MINISTÈRE D'ÉDUCATION TUNISIENNE

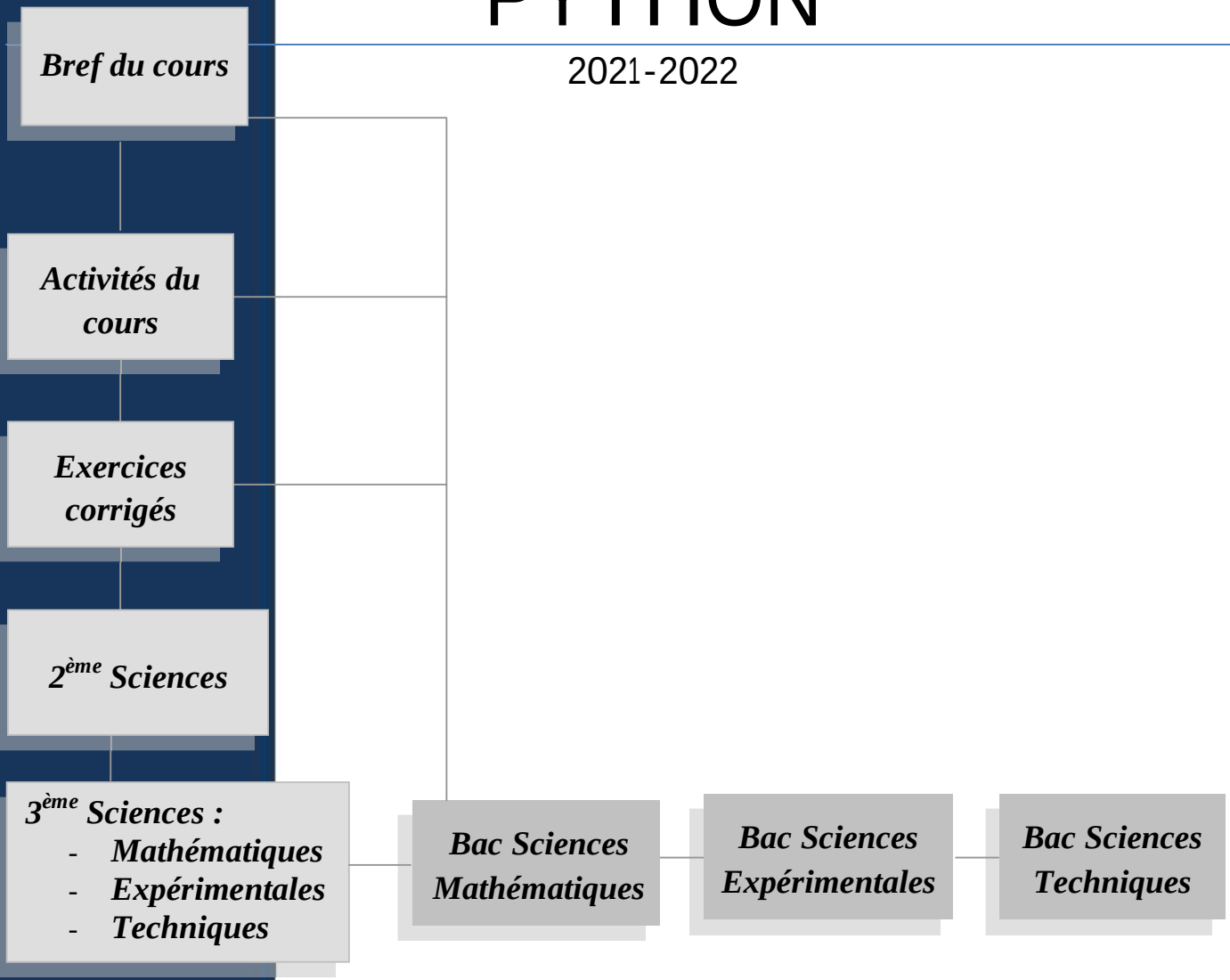
– LYCÉE SECONDAIRE 7 AVRIL 1943–MENZEL TEMIME-

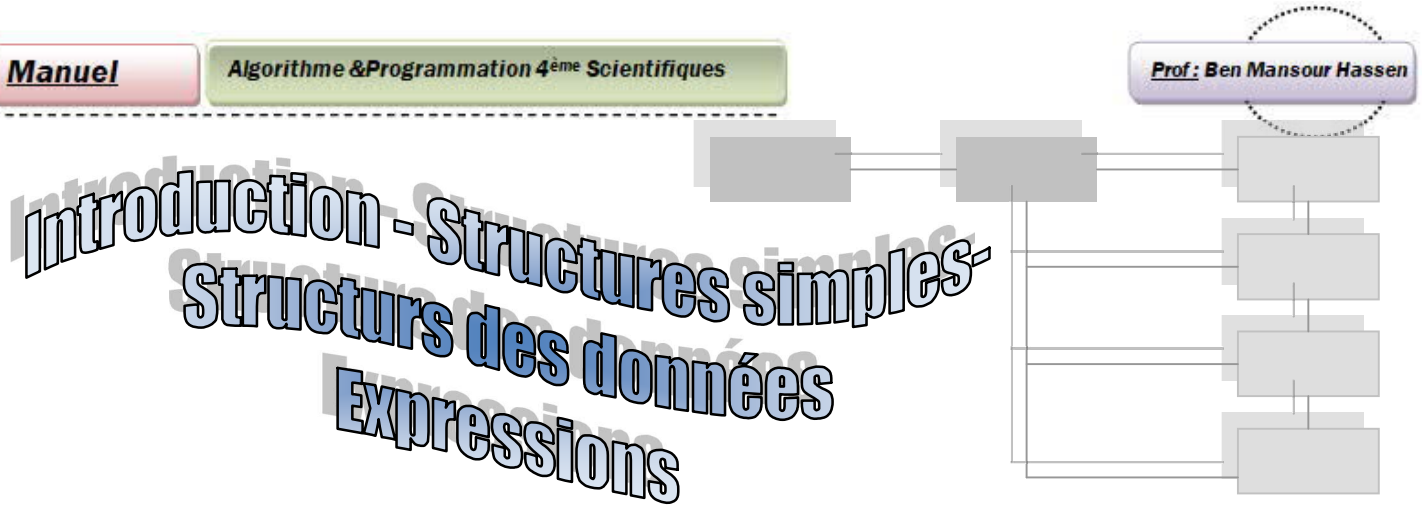
NOM & PRÉNOM & CLASSE

.....
.....
.....

Algorithme et programmation PYTHON

2021-2022





1- Notion de programme / algorithme

1.1 Définition d'un algorithme

……

……..

1.2 Définition d'un programme

……

……

En programmation il existe 3 types d'actions (écriture, lecture et affectation)

✧ L'action de lecture

Cette action n'est autorisée que sur tous les variables de types ordinaires sauf les booléens.

	En Algorithme	En Python
Lecture d'une variable x	Écrire ('commentaire') Lire (x)	x= input('commentaire ')
Lecture d'un entier n	Lire(n)	n=int(input('Donner un entier')
Lecture d'un réel r	Lire(r)	r=int(input('Donner un reel')

Remarque : input est une fonction qui retourne toujours une chaine (str)

✧ L'action d'écriture

- Algorithme :

```
Ecrire (″texte″)
Ecrire (variable)
Ecrire (″texte1″, var1, ...)
Ecrire (expression)
```

- Python :

```
print (″texte″)    # affichage d'un texte
print (variable)    # affiche du contenu d'une variable
print (″texte1″, var1, ″texte2″, var2, ...)    # affichage mixte
print (expression)    # affichage du résultat de l'expression
```

Remarque :

- print par défaut retourne à la ligne
- En python print possède deux paramètres sep (définit le séparateur entre différents objets affichés par default ‘ ‘) et end (définit le caractère qui termine l’affichage par défaut ‘\n’)
  Exemple
  print(‘Mohamed’,’ali’) affiche Mouhamed ali \n
  print (‘Mohamed’, ’ali’, sep =’-‘,end=’#’) affiche Mohamed-ali#
- On peut formater l’affichage d’une variable en utilisant la syntaxe suivante :

- Affichage normal d’un réel

```
>>> print(1.5)
1.5
```

- Affichage du même reel mais avec 8 chiffres après la virgule

```
>>> print("%8f"%1.5)
1.500000
```

- Affichage du même réel dans 10 cases dont 3 après la virgule

```
>>> print("%10.3f"%1.5)
     1.500
```

- Même rôle que le précédent en remplaçant ‘ ‘ avec 0.

```
>>> print("%010.3f"%1.5)
000001.500
```

Remarque : on peut utilise print(round(x,nb)) pour afficher un réel avec nb décimals.

✧ L’action d’affectation

Syntaxe

➢ Algorithme :

Nom_Variable ← Expression

Remarque

Cette action n’est pas autorisée sur les **constantes**. Elle permet à un ordinateur de remplacer l’ancien contenu d’une variable par :

➢ Python :

```
Nom_variable = Expression
Nom_variable = valeur
Nom_variable = nom_variable2
Nom_variable = nom_variable2=valeur
Nom_variable, nom_variable2 = valeur1, valeur2
```

2- Notion de langage de programmation

Étant donné qu’un ordinateur ne connait que le langage ……. ou le langage ……, on doit connaitre ce langage, mais puisque ce dernier est difficile à manipuler, on va utiliser des logiciels qui vont jouer le rôle d’un traducteur entre l’utilisateur et une machine. Ces logiciels utilisent une langue plus facile à comprendre pour nous qui sont appelés …….. comme C, C++, Java, Pascal et Python puis corriger le code écrit sous ce langage par l’opération de …… ou …… et le traduire en langage machine compréhensible par l’ordinateur .

3- Notion d'objet

a- En algorithme

Un algorithme est une suite finie d'actions écrites qui manipule des …….. Ces …… peuvent êtres des …….. (……..) ou bien des ……..(……)

Chaque objet **doit** posséder un ***nom*** (qui *l'identifie* dans le programme donc doit être unique), un ***type*** et une ***valeur.***

Une ***variable*** est identifiée par son *nom* et son *type* alors qu'une constante n'est identifiée que par son *nom*

Remarque :

- En algorithme, les noms des objets doivent êtres composés des lettres (majuscule et minuscule), chiffres et '_' non accentués (sans accents), non espacés (sans espaces)
- Il doit commencer toujours par une lettre.
- Un nom de variable ne peut pas être un mot clé du langage.

b- En Python

- En python tout est objet
- Tout objet possède un nom (*identificateur*)
- les noms des objets doivent êtres :
  - Composés par des lettres (majuscules, minuscules),chiffres,'_'
  - Doit être non espacé.
  - Ne doit pas commencé par un chiffre
  - Les mots-clés du langage ne peuvent pas êtres utilisées comme des noms pour les objets. (exemple for, import…)

Remarques

- il est préférable de choisir des noms significatifs
- il est préférable d'utiliser que des lettres minuscules pour les noms des objets sauf pour les classes (les noms des classes sont en majuscules)
- En python 3 les noms des objets peuvent êtres accentués (à éviter).
- ***Attention*** : Python est sensible à la casse c.à.d. il différencie entre majuscule et minuscule

4- Notion de type

En informatique, un objet est vu comme une plage mémoire dans une zone de données d'un programme.

Le **type** d'un objet définit la manière dont il est représenté en mémoire et les opérations que l'on peut faire dessus.

Par exemple on ne peut pas mettre 2 dans une variable de type caractère car 2 est de type entier.

5- Déclaration des objets.

En informatique tout objet doit être déclaré à la machine.
Cette déclaration se fait comme suit :

a- En Algorithme

À travers le tableau de déclaration des objets comme suit :

TDO

Objet	Type/Nature
Nom_constante	Constante=valeur
Nom_variable	Type_variable

b- En Python

Étant donnée que python utilise le typage dynamique donc l'étape déclaration n'est plus obligatoire au début du code.
Juste pour déclarer une variable on écrit son nom et on l'initialise par " = " (l'assignation / affectation)

Par exemple **a=2**

- Puisque python est un langage au typage dynamique, il a deviné que a est une variable entière (int)
- Python a alloué (réservé) l'espace en mémoire pour y accueillir un entier. Chaque type de variable prend plus ou moins d'espace en mémoire. Python a aussi fait en sorte qu'on puisse retrouver la variable sous le nom a.
- Enfin, Python a assigné la valeur 2 à la variable x.

6- Les types des données

a- En Algorithme

Il existe 5 types simples des données qui sont :

- Entier
- Réel
- Booléen
- Caractère
- Chaîne de caractères

Et d'autres types composés ou complexes (structures des données) :

- Tableau (à une ou à deux dimensions)
- Enregistrement
- Fichier …

b- En Python

Il existe plusieurs types

- int (integer)
- float(decimal)
- bool(boolean)
- str (string)
- complex
- none (non définit)
- list
- dictionnaire
- tuple
- class

Les types numériques

I- Les types simples

Nom Du Type		Opérations autorisées	Résultat
Algorithme	Python		
Booléen	Bool Vrai(True) Faux(False)		
Entier	int		
Réel	float		

Remarque *Python offre un autre type None (inconnu)*

Activité 1

Donner une instruction qui permet de :

- Afficher le message ″ Bonjour pour tout le monde ″

En Algorithme	En Python

- Afficher le contenue de la variable (b)

En Algorithme	En Python

- Afficher la moyenne (moy) de l'élève n° 1

En Algorithme	En Python

- Afficher 12.5 sous la forme de 10 chiffres dont 3 après la virgule

En Algorithme	En Python

Activité 2

- Compléter le tableau suivant :

Le nom d'une variable peut être comme suit	Algorithme			Python 3		
	Juste	Fausse	Justification (si fausse)	Juste	Fausse	Justification (si fausse)
From						
Numéro						
num eleve						
2ds						
Ds2						
_ds2						
Num_eleve						

- Est-ce qu'on pourra utiliser le type entier pour représenter les quantités suivantes ?

a. Nombre de jours de l'année ….. | c. Le résultat de 17 // 4 ……

b. Durée en heures d'une séance de cours ou TP ….. | d. Les résultats de 16/4. ……

Activité 3

Soit le programme Python suivant :

```
#Saisie de la variable n de type entier (int)
n=int(input("Donner un entier positif de 3 chiffres :"))
#Affecter à la variable c le chiffre de centaines de n
.....
#Affecter à la variable d le chiffre de dizaines de n
.....
#Affecter à la variable u le chiffre de unités de n
.....
#Affecter à la variable s la somme des chiffres de de n
.....
#Afficher la somme des chiffres de de n
print("La somme des chiffres de ",n," est :",s)
```

a- Compléter le programme ci-dessus par les affectations nécessaires pour afficher la somme des chiffres de l'entier **n**

b- Si on supprime int() de l'instruction numéro 1 du programme, que se passe t-il, expliquer brièvement pourquoi

……
……
……

Activité 4

Soit l'algorithme suivant :

```
Algorithme Inconnu
Début
Ecrire ("Donner un entier :"), Lire (a)
Ecrire ("Donner un autre entier :"), Lire (b)
aux ← a
a ← b
b ← aux
Ecrire ("a=",a,"et b=",b)
Fin
```

.....
.....
.....
.....
.....
.....
.....

Questions :

1) Écrire une implémentation en code Python de cet algorithme.
2) Exécuter le programme et donner l'affichage pour a=8 et b=3.

.....

3) Quel est le rôle de cet algorithme ?

.....

4) Quel est le rôle de la variable **aux** ?

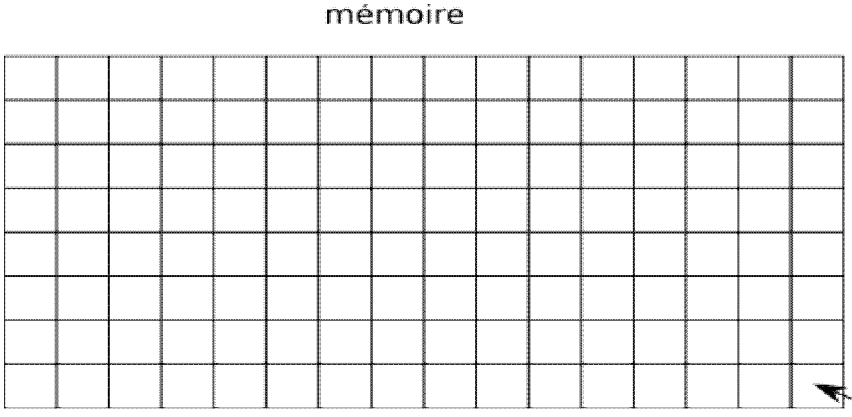
.....

5) Proposer une autre solution algorithmique sans utiliser une variable intermédiaire.

```
Algorithme Inconnu
Début
        Ecrire ("Donner un entier :"), Lire (a)
        Ecrire ("Donner un autre entier :"), Lire (b)

        .....

        .....

        .....

        Ecrire ("a=",a,"et b=",b)
Fin
```

N.B : En Python la permutation de deux variables a et b est : .....

6) Expliquer graphiquement cette permutation spéciale à Python



Activité 5

En utilisant **THONNY**, compléter le tableau suivant :

- Les fonctions arithmétiques standards : on doit importer le module math pour certaine fonctions (from math import *) x : paramètre (entier, réel) ; n : paramètre (entier)

SYNTAXE		RÔLE	Type du		EXEMPLE
Algorithme	Python		Paramètre	Résultat	
Ent (x)	trunc (x) floor(x)	....	…… …… …..	……..	Ent (3.14) =….. Ent (-3.14) =…..
Arrondi(x)	round(x)	....	…… …… …….	…….. …… …...	Arrondi (9.5) = ….. Arrondi (9.4) = ….. Arrondi(-9.5)=…..
Carrée(x)	square (x)	....	……	……...	Sqr(2) = 4
Racine_carré(x)	sqrt (x)	....	…….	……...	Sqrt (4)=….
Sin (x)	sin(x)	Donne le Sinus de x (x en radian)	…….	……... …	Sin (π /2) =1
Cos(x)	cos(x)	Donne le cosinus de x (x en radian)	……	……... …	Cos(π/2)= 0
Abs(x)	fabs(x) ou abs(x)	....	…….	…….. …	Abs(1)=…..Abs (-3)=….
Aléa	random()	....		……..	…….
Aléa(vi,vf)	randint(vi,vf)	....	……	……..	Aléa(100,200)=……

Remarque : en Python, pour que les fonctions **random** et **randint** nous donne des résultats il faut importer le module random (from random import *)
Il existe **uniform**(vi,vf) qui donne aléatoirement un réel entre vi et vf

Activité 6

Affecter aléatoirement à une la variable a un entier de deux chiffres

Affecter aléatoirement à un variable b un entier de maximum 3 chiffres

Affecter aléatoirement à un variable b un réel entre 5 et 10

Activité 7

◆ Écrire l'algorithme d'un programme qui :

- Saisit aléatoirement un entier positif **n** de deux chiffres,
- Affiche la somme des chiffres de cet entier.

◆ Écrire une implémentation en code Python de cet algorithme.

Algorithme	Code Python
Algorithme Somme **Début** .... **Fin** **Déclaration des objets :**	.....

Objet	Type/Nature

Activité 8

Écrire un algorithme et l'implémentation Python d'un programme intitulé CONVERSION qui saisit, automatiquement, une durée T en secondes ($10000 \leq T \leq 36000$) et la traduit en heures, minutes et secondes puis affiche le résultat.

Ex : si T= 12360 le programme affichera ″ 12362 est convertit en 3 h : 26 min : 2 s ″

Algorithme	Code Python
Algorithme Somme **Début** .... **Fin** **Déclaration des objets :**	.....

Objet	Type/Nature

Activité 9

Écrire un algorithme et son implémentation Python qui permet de saisir, automatiquement, un entier N de 4 chiffre puis afficher sa symétrie.

Ex : si N=1425 → sa symétrie est 5241

Algorithme	Code Python
Algorithme Somme **Début** .... **Fin** **Déclaration des objets :**	....

Objet	Type/Nature

Activité 10

Donner, en justifiant le choix des types, la traduction en Python de l'algorithme suivant :

```
Algorithme Exercice
Début
    Lire(x, y)
    r←x + carré(y)
    y←r mod x
    n←racine_carré(r) + y
    Ecrire (n)
Fin
```

Implémentation	Type	Justification
……	……	……
……	……	……
……	……	……
……	……	……...
……	……	……
……	……	……
……	……	……
……	……	……

Activité 11

Soit le programme Python suivant :

```
#..... ……..
import math as mt
Nom=cercle
#Déclaration ……..}
r=input(Donner r : )
r ←2,5;
air = =mt.sqr(r) X π
upper(nom)
input(nom ,″ possède comme air ″, air)
```

Correction	Justification

1- Compléter les commentaires par ce qui convient

2- Corriger, en justifiant vos réponses, les erreurs du programme ci-dessus.

Le type caractère et le type chaine de caractères

Activité 12

Nom Du Type		Domaine des valeurs	Opérations autorisées	Résultat
En Algo	En Python3			
Caractère	n'existe pas	Cas particulier d'une chaine	Comparaison (>,>=, <, <=, !=,= =)	Booléen
			Appartenance (Dans)	Booléen
Chaine de caractère	Str (string)	Ensemble de caractères	Comparaison (>,>=, <, <=, !=,= =)	Booléen

◆ Soit **x** une constante = "9", **b** une constante ="h"et **m** une constante = "H"
Ordonner le caractère "m" avec **x, b** et **m.**

……

◆ Compléter le tableau suivant **(Les fonctions/opérateur applicables sur le type caractère)**

Nom en Algo	Code en Python	Rôle	Exemples
ORD(c)	**ORD(c)**	……	**ORD("B") vaut ….**
CHR(n)	**CHR(n)**	……	**CHR(97) vaut ….....**

Activité 13

1) Écrire l'algorithme d'un programme Qui affiche aléatoirement une lettre majuscule.
2) Écrire une implémentation en code Python de cet algorithme.

Algorithme	Code Python
Algorithme Affichage **Début** …… …… **Fin** **Tableau de déclaration des objets :**	…… …… ……

Objet	Type/Nature

Type chaine de caractère

Rappel de cours :

Le type chaîne de caractères (str)

Une donnée de type chaîne est une suite quelconque de caractères :

- Les lettres majuscules de **A** à **Z**
- Les lettres minuscules de **a** à **z**
- Les chiffres de **0** à **9**
- Les ponctuations et les symboles

Accès aux caractères d'une chaîne :

Python offre des mécanismes permettant d'accéder séparément à chacun des caractères d'une chaîne. L'accès aux caractères se fait par index.

Exemple :

Ch =	"B	o	n	j	o	u	r"
Indice positif	0	1	2	3	4	5	6
			Ou bien				
Indice négatif	-7	-6	-5	-4	-3	-2	-1

- Ch[0] = "B" et Ch[3] = "j"
- Ch[-7] = "B" et Ch[-4] = "j"

◆ Soit la chaîne ch = 'Longue'

- Afficher les caractères d'indices respectifs 0 et 3 dans ch.

.....

- Afficher les caractères d'indices respectifs -1 et -6 dans ch.

.....

- Essayer de remplacer le caractère 'o' par 'a'.

.....

.....

Remarque

Les chaînes de caractères sont immuables (on ne peut pas changer leurs valeurs). Il est donc, interdit d'écrire par exemple : ch[0]='a'

◆ Comparer les chaînes suivantes :

- ch1 = ″Famille ″ et ch2=″ Family ″ …….
- ch1= ″ famille ″ et ch2=″ Family ″ …….
- ch1= ″ 2 ″ et ch2= ″ 19 ″ …….

◆ Opérations sur les chaines de caractères

Python intègre de nombreuses fonctions qui permettent d'effectuer divers traitements sur les chaînes de caractères (conversions majuscules/minuscules, recherche de mots, découpage, etc).

Algorithmique	Python	Rôle	Exemple
+	+	Permet la concaténation d'un ensemble de chaînes de caractères.	***Exemple 1***
Long(ch)	len(ch)	Retourne le nombre de caractères de la chaîne ch.	***Exemple 2***
Pos (ch1, ch2)	ch.find(s)	Retourne la première position de la chaîne **ch1** dans la chaîne **ch2**.	***Exemple 3***
Convch (x)	str(x)	Retourne la conversion d'un nombre x en une chaîne de caractères.	***Exemple 4***
Estnum (ch)	ch.isnumeric () ch.isdigit()	Retourne **Vrai** si la chaîne ch est convertible en une valeur numérique, elle retourne **Faux** sinon.	***Exemple 5***

Valeur (ch)	int(ch)	Retourne la conversion d'une chaine ch en un entier, sinon elle provoque une erreur.	***Exemple6***
	float(ch)	Retourne la conversion d'une chaine ch en un réel, sinon elle provoque une erreur.	***Exemple7***
Sous_chaine (ch, d, f)	ch [d:f]	Retourne une partie de la chaîne ch à partir de la position d jusqu'à la position f (f exclue).	***Exemple8***
Effacer (ch, d, f)	ch=ch [:d]+ch [f:]	Efface des caractères de la chaîne ch à partir de la position d jusqu'à la position f (f exclue).	***Exemple9***
Majus (ch)	ch.upper()	Convertit la chaîne ch en majuscules.	***Exemple10***

Exemple 1 :

ch1 = 'Bon' et ch2 = 'jour'

ch = ch1 + ch2 donne : .....

Exemple 2 :

ch = 'Code Python'

L1 = len(ch) donne : .....

L2 = len('ch') donne : .....

Exemple 3 :

ch1 = 'Bonjour' ; ch2 = 'jour' ch3 = 'bon'

p1 = ch1.find(ch2) donne : .....

p2 = ch1.find(ch3) donne : .....

p3 = ch1.find('o') donne : .....

Exemple 4 :

n = 2020 et ch = "Année "

msg = ch + n donne : .....

.....

Exemple 5 :

Ch1= "2020" et ch2 = "14.25 "

int (ch1) donne : .....

int (ch2) donne : .....

Exemple 6 :

Ch1= "2020" et ch2 = "14.25 "

float (ch1) donne : .....

float (ch2) donne : .....

Exemple 7 :

ch = 'Python'

ch*3 donne : .....

Exemple 8 :

ch = 'Python'

ch.isupper () donne : .....

Exemple 10 :

ch = 'Python'

ch. upper () donne : .....

Activité 14

✧ Compléter le tableau d'ordre de priorité

Opérateur		Ordre de priorité
En algorithmique	En Python	

Remarque :

Si deux opérateurs de même priorité se succèdent, la propriété s'accorde à l'opérateur placé à gauche

✧ Compléter le tableau suivant (Table de Vérités)

X	Y	NON (X)	X ET Y	X OU Y	X OUex Y
Faux	Faux				
Faux	Vrai				
Vrai	Faux				
Vrai	Vrai				

✧ Évaluer les expressions arithmétiques suivantes :

- 17 DIV 5
- 17 MOD 5
- ((58 DIV 7) MOD 2) + 5
- (49 MOD 17) DIV (4 * 3)
- 45+5*5-3
- 2/4+3*4 MOD 3
- (10<5) Et (8>0)
- (Arrondi (7.5)=7) Ou (RacineCarre (2)>1)
- (Ent (5.25)=5) Et (Non (sin (Pi)>1))

✧ Compléter les tableaux suivants

A	b	Expressions	Résultats
Faux	Vrai	((a et b) ou b) Ouex a	
13	Faux	Non(a >= 5 et b)	
2	-2	((a<b ou (a<=2)) et (b div 3 >=0)	
1	2	(a-b>=0) et non((a+b)<2*a)	
31	7	Non(((a mod b) =3) ou (-12<=-21))	
18.5	2	Arrondi(a+b)	
6.98	-5	Ent(a-b)*abs(b)	
4	3	A+carré(a)-racine_carré(a))+b	

Expressions	Résultats
Y ← Chr(26*Long(″bac″))	Y =
X← Convch(Ord(″F″)-3)	X =
G ← Ord(Chr(100))	G =
Ch ← ″Langages″ ; Y ←Pos(Ch,″Langage de programmation″)	Y =
Valeur(″21-11-05″)	

II- Les types composés (les types personnalisés) (les structures des données)

1- Les tableaux à une ~~ou deux dimensions~~

A. *En algorithme*

i- Définition

Une variable de type tableau est……
……..

ii- Déclaration

- 1ère méthode

Objet	Type/Nature
Nom_tableau	Tableau de nmax de type_el
~~Nom_tab_2_dim~~	~~Tableau de nb_ligne*nb_colonne de type_élément~~

Nmax=nombre maximale de case

Type_elt=type des éléments du tableau

~~nb_ligne, nb_colonne sont respectivement les nombres des lignes et les nombres de colonne de la matrice Nom_tab_2_dim~~

- 2ème méthode

En utilisant un nouveau type (création d'un nouveau type)

TDNT

TYPE
Nom_type=tableau de nmaxde type_elt
~~Nom_type2=Tableau de nb_ligne*nb_colonne de type_élément~~

TDO

Objet	Type/Nature
Nom_tableau	Nom_type
~~matrice~~	~~Nom_type2~~

iii- Accès à un élément d'un tableau

En algorithme les tableaux possèdent des indicent qui commence toujours par 0
Pour accéder à l'ième case du tableau, on écrit :

```
nom_tableau[i-1]
```

B. *En Python*

i- *Définition*

La notion de tableau dans Python est un peut différente par rapport à celle de l'algorithme.

En effet, un tableau en python est un ensemble structuré d'éléments (mais qui peuvent êtres de types différents et même d'un nombre très grand (on parle des milliards d'éléments))

En général on représente un tableau à une dimension dans python par une liste ~~et une matrice (tableau à deux dimensions) par une liste de liste~~.

ii- *Déclaration*

Python offre plusieurs méthodes pour déclarer une liste :

- 1ère méthode (création d'une liste vide)

```
tableau=[ ] ou bien tableau=liste()
```

- 2ème méthode (création d'une liste à partir d'un autre itérable (chaine de caractère, le résultat de la fonction range,une autre liste…)

```
tableau=list(itérable)
```

Remarque
On peut utiliser la fonction split du module str pour créer une liste à partir des mots d'une phrase
liste=phrase.split()

- 3ème méthode (création explicite de la liste)

```
tableau=[v1,v2,….vn]
```

- 4ème méthode création d'une liste à un nombre d'élément fixé en avance

```
tableau_zero=[0]*nb
```

Remarque ~~Dans le cas d'une matrice on utilise une liste de liste.~~

Activité 15

En utilisant **THONNY**, donner le résultat de chacune des instructions suivantes :

Instructions	Résultat (écran)
l=[ ]	
l= liste()	
l=[1,2,5]	
l=list(range(10))	
l=list(range(0,10,3)	
l=list('bonjour')	
l=list('bac science')	
l='bac science'.split()	
l='bac science'.split('c')	
l=[5]*3	
~~l=[[]]~~	Matrice hors programme pour cette année.
~~l=[[2]*3]~~	
~~l=[[1,2,3],[4,2,5]]~~	
~~l=[list(range(0,10,2),list(range(1,10,2))]~~	

iii- Accès aux éléments d'une liste

Les listes en python sont indicées de deux manières :

- Soit **croissante** de (0 vers le dernier élément)
- Soit de manière **décroissante** de -1 (dernier élément) vers le premier élément :

Si t un tableau de n éléments on obtient des indices comme suit :

Indice positif	*0*	*1*		*n-2*	*n-1*
t=	**14**	**5**	**…..**	**0**	**10**
	-n+n	*-n + (n-1)*	…..	*-2*	*-1*

Indice négatif

Pour accéder à une valeur d'une liste on écrit nom_liste[indice]

Remarque : python offre la possibilité d'accès à plusieurs éléments d'une liste en utilisant la technique de tranche ou (Slicing)

En générale non_liste[indice_debut:indice_fin] ou nom_liste[indice_debut:indice_fin,pas]

Ou :
- **indice_debut** est inclus (indice de debut)
- **indice_fin** est exclu (indice de fin)
- **Pas** c'est le pas pour se déplacer d'un indice à un autre)

Activité 16

Soit la séquence d'affectations suivantes :

T[0] ← 2
T[1] ← 10
T[2] ← T[0] * T[1] Div 6
T[3] ← 3* T[2] - T[1]
T[4] ← Pos("Bienvenu","venu") + 4
T[5] ← Long("Devoir")
T[6]←Valeur(Sous_Chaine("tu dois répondre correctement",8,9))
T[7] ← T[6] - T[2] Mod 30
T[9] ← T[4] -1

Déclarer le tableau T puis donner le contenue final de chacune de ses cases

Objet	Type/Nature	Rôle

Activité 17

En utilisant **THONNY**, donner le résultat de chacune des instructions suivantes :

Instructions	Résultat (écran)
l=list(range(2,20,3))	
l[:]	
l[1 :5]	
l[:5]	
l[2 :3]	
l[-1 :]	
l[-2 :]	
l[: :-1]	
l[-1 :-5 :4]	
l[-1 :-5 :-2]	
l[2]=0	
l[1 :5 :2]='98'	
l[1 :5 :2]=98	
l[1 :5 :2]=[9,8]	
l[1,5]=[0,7]	
l[2 :2]	
l[:-1]	

iv- Les fonctions prédéfinies sur les listes

Nom fonction	Thème	Rôle	Exemple
len(*liste*)	**Longueur d'une liste**	**Donne le nombre des éléments de la liste**	***Exp1***
liste.append(*elt*)	**Ajout des éléments**	**Ajouter *elt* dans liste**	***Exp2***
liste.insert(*indice*,*elt*)		Ajouter *elt* dans liste dans l'indice	
liste.extend(*liste2*)		Ajouter les éléments de la liste2 dans liste	
liste.remove(*elt*)	Suppression des éléments	Supprime la première occurrence d'elt ou erreur si elt n'existe pas	***Exp3***
del(*liste*[*indice*])		Supprime liste[indice] ou erreur si indice ne fait pas référence à un élément de la liste	
del liste[*debut* : *fin*]		Supprime toutes les éléments de la sélection de la liste entre debut (inclus) et fin (exclu)	
liste.pop(*indice*)		Même rôle que del Indice est par defaut -1	
liste.clear()		Supprime tous les éléments de la liste	
elt in liste	Recherche des éléments	True si elt existe False sinon	***Exp4***
liste.index(*elt*)		Le premier indice d'elt ou erreur s'il n'existe pas	
liste.count(*elt*)		Le nombre d'occurrence d'elt dans liste	
liste.sort()	Tri d'une liste	Tri croissante	***Exp5***
liste.sort(*reverse*=True)		Tri décroissante	
liste.reverse()	Inverser une liste	Inverser l'ordre d'apparition des éléments de liste	***Exp6***

Activité 18

En utilisant Thonny, donner le résultat de chacune des instructions suivantes :

Instructions	Résultat (écran)
l=[ ]	
l.append(1)	
len(l)	
l.append(list(range(3,10)))	
len(l)	
del(l[0])	
l.extend(list(range(3,10)))	
l.insert(1,2)	
l.append(0)	
l.pop()	
100 in l	
l.append(2)	
l.count(2)	
l.remove(2)	

Remarques

- Pour connaitre le type d'une variable python offre la fonction type() qui renvoie le type d'un objet donnée.

Il offre aussi la possibilité de transtypage :

- de texte vers entier (int (″ 3 ″) donne 3
- du texte vers float (float(″ 3 ″) donne 3.0
- d'un nombre (int ou float) vers texte str(123) donne 123

- En python les types sont divisés en deux grandes catégories les types mutables (modifiables) et les types immuables (non modifiables)) :

Types immuables	Types modifiables
Entier (int)	Liste
Flottant (float)	Dictionnaire
Complexe (complexe)	
Booléens (bool)	
Chaines de caratères (str)	
n-uplet(tuple)	

- En python si une affectation s'exerce sur une variable de type immutable, elle permet de créer une nouvelle variable du même nom.
- Une affectation modifie le contenue d'une variable mutable.

Les structures de contrôles

- ***Conditionnelles :*** permet à un ordinateur de choisir entre plusieurs traitement selon la valeur d'une condition

	Algorithme	Python
Structure simple réduite	[init] SI *condition* ALORS Traitement FIN SI	{init } if *condition:* Traitement
Structure simple complète	[init] SI *condition* ALORS Traitement1 SINON Traitement2 FIN SI	{init } if *condition :* Traitement1 else : Traitement2

Structure généralisée	[init] SI *condition1* ALORS
	Traitement1
	SINON SI *condition2* ALORS
	Traitement2
	:
	SINON SI *condition n* ALORS
	Traitement n
	SINON
	Traitement n+1
	FIN SI
		{init }
		if *condition1:*
		Traitement1
		elif *condition2*:
		Traiment2
		:
		elif *condition n*:
		Traitement n
		else :
		Traitement n+1 ;

Structure à choix multiple	[init]SELON *selecteur* faire Valeur1 : tratement1 Valeur2 : traitement2 : Valeur n : traitement n [Sinon traitement n+1] FIN SELON	

REMARQUE

- En Python, l'espace avant le traitement est appelé ***indentation*** c'est lui qui mentionne à la machine le traitement.
- [init] représente une éventuelle initialisation. [Sinon traitement n+1] cette instruction est facultative dans selon.
- Le *sélecteur* dans la boucle SELON doit être de type scalaire (entier, caractère, booléen, ~~scalaire énuméré~~).

- ***Itératives / répétitives***

Ces structures de contrôles permettent à l'ordinateur de répéter un traitement plusieurs fois.

Structure Itératives		Algorithme	Python	Condition d'utilisation
complète	POUR	[init]POUR *c* de *VI* à *VF* FAIRE Traitement FIN POUR	{init } FOR *c* in range(Vi,Vf): Traitement;	Lorsque le nombre d'itération est : - fini - connue à l'avance
À condition d'arrêt	Répéter.. Jusqu'à	[init] Répéter Traitement Jusqu'à *condition*	{init } While 1 : Traitement if *condition* : break	- Lorsque le nombre d'itération n'est pas connu à l'avance - Le traitement doit s'exécuter au moins une fois (nombre d'itération doit >0)
	Tant que .. Faire	[init] Tant que *condition* FAIRE Traitement FIN TANTQUE	{init }WHILE *condition* : TRAITEMENT	- Lorsque le nombre d'itération n'est pas connu à l'avance - Le traitement peut ne pas être exécuté (nombre d'itération peut être =0)

REMARQUE

- En Python, break permet de sortir de la boucle
- range est une fonction qui permet de générer des valeurs entière entre VI et VF
- init représente une éventuelle initialisation.
- *c* est appelé compteur (itérable), *VI* = valeur initiale et *VF* = valeur finale.il sont tous de même type (entier, caractère, booléen)
- Si *VI* < *VF*, on parle d'une incrémentation.
- Si *VF* > *VI* alors on parle d'une décrémentation le boucle pour devient :

Décrémentation	[init]POUR *c* de *VF* à *VI* FAIRE (pas = -1) Traitement FIN POUR	{init } FOR *c* in range(*VF,VF,PAS):* Traitement

- Le nombre d'itérations est égale à | *VI-VF* |+1

Les sous programmes

1- Introduction

1.1 Mise en situation

Pour la révision pour les examens de baccalauréat est ce qu'on peut résoudre toutes les matières ensembles,

Certainement la réponse est non, car la révision de toutes les matières au même temps c'est une tache est très difficile et complexe c'est pour cela, un élève doit décomposer cette tache à plusieurs taches ou modules plus faciles à manipuler d'où la résolution de ce problème devient comme suit :

0- Début revision_bac
1- Module Math
2- Module Langue
3- Module physique
4- Module informatique
5- Module option
6- Fin revision_bac

On remarque que chaque module on peut le décomposer en d'autres modules plus simple, par exemple le module des langues devient :

0 – module langue
1 – module arabe
2 – module français
3 – module anglais
4 – fin module Langue

1.2 Constatation

Par analogie, en programmation, devant un problème complexe, on doit utiliser une approche ***modulaire*** pour rendre sa résolution plus *simple* et plus *facile*. Donc on doit le décomposer en modules ***indépendants et plus simples***, ces modules peuvent être soit :

- Des fonctions qui ……..……..
- Des procédures qui ……..……....
- En Python ……

1.3 Remarques :

Pour exécuter un module, on doit le définir puis l'appeler.

Définition d'une fonction	
En algorithme	En Python
……	…….
…. ……	……
……	……
……..	……

Définition d'une procédure	
En algorithme	En Python
…… …… …… ……	…… …… …… ……

Appel d'une fonction	
En algorithme	En Python
soit : …… Soit : ……	Soit …… Soit ……

Appel d'une procédure	
En algorithme	En Python
……	……

Notion d'objets globaux / objets locaux

1.4 Mise en situation

Dans notre exemple de révision, les papiers et les stylos sont des objets utilisés dans tout le programme et tous ses modules c'est pour cela on les appelle variables globales.

Alors que, dans le module de l'informatique, on doit disposer outre des variables globales, un ordinateur par exemple, celui-ci est appelé alors variable local.

1.5 Constatation

En Programmation

Les objets globaux ……

……

Les objets locaux ……

……

Notion des paramètres formels / Paramètres effectifs

Les paramètres formels ……

Les paramètres effectifs ……

En python, les arguments ……..

……

Module appelant/ Module appelé.

Un module est dit appelant ……...

……...

Mode de passage des paramètres

Le ***dialogue*** entre les différents modules s'appelle mode de passage de paramètre. Il permet de préciser si un module va émettre ses modifications sur ses paramètres au module appelant ou non. Il existe deux modes de passage des paramètres :

1.6 Mode de passage par valeur :

Dans ce cas le module appelé ……

……

Les valeurs des paramètres sont toujours émises par le module appelant. On parle d'un seul sens de dialogue

Module Appelant → Donne les valeurs de ses paramètres → Module Appelé

1.7 Mode de passage par adresse ou par variable

Dans ce mode le module appelé ……... On parle d'un dialogue ***bidirectionnel***.

Dans ce cas on doit précéder, dans la définition des modules appelés, les paramètres concernés par la modification par le mot clé **VAR (@)**

Module Appelant ↔ Donne les valeurs de ses paramètres ↔ Module Appelé

Remarque :

……

……

……

TP Interface Homme Machine

(Programmation évènementielle à l'aide de PyQt et QTDesigner)

Mise en situation

Il est courant qu'un programme interagisse avec l'extérieur en échangeant des données en entrée et en sortie. Cependant il peut aussi échanger des événements, et déclencher des actions en fonctions d'autres actions extérieures. C'est le cas des sites Web dynamiques, qui réagissent pour changer l'état ou l'apparence de la page, par exemple lorsque l'on clique sur un bouton. La programmation événementielle se focalise ainsi sur l'exécution de code en fonction d'événements, plutôt qu'une simple exécution séquentielle des instructions.

Introduction

a- Définition IHM

IHM signifie **interface homme-machine** et fait référence à un tableau de bord qui permet à un utilisateur de communiquer avec une **machine**, un programme informatique ou un système.

b- Définition Programmation évènementielle

Ce paradigme est fondé sur les événements qui se produisent dans le programme. En d'autres termes, l'exécution du programme sera déterminée par ce qui se produit à un instant donné.

QT Designer

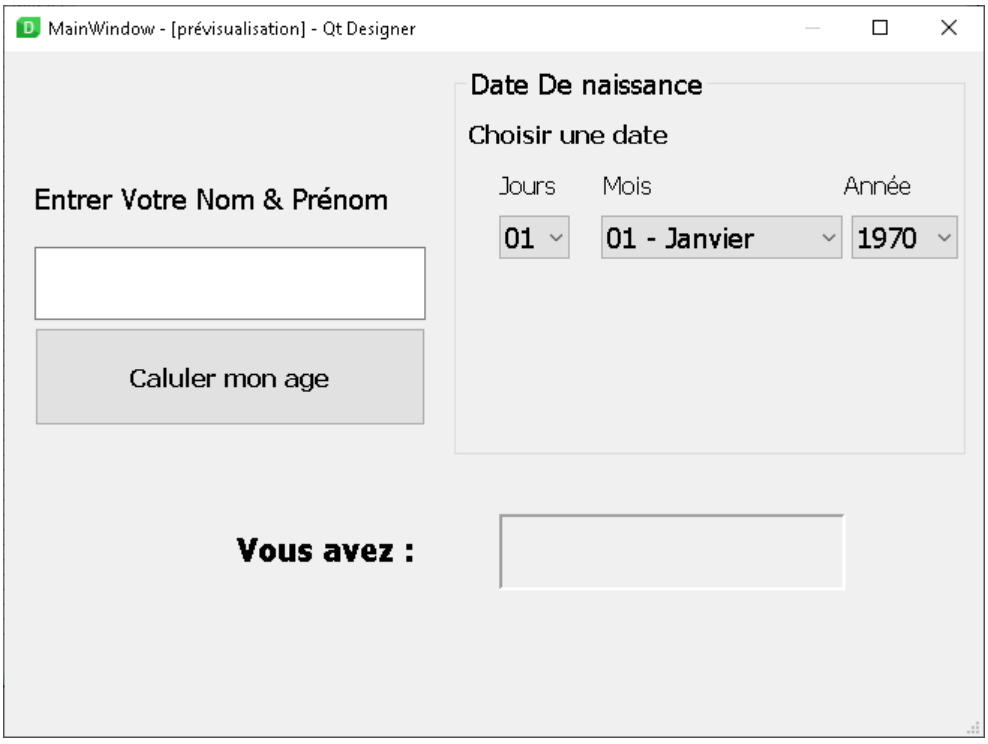
Qt Designer est l'outil de Qt pour la conception et la construction d'interfaces utilisateur graphiques (GUI) à partir de composants Qt. Vous pouvez composer et personnaliser vos widgets ou boîtes de dialogue à la manière de ce que vous voyez est ce que vous obtenez (WYSIWYG) et les tester en utilisant différents styles et résolutions.

Activité 1

Établir une interface qui permet d'entrer un entier puis afficher s'il est premier ou non

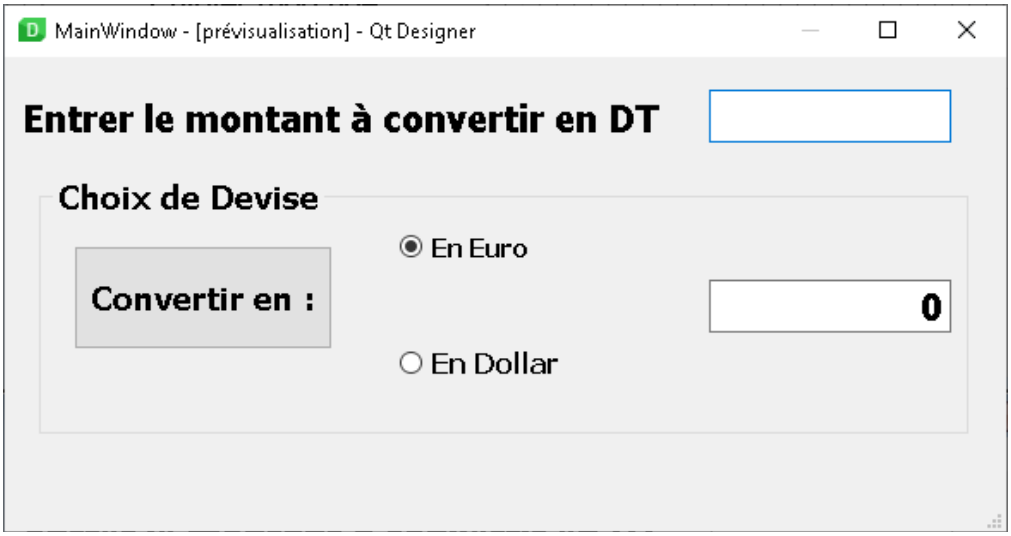
Activité 2 :

On veut créer une interface graphique qui permet d'introduire le nom et prénom d'un utilisateur et de sélectionner sa date naissance puis affiche son âge.



Activité 3 :

Établir une interface qui permet de convertir un montant en DT soit en Euro soit En dollar Américain



Activité 4 :

Établir une interface qui permet de construire une calculatrice :