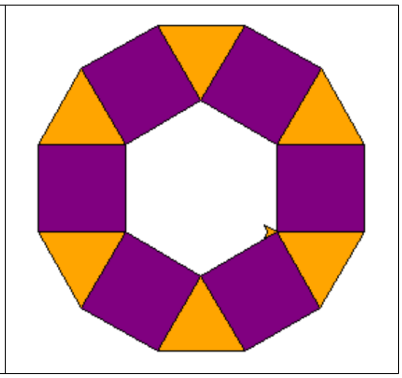
A faire vous même 45.

- Ecrire et programmer les algorithmes permettant de réaliser les figures ci-dessous en Python. Vous pourrez utiliser l'instruction « répéter »
-

A faire vous même 46. Pour les costauds

- Réalisez les figures suivantes :

A faire vous même 47.

- Dans la fenêtre script écrivez

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

from turtle import *
def dessineUnCarre2(couleur_choisie):
    pendown()
    fillcolor(couleur_choisie)
    begin_fill()
    for i in range(4) :
        forward(75)
        left(90)
    end_fill()
    penup()


for j in range(3):
    dessineUnCarre2('blue')
    forward(100)
done()
```

- Exécutez-le
- L'ordinateur a en tête les instructions pour faire un carré et, en fin de script, appelle cette fonction.
- On vient de faire appel 3 fois à la fonction `dessineUnCarre2`

A faire vous même 48.

- Modifiez le script ci-dessus et ajoutez une fonction dessineUnRectangle qui utilisera 3 paramètres : couleur, longueur, largeur

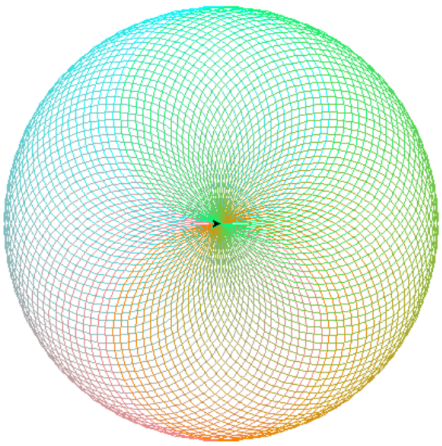
```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
def dessineUnRectangle(couleur_choisie, longueur, largeur):
    pendown()
    fillcolor(couleur_choisie)
    begin_fill()
    …
    ...
    end_fill()
    penup()
```

A faire vous même 49.

- Modifiez le script ci-dessus et ajoutez une fonction `dessineUnTriangleEquilateral` qui utilisera 2 paramètres : couleur, longueur

A faire vous même 50. Pour les costauds

- Utilisez la fonction `circle()` pour reproduire le dessin ci-après :



13 Mini-projet pour manipuler chaînes de caractères et fichier texte

13.0 Modalités :

* Par groupe de 2 élèves.
* Rendre un script Python répondant à ce qui est demandé
* Voir document : http://ninoo.fr/LC/1ere_NSI/seq1_le_langage_informatique_python/1_le_langage_informatique_python_projet.pdf