ALGORITHMIQUE DE BASE

v=**input('**texte'**)**	Affiche le mot 'texte' et attend une valeur qui sera stockée dans la variable v
print "texte", **print** v	Affiche le mot 'texte', Affiche la valeur de la variable v
if *cond* **:** *bloc1* **else :** *bloc2*	Effectue le *bloc1,* si la condition *cond* est vérifiée, sinon effectue le bloc 2
for *i* **in range (***n***) :** *bloc*	Répète n fois l'instruction *bloc*. **i prend les valeurs de 0 à n–1**
while *cond* **:** *bloc*	Répète l'instruction *bloc t*ant que la condition *cond* est vérifiée

OPÉRATIONS, TESTS, TYPES,....

a **=** 17	Affectation de 17 à la variable a
+ , – , * , / et **a**b**	Les quatre opérations et a^b
==, <=, >=, !=	Être respectivement égal, inférieur ou égal, supérieur ou égal, différent
and, or, not	ET, OU, et négation
type(n**)**	Renvoie le type de l'objet n . On peut l'afficher avec print (type(n))
int, float, chr, bool	Entier, décimal, caractère, booléen (**True** , **False**)

MODULES DE BASE

from *module* **import ***	Importe tout. On écrit ensuite directement les fonctions : sqrt(n)
Import *module*	Importe tout. Faire ensuite précéder les fonctions par le nom du module exemple : math.sqrt(n).
help('module**')**	Donne la liste des fonctions du module

Le module math

sqrt(x **)**	Racine carrée de x
sin(x**), cos(**x**),tan(**x**)**	Les fonctions trigonométriques
floor(x**)**	Partie entière de x
abs(x**)**	Valeur absolue de x
a / b	Quotient <u>entier</u> de a par b (avec Python 2) Quotient décimal de a par b (avec le module **lycee** ou avec Python 3 **)**
a % b	Reste de la division de a par b

Le module random

randint (a,b)	Renvoie un entier aléatoire dans [a ; b]
random ()	Renvoie un flottant dans [0 ; 1]
uniform(a,b)	Renvoie un flottant aléatoire dans [a ; b[selon la loi uniforme

AIDE-MEMOIRE PYTHON Lycée Europe R.Schuman Cholet

INSTRUMENTATION

print ()	A appliquer à des variables clefs dés qu'une vérification est nécessaire
#	Désactive une ligne de programme (test ou commentaires)
" " " **" " "**	Commentaires des fonctions.
Ctrl F2	Pour reprendre la main, réinitialiser python (ex si on fait une boucle infinie)

LES CHAÎNES

str	Nom de type des chaînes de caractères. Signifie 'string'
ch**=raw_input('**quest' **)**	Ouvre une fenêtre avec la phrase 'quest' et attend une texte qui sera stockée dans la variable ch
len*(ch)*	Renvoie la **longueur** de la chaîne ch
ch[i **]**	Renvoie le caractère i de la chaîne ch. **N'est pas modifiable.** Le **1er caractère est au rang 0,** et le dernier est au rang **len(ch) –1**

Extraire et transformer une chaîne

mot1 **+** mot2	Colle les deux chaînes mots1 et mots2 . On dit **concaténer**
mot ***nb**	Recopie nb fois la chaîne mot
mot**.upper()**	Renvoie la chaîne mot en **majuscule**
mot**.lower()**	Renvoie la chaîne mot en **minuscule**
mot **[** a **:** b **]**	Extrait de la chaîne mot, les caractères compris **du rang a au rang b–1**
str(nb**)**	Transforme en chaîne de caractères le nombre nb
eval(ch**)**	Transforme en valeur numérique la chaîne de caractères ch

Rechercher et remplacer

c **in (**ch**)**	Renvoie True si le caractère c est dans la chaîne ch, et False sinon
ch**.count('**text' **)**	Compte le nombre d'occurrence de *text* dans la chaîne ch
ch**.find('**text' **)**	Renvoie la position de *text* dans la chaîne ch, et –1 si *text* n'y est pas
ch.**replace ('**tx1'**,'**tx2'**)**	Remplace chaque *tx1* par *tx2* dans la chaîne ch

LES FONCTIONS

def *fonc* **(**a,b, .. **) :**	Définie une fonction *fonc,* de paramètres *a,b, ...* Il peut ne pas y en avoir.
I **''' documentation '''**	Descriptif pour se souvenir de ce que fait la fonction
I	
I blocs d'instructions	**Variables locales** : Les paramètres et variables dans la fonction
I	
I **return** *résultat*	Valeur *résultat* de l'appel à la fonction. Si pas de résultat : return None
fonc(x, y,..**)**	Appel la fonction *fonc.* Avec ou sans paramètre.
global a, b	Modifie dans la fonction les variables a et b du programme principal
g=**lambda** *a,b* **:***instruction*	fonction «anonyme» : une instruction, pas de return, arguments possibles

LES FICHIERS

fich=**open(**'*nom*',*mode*) *mode* : **'r', 'w', 'a'**	Ouvre le fichier *nom*, se trouvant dans le dossier contenant le programme. Modes : lecture (read), écriture (write), ou ajout (append). Dans les deux dernier cas le fichier est écrasé, ou s'il n'existe pas est créé
fich**.close()**	Fermer le fichier après utilisation

Lecture, écriture et ajout

for *ligne* **in** *fich*	La variable *ligne* prend pour valeurs successives chaque ligne de *fich*
L=fich**.readline()**	**Lit** une ligne du fichier, la stocke dans L, passe ensuite à la ligne suivante. NB : le saut de ligne **\n** compte pour un caractère. Pour l'enlever il faut faireL= fich**.readline().replace(' \n ' , ' ')**
L[i]	Renvoie le caractère i de la ligne L.
fich**.write(** text+ '\n'**)**	Écrit le texte *text* dans le fichier fich en terminant par un saut de ligne

MODULE TKINTER

from Tkinter **import ***	Importe le module Tkinter
fen**=Tk()**	Crée une fenêtre nommée fen
fen**.title(** 'titre' **)**	Affiche titre dans la fenêtre
fen**.geometry(** '500x150' **)**	Redimensionne la fenêtre en 500pixels de large par 150 de haut
fen**.mainloop()**	Gestionnaire d'événement : lance la surveillance de la fenêtre.

Widgets Button, Label, Entry

B = **Button (** fen,... **)**	Défini un **bouton** dans la fenêtre *fen*, avec certains paramètres
L = **Label(** fen, **text =**'x' , .. **)**	Défini un **texte** *x* dans la fenêtre *fen*
E = **Entry(**fen,..**)**	Défini une **zone**, où l'on peut taper un texte
*obj***.pack()**	Méthode rapide **d'affichage** du widget *obj*
obj **.place_forget()** *obj* **.destroy()**	**Cache** le widget *obj* **Efface** le Widget *obj*
obj **.cget("** …" **)**	Renvoie la valeur de l'option de *obj* indiquée entre " .. "
obj **.config(**..,.., **)**	Modifier une ou plusieurs options de *obj*

Paramètres principaux

fg=' ' , **bg=' '**	Couleur du texte , de l'arrière plan
height=' ' , **width=' '**	Hauteur, et la largeur . Pas d'effet sur la police
font= "police taille décoration**"**	Police utilisée et mise en forme. Ex : **"** 'arial' ,15 , bold **"**
text=' '	*Button et Label seuls* Précise le texte à afficher .
command= fonc	*Button seul* précise la fonction(sans parenthèse) à lancer lors du clic.

Méthodes pour Entry

AIDE-MEMOIRE PYTHON *Lycée Europe R.Schuman Cholet*

E**.get()**	Renvoie le texte entré dans Entry
E**.insert(** i , text **)** E**.insert(INSERT** , text **)** E**.insert(END** , text **)**	Insert dans E le texte *text* : à la position i ou à la place du curseur ou à la fin du contenu existant
E**.delete(** i **)** E**.delete(** a, b **)** E**.delete(** 0,END **)**	Efface le caractère à la position i ; ou les caractères entre les positions a et b ; ou tout le champ de texte

WIDGET CANVAS

c**=Canvas(** fen, options**)** *options :* ***bg=' ', height=, width=***	Appel du Canvas c dans la fenêtre fen Couleur du fond, hauteur et largeur du Canvas
c**.place(x= ,y=)**	Placement par coordonnées (0 ; 0) en haut à gauche
c**.grid(**options**)**	Placement sur une grille par n° de lignes et colonnes : **row**=..... **column**=..... **rowspan**=.... **columnspan**=....

Créer des items :dessins, texte, images.

seg=**c.create_line(**x1,y1,x2,y2,options) *options* ***: width= , fill=***	**Segment** reliant les coordonnées (x1,y1) et (x2,y2) exclu *Épaisseur, couleur*
rect=**c.create_rectangle** **(**x1,y1,x2,y2,options) *options* ***: width= , fill= ,outline=***	**Rectangle** de diagonale le segment seg précédent. *Épaisseur, couleur intérieure, couleur du trait*
ov=**c.create_oval (**x1,y1,x2,y2,options)	**Cercle** (ou ellipse) inscrit dans le rectangle précédent
txt**=c.create_texte(**x,y,options**)** *options :* ***fill=, font=, text=*** ***anchor=***	**Texte** au point de coordonnées (x,y). *Options de couleur, police,texte à afficher, et d'ancrage* *par rapport à (x,y) : n, e, s, w, nw, …. sinon centré*
img=**PhotoImage(file="** image.gif **")**	Importe l'image et la place dans la variable img
obj**=c.create_image(** x , y , ***image***= img**)** *autre option :* ***anchor=***	Crée un objet *obj* avec l'image *img*, au point (x,y) *Position par rapport à (x,y) : N, E, S,O, ...*

Modifier des items : dessins, textes, images

c.coords (*item*, ……..**)**	Modifie les coordonnées de l' item *item. 2 ou 4 coordonnées* *Renvoie la liste des coordonnées de l'item si rien de précisé.*
c.itemconfig(item,options**)**	Modifie une ou plusieurs options de l' item *item.*
c.itemcget(item,prop**)**	Renvoie la valeur de la propriété prop de l'item *item.*
c.update()	Active la modification
c.tag_raise(item**)** **c.tag_lower(**item**)**	Place l'item *item* au **premier plan** ou en **arrière plan**
c.delete(*item* **)** **c.delete(ALL)**	Efface l' item *item ou tout le contenu du Canvas c*

CODAGE, POLICE	
eval(' 0b...') ou **eval('**0x....)	Renvoie la valeur décimale d'un nombre en binaire (précédé de 0b) ou hexadécimal précédé de 0x)
bin (…) hex (...)	Renvoie la valeur binaire (précédé de 0b) ou hexadécimal (précédé de 0x) d'un nombre décimal.
text=('police',taille,'décoration'**)**	Polices fréquentes : Arial, Times New Roman, .. Décoration : bold=gras, underline=souligné, overstrike=barré

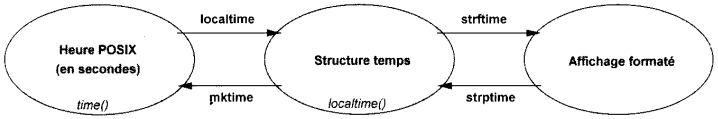
GESTION DU TEMPS

Module Tkinter

fen.after(delai**,** fonc**)**	Déclenche une fonction fonc (sans parenthèse) après un délai en millisecondes, sans bloquer le reste du programme.

Module time

Rappel : avec **import time** faire précéder chaque fonction par **time.** ...	
sleep(t**)**	Effectue une pause de t secondes
time()	**Renvoie la date POSIX** (depuis 01/01/1970), **en secondes.**
localtime( )	**Renvoie la structure de temps** de la date POSIX
*st***.tm_year,** *st***.tm_mon,** *st***.tm_day** *st***.tm_hour,** *st***.tm_min,** *st*. **tm_sec**	**Renvoie la valeur numérique d'un élément de la structure** de temps *st* : l'année, le mois**,** ……,les secondes
strftime('fmt',st**)** 'fmt**='**%H %M %S %d %m %Y'	**Converti en chaîne de caractère au format fmt** la structure de temps st : les éléments de la date sont précédés de %...
strptime('txt**','**fmt'**)**	**Convertit en structure temps** une chaîne txt au format fmt
mktime(st **)**	**Convertit en heure POSIX** une structure temps st



AIDE-MEMOIRE PYTHON *Lycée Europe R.Schuman Cholet*

LES LISTES

*list***=[ ]** ou *list***=[** e1**,** e2**, ...]**	Crée une liste vide ou avec les éléments e1, e2, …..
*list***=[** randint(a,b) for i in range(c) **]**	Crée une liste aléatoire de c entiers entre a et b
list **[i]**	Renvoie l'élément d'indice i de la liste.
list **[i] =** *elem*	**Stocke** dans l'élément d'indice i l'élément *elem.*
len(*list*)	Renvoie le nombre d'éléments de la liste.
elem **in** *list*	**Teste** si l'élément *elem* est dans la liste. Renvoie vrai ou faux
for e **in** *list* **:**	**Parcourt** tous les éléments d'une liste

Méthodes (modifie la liste)

list.**append**(*e*)	**Ajoute** l'élément *e* à la fin de la liste.
list.**remove**(*e*)	**Supprime** la première occurrence de l'élément *e.* *Tester avant si elem est présent, sinon message d'erreur.*
list.**pop**(i)	**Supprime** l'élément d'indice i de la liste
list.**insert**(i,*e*)	**Insère** l'élément *e* au rang i de la liste
list.**sort**()	Trie la liste appelée (la modifie)
list.**index**(e)	Renvoie l'indice de la première occurrence de e
list.**count**(e)	Renvoie le nombre de e dans la liste

Fonctions (ne modifie pas la liste**)**

min(*list*) et **max**(*liste*)	Donne le plus petit et le plus grand élément de la liste
choice(*liste*)	Choisit au hasard un élément de la liste. (avec module random)
sorted(*liste*)	Renvoie une nouvelle liste ordonnée avec les éléments de la liste *liste*

Conversion entre chaînes et listes

ch.**split**(*'sep'*)	Renvoie une liste en découpant la chaîne *ch* chaque occurrence de *sep*
sep.**join**(*list*)	Renvoie une chaîne en accolant les éléments de la liste avec le séparateur *sep*

GESTION DU CLAVIER	
….. .**bind('<KeyPress>',** *fonc*)	Surveille si une touche est appuyée et appelle la fonction *fonc.*
fen.**bind('<KeyPress>',***fonc*)	**Surveille l'appui d'une touche** pour la **fenêtre** *fen*
*E***.bind('<KeyPress>',***fonc***)**	**Surveille l'appui d'une touche** pour un **Entry**
fen.**bind('<KeyRelease>',***fonc*)	**Surveille le relachmenet d'une touche** pour la **fenêtre** *fen*
def fonc**(** evt**) :**	La fonction appelée a obligtoirement un **paramètre** (ici *evt*)
evt**.char**	Renvoie le **caractère** de la touche appuyée
evt**.keysym**	Renvoie un **chaîne** avec **le nom de la touche** appuyée : Escape, space, Up, Down, Left, Right, F1, Return, …….
<KeyPress> remplacé par a, A, 1, +, … ou par <Esc>, <space>,<Up>,<F1>, ...	**Surveille l'exécution de(s) touche(s) désignée(s).** Pour les touches multiples utiliser un tiret : '<Controle-Up>'

AIDE-MEMOIRE PYTHON *Lycée Europe R.Schuman Cholet*