Langage et programmation – Les bases de la programmation Python

Contenus	Capacités attendues
Constructions élémentaires	Mettre en évidence un corpus de constructions élémentaires.
Diversité et unité des langages de programmation	Repérer, dans un nouveau langage de programmation les traits communs et les traits particuliers à ce langage.
Mise au point de programmes	Utiliser des jeux de tests
Utilisation de bibliothèques	Utiliser la documentation d'une bibliothèque

Problématique :

Comment faire exécuter un algorithme par un ordinateur ?

1 – Premiers pas en Python

Il existe de très nombreux langage de programmation.
Un langage de programmation fait le lien entre l'utilisateur et la machine (qui n'utilise que des bits et ne stocke que des bits). Ce langage simple d'apprentissage a été créé en 1989 à Amsterdam.

Son utilisation se fera pendant les séances à l'aide de l'environnement Spider (Anaconda).

Python peut être utilisé de deux manières :

- En mode interactif : on exécute directement les commandes sans les avoir enregistrées au préalable.
- En rédigeant une séquence d'instruction dans un fichier que l'on exécute ensuite.

Python en mode calculatrice :
Les opérateurs sont :

+ addition,
* multiplication,
- différence,
/ division usuelle,
// donne le quotient d'une division (division euclidienne) ,
% désigne le reste de la division euclidienne.

Opérateurs de test :

== désigne l'égalité de deux objets ;
!= vérifie si deux objets sont différents ;
<, >, <= >= pour les tests de comparaison.
Les opérateurs de test renvoient un booléen : vrai ou faux.

Les opérandes désignent des valeurs combinées à l'aide de l'ordinateur. Ils peuvent être des nombres ou bien des variables.

ESEO – Mise à niveau informatique

Les bases de la programmation Python

Attention :

- Le séparateur décimal est le point et non la virgule !
- Il faut se méfier des résultats numériques. Python enregistre des valeurs approchées.

Remarque : pour utiliser les fonctions mathématiques, il faut importer une bibliothèque appelée aussi un module.
Pour importer le module : **import math**

Quelques fonctions issues du module :

- math.sqrt(x) renvoie la racine carrée
- math.sin(x), cos(x), tan(x) les fonctions trigonométriques
- math.floor(x) renvoie la partie entière de x
- math.abs(x) renvoie la valeur absolue de x
- math.pow(x, y) renvoie x^y
- math.exp(x) ; log10(x) …

Remarque : il est aussi possible d'importer une fonction spécifique, la syntaxe est alors différente :

```
from math import sqrt
print(sqrt(3))
```

On voit parfois également :

```
from math import* #On importe toutes les fonctions de la bibliothèque math
print(sqrt(3))
```

On préfèrera importer une bibliothèque par import bibli ; puis adopter la syntaxe bibli.fonction()

Variables :
L'affectation : elle s'effectue avec le symbole =. Ainsi x = 8 signifie que l'on range la valeur 8 dans la variable x.
Attention à bien distinguer le signe = de l'affectation du signe == teste de l'égalité.

Une variable peut être de plusieurs types :

- **int** : entiers relatifs stockés en valeur exacte
- **float** : nombre à virgule flottante
- **bool** : booléens (True False)
- **str** : chaîne de caractère

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

Afficher un texte ou une variable :
print(« Votre texte »)
print(x)
\n : saut de ligne dans un texte
: afficher un commentaire dans le fichier ; la ligne est écrite en verte et n'est pas exécutée.

Demander à l'utilisateur une valeur type str :
a = input(« texte ») # le texte doit toujours être mis entre guillemets « ».
→ Affiche la phrase texte et attend que la personne entre le type qui sera stockée dans la variable a

Demander à l'utilisateur une valeur numérique :
a = eval(input(« texte »)) # le texte doit toujours être mis entre guillemets « ».
→ Affiche la phrase texte et attend que la personne entre une valeur numérique qui sera stockée dans la variable a.

Pour approfondir : http://apprendre-python.com ; https://fr.wikibooks.org/wiki/Programmation_Python ; https://www.flossmanualsfr.net/initiation-a-python/

ESEO – Mise à niveau informatique

Les bases de la programmation Python

2 – Premiers programmes Enregistrer votre fichier (.py).

A – Règle d'implémentation et de syntaxe

Méthode résolution problème : poser en français le problème / l'algorithme sur papier, le passage en langage Python est l'étape d'**implémentation**.

Attention à bien commenter (# pour afficher un commentaire) un programme pour faciliter sa lecture par une tierce personne (notamment votre professeur ...). On prendra soin de donner également des noms judicieux aux variables introduites.

Règle d'écriture Python :

- Python distingue les majuscules des minuscules. Les instructions Python sont écrites en minuscule.
- En Python, les instructions qui suivent le début d'un bloc doivent être **indentées**.
- Attention : une ligne doit comporter moins de 80 caractères ; on pourra continuer une ligne à l'aide du symbole \
- Toujours placer un espace après une virgule, un point-virgule, deux points.

Applications :
A – Ecrire un programme qui demande un nombre à l'utilisateur puis calcul et affiche le carré de ce nombre.

B – Ecrire un programme qui convertisse en radian un angle fourni au départ en degrés~~, minutes, secondes.~~

C – Ecrire un programme affichant la TVA sur le prix d'un objet (l'utilisateur fournit le prix de l'objet).

D – Ecrire un programme donnant le prix d'un objet après réduction d'un pourcentage de remise.

B – Instructions conditionnels et itératives

I – Structure alternative

La structure algorithmique :
Si *condition :*
bloc 1
Sinon *:*
bloc 2

Devient en Python :

```
if condition :
        bloc 1
else :
        bloc 2
```

NB : les indentations sont en générales automatiques. Elles sont indispensables. Ne pas oublier les : .

Remarque :
- Clause elif (sinon si) peut être ajoutée entre un if et un else.
- Les conditions sont des structures booléennes pouvant prendre la valeur True ou la valeur False.
- Il est possible d'imbriquer des structures alternatives.

Conditions :
test d'égalité == ;
de non égalité : != ,

On dit qu'une condition est complexe lorsqu'elle s'écrit comme une combinaison logique de conditions simples. Elle fait alors appel aux opérateurs logiques OU , ET et NON :

NON : not,
OU : or ;
ET : and

Exemple :
if condition 1 and condition 2

Applications :
E – Ecrire un programme qui demande un nombre à l'utilisateur puis informe si ce nombre est positif, négatif ou nul

F – Ecrire un programme demandant la taille et le poids d'une personne, affiche son IMC ainsi qu'un conseil personnalisé.

G – Ecrire un programme qui convertissant en degrés Celsius une température exprimée au départ en Kelvin et inversement.

H – Ecrire un programme qui demande la distance parcourue par un mobile, en m, ainsi que le temps du parcours, en s. Le programme affiche : la vitesse en km/h et si le mobile dépasse la vitesse limite autorisée de 50 km/h.

I – Ecrire un programme qui demande sous forme de nombres l'heure qu'il est (un nombre pour les heures, un pour les minutes et un pour les secondes). Cet algorithme indiquera ensuite s'il s'agit d'une heure valide ou non.

J – Ecrire un programme donnant les racines d'un polynôme du second degré.

K – Ecrire un programme permettant de convertir en base 2, puis en base 16, n'importe quel entier écris en base 10.

L – Ecrire un programme donnant l'horoscope ainsi que des conseils beauté.

II – Structure répétitive :

La structure algorithmique :

Pour *i allant de 1 à n-1 faire*
Bloc 1

Devient en Python :

```
for i in range (0, n) :
    Bloc 1
```

Le Bloc 1 s'exécutera donc une fois pour chaque valeur de i.
La structure répétitive permet donc de donner une suite d'entier.

Elle permet également de donner une suite de lettre dans une chaîne de caractère :

```
nom = « Mathieu »
for lettre in nom :
    print (lettre)
```

```
M
a
t
h
i
e
u
```

Remarque : par défaut la fonction print rajoute un saut de ligne entre chaque valeur.

Applications :
M – Programmer l'algorithme "table de multiplication".

N – Ecrire un programme donnant tous les multiples de 13 pour les entiers inférieurs à 1000. Amélioration : l'utilisateur choisit le multiple et choisit la borne inférieure.

O – Ecrire un programme qui affiche une suite de 12 nombres dont chaque terme soit égal au triple du terme précédent.

P – Ecrire un programme qui affiche les 20 premiers termes de la table de multiplication par 7, en signalant au passage (à l'aide d'un astérisque) ceux qui sont des multiples de 3. Exemple : 7 14 21* 28 35 42* 49 …

III – Boucle conditionnelle while

La structure algorithmique :

Tant que *condition*
Bloc

Devient en langage Python :

```
while condition :
        Bloc
```

Attention : il faut s'assurer que la boucle ne soit pas infinie. La condition doit donc cesser d'être répétée.

Remarque : pour arrêter un programme : **CTRL + F2.**

Instruction break :

```
if condition :
        break
```

Cette instruction entraîne une sortie de boucle et permet dans certains cas de simplifier l'écriture d'un programme.

Applications :

Q – Un programme donnant le nombre de mois au bout desquels la valeur d'une voiture (d'une valeur initiale de N €) atteint 5000 € si sa cote diminue de 2% par mois.
Amélioration : afficher le nombre d'années et le nombre de mois.

R – Ecrire un programme modélisant le jeu « juste prix » (utilisation d'une recherche dichotomique). Amélioration : ajouter un compteur.

S – Ecrire un programme qui calcule le périmètre et l'aire d'un triangle quelconque dont l'utilisateur fournit les 3 côtés. Ce programme vérifie au préalable, à l'aide d'une boucle while, que les données fournissent bien un triangle. Pour calculer l'aire d'un triangle quelconque, on utilisera la formule de Héron :

S = (d*(d-a)*(d-b)*(d-c))**0,5

dans laquelle d désigne le demi-périmètre, et a,b,c celles des trois côtés.

T – 1992 représente la particularité de pouvoir s'écrire 1992 = 83 x 8 x 3. On dira que l'année 1992 est redondante. Quelle sera la prochaine année redondante, c'est-à-dire dont le millésime peut s'écrire sous la forme AB x A x B ?