40h

Introduction - Prise en main

Les bases du langage Python

Programmation Orientée Objet

20h

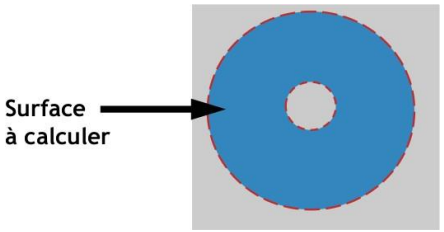
Mini-projet

Livret de TP

Les bases

Application directe du cours

1. Écrire un programme, qui définit 3 variables : une variable de type texte, une variable de type nombre entier, une variable de type nombre décimal et qui affiche leur type.
2. Affecter dans une même ligne les 3 variables précédemment définies.
3. Écrire un programme qui, à partir de la saisie d'un rayon et d'une hauteur, calcule le volume d'un cône droit.
4. Une machine découpe dans une plaque, des disques circulaires de rayon r_{Ext}, percés d'un trou circulaire de rayon r_{Int} avec $r_{Int} < r_{Ext}$ et ne débordant pas du disque.
   Quelle est la surface d'un disque découpé ?



Application réfléchie

1. Écrire un programme qui affiche le type du résultat des instructions suivantes :
   - `a=3`
   - `a==3`
2. Écrire un programme, qui ajoute une chaîne de caractères à un nombre entier (`le chat + 3`).
3. Écrire un programme, qui donne la mesure de l'angle α d'un triangle rectangle, dont on saisit le côté opposé et l'hypoténuse

Application avancée

1. Écrire un programme qui réalise la saisie d'un nombre entier puis affiche la valeur ainsi saisie et son type. Essayer de dépasser la taille maximale des entiers (on utilisera la méthode *getsizeof*() du module *sys*)
2. Lors de la saisie d'un nombre par cast (`int (input())`) : indiquer une chaîne de caractères en lieu et place d'un nombre, rechercher comment éviter ce bug (aide : commande `try`)

CHAPITRE 2

Conditions - Itérations - Répétitions

Application directe du cours

1. Écrire un programme `min_max.py`, qui demande de saisir 2 valeurs et qui affiche la plus petite des 2 valeurs.
2. Écrire un script `longeur_chaine.py`, qui demande de saisir 2 chaînes de caractères et qui affiche la plus grande des 2 chaînes.
3. Écrire le script `convertir.py`, qui effectue une conversion euros en dollars.
   - Le programme commencera par demander à l'utilisateur d'indiquer par un caractère 'E' ou '$' la devise du montant qu'il va entrer.
   - Puis le programme exécutera une action conditionnelle de la forme :

```
if devise == 'E'  :
   .........
elif devise == '$'  :
   ..........
else  :
   .......... # affichage d'un message d'erreur
```

4. Écrire un programme, qui affiche 50 fois "Je dois ranger mon bureau" à l'aide de l'instruction `for`.

5. Chanson traditionnelle bretonne
   La séquence d'instructions

```
n=10
print ("C'est dans {} ans je m'en irai j'entends le loup le renard chanter".format(n))
```

permet d'afficher le message :

```
C'est dans 10 ans je m'en irai j'entends le loup le renard chanter
```

Écrire une boucle while qui permet d'afficher :

```
C'est dans 10 ans je m'en irai j'entends le loup le renard chanter
C'est dans 9 ans je m'en irai j'entends le loup le renard chanter
C'est dans 8 ans je m'en irai j'entends le loup le renard chanter
...
C'est dans 1 ans je m'en irai j'entends le loup le renard chanter
```

Dans un premier temps, on ne s'occupera pas de la faute d'orthographe de la dernière ligne.

Application réfléchie

1. Écrire le script `multiple3.py` qui affiche en fonction d'une valeur saisie l'un des messages suivants :
   – "Ce nombre est pair"
   – "Ce nombre est impair, mais est multiple de 3"
   – "Ce nombre n'est ni pair ni multiple de 3"
2. Écrire un programme qui affiche les nombres de 2 en 2 jusqu'à 100 avec un `for` puis avec un `while`.
3. Écrire un programme qui affiche les tables de multiplications de 1 à 10.
   Aide : utiliser une boucle imbriquée.

Application avancée

1. Écrire un programme qui affiche un joli sapin de Noël, dont la taille est donnée par l'utilisateur.
   Exemple pour une taille de 6 lignes :

```
     ^
    ^^^
   ^^^^^
  ^^^^^^^
 ^^^^^^^^^
^^^^^^^^^^^
```

Utilisation d'un module : Turtle

Rappel des principales commandes

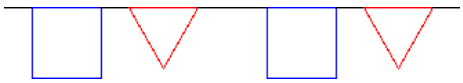
Utiliser Turtle	`import turtle`
Efface tout	`reset()`
Aller à	`goto (x,y)`
Avancer	`forward (distance)`
Reculer	`backward (distance)`
Couleur	`color (couleur)`
Tourner gauche	`left (angle)`
Tourner droite	`right (angle)`

Application directe du cours

1. Écrire un programme `carre.py` qui trace un carré.
2. Écrire un programme, qui trace un triangle équilatéral (3 angles à 60°).
3. Écrire un programme, qui trace un hexagone (polygone à 6 côtés, angles interne à 120°).

Application réfléchie

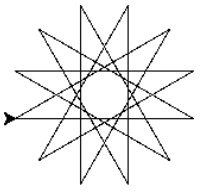
1. Écrire un programme, qui trace un carré, puis un triangle.
   Modifier ensuite votre programme pour dessiner `n` figures consécutives adjacentes.



2. Écrire un programme, qui trace un carré puis un triangle, qui grossissent au fur et à mesure.

Application avancée

1. Écrire un programme, qui trace un cercle (non parfait), sans utiliser la fonction `circle` de Turtle.
2. Écrire un programme, qui trace une étoile.



Les Listes

Rappel des principales commandes

Supprimer un élément d'une liste	`remove()`
Ajouter un élément d'une liste	`append()`
Trier une liste	`sort()`
Inverser l'ordre d'une liste	`reverse()`
Pour	`for variable in`
Tant que	`while condition :`
Longueur d'une chaîne	`len (chaine)`

Préambule

Dans l'ensemble de ces exercices, la liste suivante sera utilisée :
- lapin
- chat
- chien
- chiot
- dragon
- ornithorynque

Application directe du cours

1. Écrire un programme `liste_animaux.py`, qui initialise la liste et qui affiche l'ensemble des éléments.
2. Afficher la liste de manière inversée.
3. Afficher la liste de manière triée.
4. Ajouter (`append`) l'élément `troll` dans la liste, puis supprimer l'ensemble des animaux domestiques. Afficher le résultat.
   Afin de réaliser la suppression, on créera une liste des animaux domestiques.

Application réfléchie

1. Écrire un programme `liste_chaine.py`, qui donne le nombre de caractères de chaque élément de la liste.
   Exemple : `lapin possède 5 caractères.`

Dans les deux exercices suivants, on considèrera un tableau initialisé avec 10 valeurs aléatoires. Le but des exercices est de dire, si la valeur saisie par l'utilisateur est dans la liste ou non.

Afin de vous aider, voici le début des deux programmes.

```
import random

tableau_jeu=[]

# Initialisation d'une liste de 10 éléments
for i in range (0,10) :
   tableau_jeu.append (random.randint (1,10))
```

2. Recherche séquentielle dans une liste non triée.
   Le programme parcourt la liste des valeurs en la comparant une à une à la valeur cherchée, et sort de la boucle :
   – soit quand il a parcouru toute la liste sans trouver la valeur
   – soit quand la valeur a été trouvée
   Si la valeur a été trouvée, le programme annonce "Gagné", sinon, le programme annonce "Perdu".
3. Recherche séquentielle dans une liste triée.
   Le programme trie les valeurs tirées au sort, puis parcourt la liste des valeurs en la comparant une à une à la valeur cherchée. Il sort de la boucle :
   – soit quand il a parcouru toute la liste sans trouver la valeur
   – soit quand la valeur lue dans la liste est supérieure à la saisie
   Si la valeur a été trouvée, le programme annonce "Gagné", sinon, le programme annonce "Perdu".
4. Jeu de cartes - Une couleur
   – Créer un programme `jeu_de_cartes.py`, qui crée une liste avec l'ensemble des cartes soit : `1,2,…10,V,D,R` sans spécifier la couleur. On se permettra une saisie manuelle des valeurs des cartes.
   – Contrôler le contenu de votre liste en l'affichant
5. Jeu de cartes complet Créer une nouvelle liste avec les couleurs, afficher l'ensemble du jeu de cartes.
   Exemple : `R de Coeur`.

Application avancée

1. 2 Jeux de Cartes
   – Créer un programme `jeu_de_cartes2.py`, qui crée deux listes avec l'ensemble des cartes (soit $2 * 52$ cartes).
   `Exemple : "1Coeur"` pour As de Coeur.
   – Contrôler le contenu de vos 2 listes en les affichant
   – On tire 10 cartes au hasard dans chaque jeu. Faire afficher les 10 cartes tirées pour chacun des 2 jeux.
   – Trouver les cartes identiques dans les 2 tirages.

Fonctions

Rappel des principales commandes

Définition d'une fonction	`def nom_fonction (param1, param2, ...):`
Retour d'une valeur	`return valeur`
Vrai	`True`
Faux	`False`

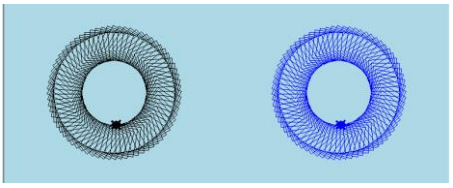
Préambule

Dans l'ensemble des exercices ci-dessous, il vous est demandé d'écrire des fonctions … mais aussi le programme, qui lui est associé afin de les tester !

Application directe du cours

1. `tortue_carre.py` : écrire une fonction `dessine_carre`, qui fait tracer un carré de 50 pixels de côté à la tortue.
2. Modifier le programme précédent de la façon suivante :
   (a) la tortue dessine un carré
   (b) la tortue tourne de 5°
   (c) la tortue avance de 5 pixels
   (d) la tortue dessine un carré

   Et ceci 72 fois.
   Nommer le `tortue_carre_tournant.py`
3. Modifier le programme précédent de manière à faire des lunettes …



4. `pair_impair.py` : écrire 2 fonctions :
   - `pair (nbre)`, qui renvoie True, si le nombre est Pair
   - `impair (nbre)`, qui renvoie True, si le nombre est Impair

   Le nombres sera demandé à l'utilisateur. Le résultat attendu est : `"La fonction Pair retourne True pour la valeur 2"`

5. `mini_maxi.py` : écrire 2 fonctions :
   - `mini (a,b)` qui renvoie le minimum entre a et b
   - `maxi (a,b)` qui renvoie le maximum entre a et b

   Les 2 nombres a et b seront demandés à l'utilisateur.

Application réfléchie

1. Modifier le programme `tortue_carre_tournant`.py en `tortue_carre_couleur.py`, pour que la couleur du carré soit passée en paramètre à la fonction `dessine_carre`.
2. Modifier le programme `tortue_carre_couleur.py` en `tortue_triangle_couleur.py`, pour qu'en lieu et place d'un carré, ce soit un triangle équilatéral, qui soit dessiné.

Application avancée

1. `tortue_carre_france.py` : Reprendre le programme `tortue_carre_couleur.py` et créer un liste des trois couleurs du drapeau français, afin de dessiner les carrés avec ces 3 couleurs.
2. `palyndrome.py` : Créer une fonction qui indique, si une chaîne de caractères est ou non un palyndrome [1].

   Aide

   La commande `list(chaine)` permet de transformer une chaîne de caractères en liste de caractères.

   La commande `"".join(list)` permet de transformer une liste en chaîne de caractères.

3. `pendu.py` : Créer un programme de Pendu.

1. Un palyndrome est un mot qui peut s'écrire dans les deux sens, exemple : radar