4ème

techniques
Sciences
Maths

Année Scolaire : 2023/2024

Cours Informatique

Enseignant : Lotfi Amama

Qu'est-ce qu'un algorithme

1) *Une recette de cuisine*

Voici une recette tirée d'un livre de cuisine
Se procurer 250 g de chocolat noir, 250 g de beurre, 4 œufs, 250 g de sucre et 75 g de farine.
a) Faire fondre le chocolat au bain-marie ; ajouter le beurre, mélanger ; ajouter la farine.
b) Battre les œufs en omelette ; ajouter le sucre et tourner le mélange.
c) Mélanger les deux préparations.
d) Verser dans un moule et faire cuire 45 minutes au four à 220°C.
Servir le gâteau froid.

Ce texte décrit les opérations à effectuer successivement pour faire un moelleux au chocolat. Il est formé de trois parties distinctes :
- les entrées : ce sont les ingrédients de la recette, avec les quantités requises.
- le traitement de la recette : il s'agit des phases a, b, c et d qui s'enchaînent séquentiellement.
- la sortie : c'est le gâteau fini, que l'on doit servir froid.

2) *Une construction géométrique*

On se donne deux points A et B du plan.
a) Tracer le cercle de centre A passant par B.
b) Tracer le cercle de centre B passant par A.
c) Nommer C et D les points d'intersection de ces cercles.
d) Construire le polygone ADBC.

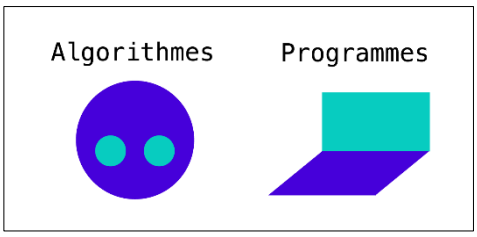
Cet algorithme décrit la construction d'un losange dont une diagonale est [AB].
Les entrées sont les points A et B.
Le traitement de la construction est décrit dans les phases a, b, c.
La sortie est le polygone ADBC.

Un algorithme est d'une suite d'instructions permettant de donner une solution à un problème. Il comprend :

- une phase d'initialisation : on entre les données ;
- une phase de traitement du problème ;
- une phase de sortie des résultats.

Différence entre un algorithme et un programme

Un algorithme doit être compréhensible par un humain, alors qu'un programme est écrit de façon à ce qu'il soit compréhensible par une machine.



C'est quoi une variable 

Si on revient à l'algorithme 1, on va faire une analogie avec la cuisine. . . En effet, lorsqu'on commence une recette, on prépare les ingrédients et on va les « stocker » dans des récipients différents : un bol pour le lait, une passoire pour les pommes de terre lavées, . . .

En informatique, il en est de même, les informations utiles au fonctionnement d'un algorithme ou d'un programme sont stockées dans des variables qui ne sont pas toutes du même type et à qui on doit donner des noms.

Retenons :

Une variable est un espace de la mémoire qui contient une (ou plusieurs) information(s). Un variable possède :

- **Un identificateur** : c'est son nom ; par exemple a, age, maVariable, . . .
- **Un type** : nombre entier, nombre décimal, Texte, . . . (En Python, on ne précise pas le type lorsqu'on définit une variable) ;
- **Un contenu** : ce que contient l'espace en mémoire.

Remarque :

Le nom d'une variable est une suite de caractères (lettres non accentuées, chiffres et le caractère _) dont le premier est obligatoirement a une lettre.

Application 1

Parmi les propositions suivantes, barrer celles qui ne peuvent être des noms de variables :

a | aB | a23 | a% | 5ps | a-b | z^k | aa_dd | âge

Quelle est la structure d'un algorithme 

ALGORITHME Nom
DEBUT
Traitement1
.....
TraitementN
FIN

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**

Quelles sont les opérations élémentaires 

L'opération d'entrée

Notation Algorithmique	Notation en Python
Ecrire ("Commentaire") **Lire** (Objet)	*Pour les chaînes de caractères :* Objet = **input** ("Commentaire") *Pour les entiers :* Objet = **int** (**input** ("Commentaire")) *Pour les réels :* Objet = **float** (**input** ("Commentaire"))

L'opération de sortie

Notation Algorithmique	Notation en Python
Ecrire ("Message", Objet) **Ecrire** ("Message", Expression)	**print** ("Message", Objet, **end** = "") **print** ("Message", Expression, **end** = "")
Ecrire_nl ("Message", Objet) **Ecrire_nl** ("Message", Expression)	**print** ("Message", Objet) **print** ("Message", Expression) ***N.B. :*** *"print" fait un retour à la ligne automatique*

N.B. : Objet est de type simple.

L'opération d'affectation

Notation Algorithmique	Notation en Python
Objet ← Valeur Objet ← Expression Objet1 ← Objet2	Objet = Valeur Objet = Expression Objet1 = Objet2

N.B. : Objet1 et Objet2 doivent être de même type ou de types compatibles.

Situation1

On pige au hasard, sans remise et sans tenir compte de l'ordre, 2 billes d'un sac contenant une bille rouge (R), une bille bleue (B), une bille jaune s(J) et une bille verte (V). Combien y a-t-il de résultats possibles ?

$$\textbf{Nombre de combinaisons possibles} = \frac{n!}{p! * (n-p)!}$$

n: Nombre d'éléments dans l'ensemble de départ
p: Nombre d'éléments sélectionnés dans l'ensemble de départ

Ecrire un algorithme qui permet de saisir deux entiers n et p (**0<p≤n**) puis calculer et afficher la valeur C_n^p.

- Quelles sont les données de ce programme ? .....
- Comment peut-on faire un contrôle de saisie sur les valeurs de p et n ?

.....

Retenons :

	Algorithme	Python
Répéter	**Répéter** Traitement **Jusqu'à (condition)**	**Initialisation** **While** not(Condition) : Traitement
TantQue	**TantQue** (Condition) **Faire** Traitement **Fin TantQue**	**While** condition : Traitement

Exemple1

Saisir une moyenne comprise entre 0 et 20.

....

....

....

- Préciser le type de la condition ?

.....

Exemple2

Le résultat de 14 > 5 est VRAI

celui de 14 < 5 est FAUX

et celui de 5 = 9 est FAUX

Le résultat de (14 > 5) ET (5 < 3) est FAUX

car (14 > 5) = VRAI et (5 < 3) = FAUX

Retenons :

Domaine de définition	Vrai et Faux	Exemples
Opérateurs Logiques	Notation algorithmique	Notation en Python
	Non (Négation)	NOT
	Et (Conjonction)	AND
	Ou (Disjonction)	OR
	Ouex (Ou exclusif)	XOR

Application 1

Compléter les tables de vérités suivantes :

Valeur de l'expression Logique A	Valeur de l'expression Logique B	Non(A)	A ET B	A OU B	A OUex B
Vrai	Vrai				
Vrai	Faux				
Faux	Vrai				
Faux	Faux				

Application 2

Sachant que a = 4, b = 5, c = 1 et d = 0, évaluer les expressions logiques suivantes :

1. (a < b) et (c >= d)
2. Non (a < b) ou (c ≠ b)
3. Non (a ≠ b^2) ou (a * c < d))

Remarque :

Lors de l'évaluation d'une expression, il faut tenir compte des règles de priorité entre les opérateurs utilisés.

Notation		Type d'opérande	Priorité
Algo	Python		
(....)	(....)	Tous les types	1
*	*	Entier/ réel	2
/	/	Réel	
Div	//	Entier	
Mod	%	Entier	
+	+	Entier/ réel	3
-	-	Entier/ réel	
=	==	Tout type ordonné	4
≠	!=	Tout type ordonné	
> ,>=	>,>=	Tout type ordonné	
<	<,<=	Tout type ordonné	

Notation	
Algorithmique	**Python**
X ∈ {v1,v2 ,.....,vn}	**X** in {v1,v2 ,.....,vn}
X ∈ [v1..vn} ou bien v1<=**x**<=vn	**X** in range (v1,vn) ou bien V1<=**x**<=vn

Application 3

Exécuter manuellement ces instructions.

x←8 **répéter** x←x+2 y←x*2 **jusqu'à** y>25	p←0 **Tant que** p <5 **faire** p←p+2 **Fin Tant que**	p←0 **Tant que** p> 5 **faire** p←p+5 **Fin Tant que**	x←5 **répéter** x←x+2 k←x*2 **jusqu'à** k <30
x= y=	p=	p=	x= k=
Nombre d'itérations :	Nombre d'itérations :	Nombre d'itérations :	Nombre d'itérations :

Situation2

Ecrire un programme permettant de simuler d'une façon simple le fonctionnement d'un distributeur automatique bancaire :
Saisir un montant en dinars (Nombre entier strictement positif qui doit contenir 0 à droite pour simplifier le problème).
Décomposer le montant saisi en billets de 20 dinars et des billets de 10 dinars.

Exemple n°1 : Montant saisi= 150, le programme affiche : 7 Billet(s) de 20 DT et1 Billets de 10 DT
Exemple n°2 : Montant saisi= 300, le programme affiche : 15 Billet(s) de 20 DT et 0 Billets de 10 DT.

Pour résoudre ce problème, on a besoin des opérateurs qui calculent le reste de la division ainsi que le quotient.

Retenons :

Algorithme	Python
div	//
mod	%

Application 1

Soit la séquence d'instructions suivante :

X ←5
Y←X/2
Z ← X MOD 2
W ← X + Y
T ← Y DIV 3

Questions :

- Présenter le tableau de déclaration des objets utilisés dans cette séquence.
- Lors de l'écriture de ces instructions une erreur a été commise, donner l'instruction qui comporte cette erreur puis proposer une solution pour la corriger.

Application 2

Saisir un entier n composé de 3 chiffres puis calculer et afficher la somme de ces chiffres. (2 méthodes) **Exemple** : n=123 → Somme=6.

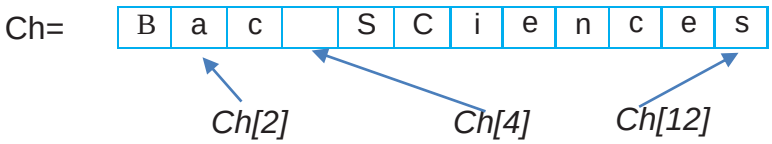
Application 3

Saisir un entier n composé de 4 chiffres puis l'inverser. (2 méthodes) **Exemple** : n=1234 → Inv=4321

Retenons :

Une chaîne de caractères est un regroupement de **n** caractères. **n** étant comprise entre 0 et 255. Si n est nulle, on parle d'une chaîne vide.
Les caractères relatifs à une chaîne sont représentés entre guillemets.

On peut accéder au **i**ème caractère d'une chaîne **ch** en utilisant la notation **ch[i]** avec 1≤ i ≤ Long (ch).



Les fonctions sur les chaînes de caractères

Les fonctions sur le type chaîne de caractères				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
Lo ← **long** (Ch)	Lo = **len** (Ch)	Retourne un entier représentant le nombre de caractères de la chaîne **Ch** (la longueur de Ch).	Lo = len ("Salut") Lo = len ("L'élève") Lo = len ("")	Lo == 5 Lo == 7 Lo == 0
Po ← **pos** (Ch1, Ch2)	Po = Ch2.**find** (Ch1)	Retourne un entier représentant la position de la 1ère occurrence de **Ch1** dans **Ch2**. Elle retourne -1 si **Ch1** n'existe pas dans **Ch2**.	Ch1 = "bo" Ch2 = "bonbon" Po = Ch2.find (Ch1)	 Po ==
Ch2 ← **sous_chaîne** (Ch1, Début, Fin)	Ch2 = Ch1 [Début : Fin]	Retourne une copie de la chaîne **Ch1** à partir de l'indice **Début** à l'indice **Fin** (position **Fin** exclu).	Ch1 = "BACCALAUREAT" Ch2 = Ch1 [5 : 12] Chr == "LAUREAT"	
Ch2 ← **effacer** (Ch1, d, f)	Ch2= Ch1[: d]+Ch1 [f:]	Retourne une chaîne **Ch2** après avoir effacer, de la chaîne **Ch1**, les caractères de la position **d** à la position **f** (**f** exclu).	Ch1 = "INFORMATIQUE" Ch2 = Ch1[:6] + Ch1[11:] Ch2 == "INFORME"	
ChM ← **majus** (Ch)	ChM = Ch.**upper** ( )	Retourne la chaîne **ChM** représentant la conversion en Majuscule de la chaîne **Ch**.	Ch = "Bonjour" ChM = Ch.upper ( )	ChM =="BONJOUR"
Ch ← **convch** (X)	Ch = **str** (X)	Retourne la conversion du nombre **X** en une chaîne de caractères.	N = 358 Ch = str (N)	Ch == "358"
Test ← **estnum** (Ch)	**Pas de correspondance.** Toutefois, on pourra utiliser **isnumeric** ( ) malgré qu'elle ne répond pas aux exigences ou bien développer un module qui permet de réaliser cette tâche.	Retourne **VRAI** si la chaîne **Ch** est convertible en une valeur numérique et **FAUX** dans le cas contraire.	Ch = "489" Test = Ch . isnumeric ( ) Ch = "489.56" Test = Ch . isnumeric ( )	Test == True Test == False
N ← **valeur** (Ch)	N = **int** (Ch) *ou bien* N = **float** (Ch)	Retourne la conversion d'une chaîne **Ch** en une valeur numérique, si c'est possible.	Ch = "489" N = int (Ch) Ch = "489" N = float (Ch)	N == 489 N == 489.0

Application 4

Compléter le tableau suivant :

Instruction	Résultat	Type du résultat
A← 1+2		
B← ''1''+''23''		
C ← non (''b''≥''Baccalauréat'')		

Application 5

Saisir une chaine de caractères **ch** de taille impaire et strictement supérieure ou égale à 3, puis afficher la chaine contenant sa longueur, son premier caractère, le caractère du milieu ainsi que le dernier caractère.

Exemple : ch=''poste'' → Le programme affiche : ''5pse''.

Situation3

On veut écrire un programme permettant de coder un message selon le procédé suivant : permuter chaque caractère d'indice pair avec le caractère qui le précède.

Exemple : Le codage de la chaîne : ″Baccalauréat″ donne ″aBcclauaérta″

.

Retenons :

La boucle Pour … Faire …	
Notation en algorithmique	**Notation en Python**
Pour Compteur de **Début** à **Fin** [Pas= valeur_pas] **Faire** Traitement **FinPour**	**for** Compteur **in range** (Début, Fin+1, Pas) : Traitement
Remarques : • Le nombre de répétitions est connu avant le traitement et il est égal à |**Fin – Début** |+ 1. • Le **Pas** peut être **Positif** ou **Négatif**. • Par défaut, le **Pas** est égal à **1**. • Il faut éviter de modifier la valeur du compteur de la boucle **Pour… Faire…** au niveau du traitement.	• En python, on accepte seulement la forme décrite précédemment. • **range** (5) *le compteur prendra les valeurs suivantes : 0 , 1 , 2 , 3 , 4* • **range** (2, 5) *le compteur prendra les valeurs suivantes : 2 , 3 , 4* • **range** (5, 2, -1) *le compteur prendra les valeurs suivantes : 5 , 4 , 3* • **range** (0, 10, 3) *le compteur prendra les valeurs suivantes : 0 , 3 , 6 , 9*

Application 1

Saisir une chaîne numérique puis calculer la somme de ses chiffres.

Exemple : ch=''123456'' → S=21

Application 2

Saisir une chaîne ch de taille>1 puis l'inverser et l'afficher.

Exemple : ch=''123456'' → Inv= ch=''654321''

Application 3

Saisir une chaîne **ch** de taille paire (supposée alphabétique) puis afficher le nombre de voyelles et la chaîne de consonnes.

Exemple : **ch**=''anniversaire'' → Nbvoy=6, chcons= ''nnvrsr''.

Situation4

On pige au hasard, sans remise et sans tenir compte de l'ordre, 2 billes d'un sac contenant une bille rouge (R), une bille bleue (B), une bille jaune (J) et une bille verte (V). Combien y a-t-il de résultats possibles ?

$$\textbf{Nombre de combinaisons possibles} = \frac{n!}{n! * (n-p)!}$$

- Ecrire un algorithme sans modules pour calculer la valeur de C_n^p.
- Ecrire un algorithme en utilisant les modules pour calculer et afficher la valeur de C_n^p.

Retenons :

Notation algorithmique

```
FONCTION Nom_fonction ( Pf1 : Type1 , … , Pfn : Typen ) : Type_Résultat
DEBUT
        Instruction1
        InstructionN
        Retourner  Résultat
FIN
```

Notation en Python

```
def  Nom_fonction ( Pf1 , … , Pfn ) :
     Instruction1
     InstructionN
     return   Résultat
```

Remarque :

- La fonction retourne un seul résultat (Entier, Réel, Booléen, Caractère ou Chaîne).
- Les paramètres effectifs (Pe1 à Pen) et les paramètres formels (Pf1 à Pfn) doivent s'accorder de point de vue ordre, nombre et type.
- Une fonction possède un type qui est celui de son résultat.
- Le résultat d'une fonction est une valeur qui peut être affecter dans une variable ou bien la faire figurer dans une expression ou bien l'afficher directement.

Application 1

Saisir un nombre composé de 2 chiffres, calculer et afficher la somme des factorielles de ces 2 chiffres.

Exemple : n=15 → S=1 !+5 !=1+120=121.

Application 2

Chaque élève prend une étiquette sur laquelle est écrit un entier strictement positif inférieur à 5000, on se propose de jouer par paire d'élèves le jeu suivant des nombres amis.

Travail demandé :

Ecrire un programme permettant de tester si deux entiers a et b saisis strictement positifs inferieurs à 5000 sont amis.

Exemple n°1 : Pour a=220 et b = 284 le programme affichera : 220 et 284 sont des nombres amis.

Somme des diviseurs propres de 220 : 1, 2,4, 5, 10,11, 20,22, 44,55 et 110 =284

Somme des diviseurs propres de 284 : 1,2, 4,71, et 142=220

Retenons :

Structure conditionnelle réduite simple : Syntaxe

Notation en algorithmique	Notation en Python
Si Condition **Alors** Traitement **FinSi**	**if** Condition : Traitement

Structure conditionnelle complète : Syntaxe

Notation en algorithmique	Notation en Python
Si Condition **Alors** Traitement1 **Sinon** Traitement2 **FinSi**	**if** Condition : Traitement1 **else :** Traitement2

Notation algorithmique

```
PROCEDURE Nom_procédure ( Pf1 : Type1 , Pf2 : Type2 , … , Pfn : Typen )
DEBUT
        Instruction1
        Instruction2
        InstructionN
FIN
```

Notation en Python

```
def Nom_procédure ( Pf1 , … , Pfn ) :
        Instruction1
        Instruction2
        InstructionN
```

Remarque :

- Les paramètres effectifs (Pe1 à Pen) et les paramètres formels (Pf1 à Pfn) doivent s'accorder de point de vue ordre, nombre et type.
- L'appel d'une procédure est une instruction.
- Le passage de paramètre par adresse (par référence) permet au programme appelant (PP) de transmettre une valeur à la procédure appelée (SP) et vice versa.
- On ajoutera le symbole « @ » avant le paramètre formel passé par adresse.

En Python, pour résoudre le problème de passage par adresse, on peut suivre l'une des deux démarches suivantes :

La 1ère démarche :

1) Ne pas mettre les paramètres formels passés par adresse dans l'entête de la procédure.
2) Mettre les paramètres formels passés par adresse dans le corps de la procédure précédés du mot « **global** ».

La 2ème démarche :

1) Ne pas mettre les paramètres formels passés par adresse dans l'entête de la procédure.
2) Utiliser le mot « **return** » pour retourner les valeurs des paramètres formels passés par adresse (au niveau algorithme).
L'exemple ci-après illustre le passage par adresse en algorithmique et en Python.

Notation en algorithmique	
Déclaration de la procédure "Saisir"	**L'appel de la procédure "Saisir"**
Procédure Saisir (@ n : entier) Début Répéter Écrire ("Saisir un entier entre 5 et 20 : ") Lire (n) Jusqu'à (5 ≤ n ≤ 20) Fin	*Saisir* (n)
Notation en Python de la 1ère démarche	
Déclaration de la procédure "Saisir"	**L'appel de la procédure "Saisir"**
def Saisir ( ) : **global** n valid = False while valid == False : n = int (input("Saisir un entier entre 5 et 20 : ")) valid = (5 ≤ n ≤ 20)	*Saisir* ( )
Notation en Python de la 2ème démarche	
Déclaration de la procédure "Saisir"	**L'appel de la procédure "Saisir"**
def Saisir ( ) : valid = False while valid == False : n = int (input("Saisir un entier entre 5 et 20 : ")) valid = (5 ≤ n ≤ 20) **return** n	*n = Saisir* ( )

Application 3

Saisir un nombre composé de 2 chiffres, calculer et afficher la somme des factorielles de ces 2 chiffres.

Exemple : n=15 → S=1 !+5 !=1+120=121.

Application 4

Saisir un nombre n>10 puis affiche s'il est sublime ou non. Un nombre est dit sublime si le nombre de ses diviseurs et la somme de ses diviseurs sont parfaits.

Un nombre est dit parfait s'il égal à la somme de ses diviseurs sauf lui-même.

Exemple : 12 est un nombre sublime

Diviseurs : 1, 2, 3, 4, 6, 12

Nombre de diviseurs = 6 → parfait // Somme des diviseurs = 28 → parfait.

Application 5

Saisir n1 et n2 (n1>0, n2> 0 et n1≠n2) puis afficher s'ils sont jumeaux.

Deux nombres n1 et n2 sont dits jumeaux si n1 et n2 sont premiers et n2=n1+2

Exemple : 5 et 7 sont jumeaux.

Situation5

Saisir une chaîne ch de taille minimale 4, saisir aléatoirement un entier p (0<p<long(ch)-1), inverser la chaîne (caractère d'indice 1, caractère d'indice p), inverser la chaîne (caractère d'indice p+1, caractère d'indice long(ch)-1) puis concaténer les deux chaînes inversées et afficher la chaîne résultat.

Exemple : ***ch***=''abc12+59?'', ***p***=3, chaîne résultat : ''cba92+52? ''.

Retenons :

Alea est une fonction qui permet de saisir …… une valeur comprise entre une valeur initiale Vi et une valeur finale Vf.

Application 1

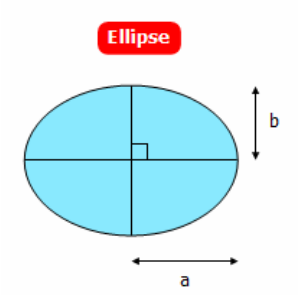
Lancer Thonny et remplir le tableau suivant :

Description	Algorithme	Python
…… ……	X ← aléa()	…… X = random() print (X) ……
Générer un nombre entier dans [val1, val2]	X ← Aléa (val1, val2)	…… X = …… (14, 30) print(X) ……

Les fonctions sur les types numériques				
Notation algorithmique	Notation Python	Rôle	Exemples en Python - Résultat	
N ← **abs** (X)	N = **abs** (X)	Retourne la valeur absolue de **X**.	N = abs (-20) N = abs (.5.8)	N == 20 N == 5.8
N ← **ent** (X)	N = **int** (X)	Retourne un **Entier** représentant la partie entière de **X**.	N = int (5.2) N = int (-5.8)	N == 5 N == -5
N ← **arrondi** (X)	N = **round** (X)	Retourne **l'Entier** le plus proche de **X**. **N.B. :** En Python, si la partie fractionnaire est égale à **5**, l'entier **Pair** le plus proche est retourné.	N = round (2.2) N = round (2.8) N = round (2.5) N = round (3.5)	N == 2 N == 3 N == 2 N == 4
N ← **racinecarré** (X)	**from math import sqrt** N = **sqrt** (X)	Retourne un **Réel** représentant la racine carrée de **X**. Si **X** < 0, elle provoque une erreur.	N = sqrt (9) N = sqrt (25.0) N = sqrt (-5)	N == 3.0 N == 5.0 Erreur
N ← **aléa** (Vi , Vf)	**from random import randint** N = **randint** (Vi , Vf)	Retourne un entier d'une façon aléatoire et automatique de l'intervalle [**Vi** , **Vf**].	N = randint (2 , 5) N pourra avoir 2 ou 3 ou 4 ou 5	

Application 2

Saisir un réel b et un réel a une valeur aléatoire comprise entre [b+2, b+20] puis calculer et afficher la surface et le périmètre d'une éclipse.



$$aire = \pi \times a \times b$$

$$périmètre \approx \pi \sqrt{2(a^2 + b^2)}$$

Application 3

Saisir une chaîne de caractère ch représentant un code binaire de 8 bits. Le programme déterminera le code ASCII et le caractère qui correspondent à ce code.

Exemple : ch = "01011001" → le programme affichera : "89 : Y". Pour cet exemple, on détermine le code ASCII comme suit : Code binaire 0 1 0 1 1 0 0 1 Code ASCII $0*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$.

Les lettres alphabétiques majuscules ou minuscules (''A''..''Z'', ''a''..''z''), les chiffres (''0''..''9'), les signes de ponctuation (''.'', ''?'',…) et les symboles (''#'', ''&'','' ''…) // '' '' → le caractère espace.

Algorithme & Python	Rôle	Exemples
A← ord(c)	A contient le code ASCII du caractère c	print(ord(''A''))=…… → ord(''B'')=…… print(ord(''a''))=…… → ord(''b'')=……
C← chr(d)	C contient le caractère dont le code ASCII est d	print(chr(65))=…… → chr(66)=…… print(chr(97))=…… → chr(98)=……

Application 4

Saisir une chaîne de caractères ch alphabétique et afficher l'occurrence de chaque lettre de l'alphabet dans cette chaîne.

Exemple : ch="Anniversaire" → le programme affiche :

2 Lettre A	2 Lettre I	0 Lettre R
0 Lettre B	0 Lettre J	1 Lettre S
…	..	…
2 Lettre E	2 Lettre N	0 lettre X
0 Lettre F	0 Lettre O	0 Lettre Y
…	….	0 Lettre Z

Situation6

Ecrire un programme (Algorithme et implémentation en langage Python) permettant de déterminer :

Si une année A est bissextile ou non.
Une année A est bissextile sauf si (A est divisible par 4 et non divisible par 100) ou (A est divisible par 400).
L'année A saisie par clavier doit être un entier compris entre 1900 et 9999.

Exemple n°1 : Pour A= **2020** , le programme affichera: bissextile
Exemple n°2 : Pour A= **2019** , le programme affichera: non bissextile.

Retenons :

Notation en algorithmique	Notation en Python
Si Condition1 **Alors** Traitement1 **Sinon** **Si** Condition2 **Alors** Traitement2 **Sinon** TraitementN **FinSi**	**if** Condition1 **:** Traitement1 **elif** Condition2 **:** Traitement2 **else :** TraitementN

Application 1

Ecrire un algorithme qui permet de saisir deux dates valides d1 et d2 sous forme jj/mm/aaaa puis afficher la date la plus récente.

Exemple : d1=31/01/2017, d2=02/02/2018 → 02/02/2018 est plus récente que 31/01/2017

Application 2

Déterminer et afficher le nombre total d'appuis sur les touches du clavier d'un téléphone portable pour saisir un mot donné de N lettres, supposées non accentuées (4≤N≤9).

Sur les touches (2, 3, 4, 5, 6, 7, 8 et 9) du clavier d'un téléphone portable, sont inscrites des lettres pour écrire des messages en plus des chiffres.
Par exemple, sur la touche 5 sont inscrites les lettres **J**, **K** et **L**.

- Pour taper la lettre **J** on appuie **une seule fois**.
- Pour taper la lettre **K** on appuie **deux fois**.
- Pour taper la lettre **L** on appuie **trois fois**.

Retenons :

Notation en algorithmique	Notation en Python (Versions ≥3.10)
Selon Sélecteur Val1 : Traitement1 Val2 , Val3 , Val4 : Traitement2 Val5 .. Val 6 : Traitement3 …….. **Sinon** TraitementN **FinSelon**	**match** Sélecteur : **case** Val1 : Traitement1 **case** Val2 | Val3 | Val4 : Traitement2 **case** Sélecteur if Val5<=Sélecteur<=Val6 : Traitement3 **case** _: TraitementN

Application 3

Saisir l'heure (comprise entre 0 et 23) et les minutes (comprises entre 0 et 59) puis afficher l'heure qu'il sera une minute plus tard.

Exemple :

Si l'utilisateur tape 21 puis 32, le programme affiche : "Dans une minute, il sera 21 h 33 mns".
Si l'utilisateur tape 21 puis 59, le programme affiche : "Dans une minute, il sera 22 h 00 mns".
Si l'utilisateur tape 23 puis 59, le programme affiche : "Dans une minute, il sera 00 h 00 mns".

Situation7 Extrait Bac 2021

Le jeu de dominos est un jeu chinois qui comporte 28 dominos. Un domino est formé de 2 parties. Chaque partie contient de 0 à 6 points.

Une suite de dominos est dite **valide** lorsque la 2ème partie d'un domino est identique à la 1ère partie du domino voisin.

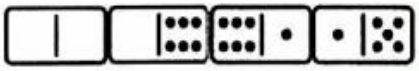


Figure 1- Suite valide de dominos

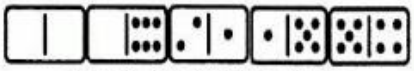
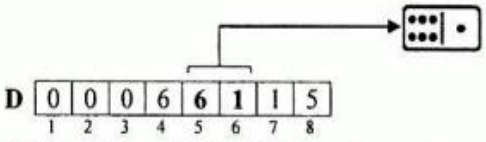


Figure 2- Suite invalide de dominos

La suite de dominos de la **Figure 2** est **invalide** car la 2ème partie du 2ème domino (6 points) est différente de la 1ère partie du domino voisin (2 points).

Afin d'automatiser ce jeu, on se propose d'utiliser un tableau **D** d'entiers. Chaque case contient une valeur comprise entre **0** et **6**. Ainsi, le tableau correspondant à la **Figure 1** est représenté comme suit :



Les paires **(D[1], D[2])**, **(D[3], D[4])**, **(D[5], D[6])** et **(D[7], D[8])** représentent une suite de 4 dominos

Questions

1) Présenter, sous forme d'un tableau **D**, la suite de dominos de la **Figure 2**.
2) Ecrire un algorithme nommé **Suite_Dominos** qui permet de :
   - Saisir la taille **n** du tableau **D**, avec **n** un entier pair et **4≤n≤56**.
   - Remplir le tableau **D** par **n** entiers compris entre **0** et **6**.
   - Vérifier si la suite de dominos représentée par le tableau **D** est **valide** ou **invalide** puis afficher un message adéquat.
3) Dresser le tableau de déclaration des objets utilisés dans l'algorithme **Suite_Dominos**, en adoptant l'en-tête suivant :

Objet	Type	Rôle

Les tableaux à une seule dimension

1. Déclaration en algorithmique

❖ 1ère méthode

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Tableau	**Tableau** de **N** **Type_élément**

❖ 2ème méthode

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Nom_Type_Tableau = **Tableau** de **N** **Type_élément**

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Tableau	Nom_Type_Tableau

2. Déclaration en Python en utilisant la bibliothèque Numpy

Déclaration dans le cas général
import numpy as np **Nom_Tableau = np.array ([Type_élément ( )] * N [,dtype = object])**

Exemples de déclarations en Python	
Déclaration	**Explication**
import numpy as np T = np.array ([int ( )] * 20)	Pour déclarer un tableau de 20 entiers avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([float ()] * 100)	Pour déclarer un tableau de 100 réels avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([str ( )] * 50 , dtype = object) *ou bien* T = np.array ([str] * 50)	Pour déclarer un tableau de 50 chaînes de caractères avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([bool ( )] * 10)	Pour déclarer un tableau de 10 booléens avec importation de la bibliothèque Numpy.

Remarque :

- Les éléments d'un tableau à une seule dimension doivent être de même type.
- Les indices des éléments d'un tableau sont de type scalaire.
- Pour accéder à un élément d'indice « **i** » d'un tableau : Nom_Tableau [i]

Application 1

Saisir un entier n pair, remplir un tableau T par n entiers positifs, calculer et afficher la somme, le maximum, le minimum des éléments de T.

Exemple : n=6

T	15	125	4	0	-2	-13
	1	2	3	4	5	6

S=129, max=125, min=-13

Application 2

Saisir un entier n>0, remplir T1 par n chaînes numériques non vides et de taille maximale 7, remplir un tableau T2 telle que : T2[i] = Somme des chiffres de T1[i] puis afficher T2.

Exemple : n=4

T1	"157"	"11"	"4"	"1078"

T2	13	2	4	16

Application 3

Saisir n (2<n<10), de remplir aléatoirement T par n lettres minuscules, de diviser T en 2 tableaux (T1 : tableau de voyelles, T2 : tableau de consonnes) et d'afficher T1 et T2.

Exemple : n=5

T	"a"	"B"	"C"	"d"	"E"

T1	"a"	"E"

T2	"B"	"C"	"d"

Application 4

Saisir 5≤n≤20, remplir T par n entiers composés de 2 chiffres, saisir p (compris entre 0 et n-1), saisir X composé de 2 chiffres, insérer X dans le tableau T à la position p et afficher T.

Exemple : n=5, p=3, X=12

T	25	65	10	77	45
	0	1	2	3	4

T après insertion de X à la position p

T	25	65	10	12	77	45
	0	1	2	3	4	5

Application 5 : Extrait (Bac pratique 2006)

Ecrire un algorithme qui permet de :

- de remplir un tableau **T** de **n** chaînes de caractères (**2<n<20**). Chaque chaîne doit avoir un nombre de caractères supérieur ou égal à son indice dans le tableau.
- d'afficher pour chaque élément T[i] du tableau, les **i** premiers caractères de la chaîne.

Exemple : Soit T un tableau de 6 chaînes de caractères.

T	Bit	Modem	Ecran	Souris	Processeur	Mémoire
	1	2	3	4	5	6

Le programme affichera :

B
Mo
Ecr
Sour
Proce
Mémoir

Application 6 : Extrait (Bac pratique 2006)

On se propose d'écrire un Algorithme permettant de remplir deux tableaux **T1** et **T2** de N entiers à deux chiffres chacun ($2 \leq N \leq 15$) puis de former un tableau **T** tel que un élément **T[i]** est le résultat de la fusion des deux éléments **T1[i]** et **T2[i]** selon le principe suivant :

- Insérer le chiffre des dizaines du plus petit nombre parmi **T1[i]** et **T2[i]**, entre les deux chiffres du plus grand nombre parmi **T1[i]** et **T2[i]**.
- Mettre le chiffre des unités du plus petit nombre parmi **T1[i]** et **T2[i]**, à droite du nombre obtenu.

Exemples :
- Pour T1[i] = 52 et T2[i] = 36 , T[i] sera égal à 5326
- Pour T1[i] = 13 et T2[i] = 47 , T[i] sera égal à 4173

Application 7 : Extrait (Bac pratique 2006)

Un nombre est dit **riche** si au moins un de ses facteurs premiers est répété deux fois ou plus dans la décomposition du nombre en facteurs premiers.

Exemples :

- Le nombre **72** est dit **riche**, car 2 et 3 se répètent plus qu'une fois dans sa décomposition en facteurs premiers ($72 = 2^3 * 3^2$).
- Le nombre **22 n'est pas riche**, car tous ses facteurs premiers existent une seule fois ($22 = 2*11$).

Travail demandé :

Ecrire un programme Pascal qui permet de remplir un tableau **T** par **N** (**3 < N < 10**) entiers positifs non nuls à deux chiffres ou à trois chiffres, de trouver et d'afficher le ou les nombre(s) riche(s) du tableau **T**.

Exemple :

Pour **N = 6** et le tableau **T** suivant :

T	22	15	90	540	30	72
	1	2	3	4	5	6

Le programme affiche : "les nombres riches sont : 90, 540, 72"

En effet, 22 = 2*11, 15 = 3*5, **90 = 2*3*3*5**, **540 = 2*2*3*3*3*5**, 30 = 2*3*5 et **72= 2*2*2*3*3**

Application 6 : Extrait (Bac pratique 2015)

Pour sécuriser l'envoi des messages, deux chercheurs cryptent leurs messages en utilisant le principe suivant :

1. Saisir le message à crypter **msg**, sachant qu'il est composé uniquement par des lettres,
2. Remplir un tableau **T** par les ordres alphabétiques des lettres de **msg** de façon à ce que **T[i]** lui correspond de **msg[i]** (Sachant que "**A**" et "**a**" sont d'ordre 1, "**B**" et "**b**" sont d'ordre 2, ...),
3. Remplacer chaque **T[i]** par $(T[i])^e$ **mod (p*q)** avec **p**, **q** et **e** trois constantes ayant pour valeurs respectivement **17**, **19** et **5**.

   Le tableau **T** ainsi obtenu représente le code de la chaine **msg**.

Exemple :

Pour la chaîne **msg**="Bonjour", **T** sera remplit initialement comme suit :

T	2	15	14	10	15	21	18
	1	2	3	4	5	6	7

En effet "B" et d'ordre alphabétique 2, "o" est d'ordre alphabétique 15, …

Après avoir coder en remplaçant chaque **T[i]** par $(T[i])^e$ **mod (p*q) on obtient :**

T	32	2	29	193	2	89	18
	1	2	3	4	5	6	7

En effet :
T[1] est remplacé par $(T[1])^e$ mod (p*q)= 2^5 mod (17*19) = 32
T[2] est remplacé par $(T[2])^e$ mod (p*q)= 15^5 mod (17*19) = 2
Etc.

Travail demandé :

Ecrire un programme Python qui permet de saisir une chaîne non vide formée uniquement par des lettres, de la crypter selon le principe décrit ci-dessus et d'afficher le tableau de code obtenu.

Situation 8

Ecrire un programme qui permet de lire N mesures de température et de réaliser les traitements suivants :

1. Le nombre des mesures à analyser est compris entre 0 inclus et 100 exclu
2. Chaque mesure est comprise entre 0 et 50
3. Afficher la liste des températures
4. Trouver la température minimum et la température maximum.
5. Chercher l'existence d'une mesure **M** donnée.
6. Trier le tableau des mesures selon un ordre croissant.
7. Chercher l'existence d'une mesure **Me** donnée en appliquant le principe de la recherche dichotomique.

Retenons :

Trier un tableau / une chaîne = Ordonner ses éléments par ordre croissant ou décroissant. Il existe plusieurs principes de tri : Tri par sélection, Tri à bulles, Tri par insertion.

Principe

Le principe du tri par sélection consiste à :

1. Déterminer le plus petit élément de l'élément (ou le plus grand)
2. Le placer à sa position finale (c'est-à-dire en dernière ou en première positiondu tableau)
3. Rechercher le second plus grand élément (ou le second plus petit élément)
4. Le placer à sa position finale (c'est-à-dire en avant-dernière ou en seconde position).
5. Répéter ce traitement jusqu'à ce que le tableau soit trié.

Exemple

On se propose d'utiliser la méthode du tri par sélection pour trier un tableau T selon l'ordre décroissant.
Considérons le tableau T contenant les 7 éléments suivants :

T=	14	2	47	10	18	13	5
	1	2	3	4	5	6	7

Etape 1

- Cherchez la valeur maximale dans T(1 → 7) : Cette valeur est 47 et son indice est 3.
- Permutez cette case avec la case n°1

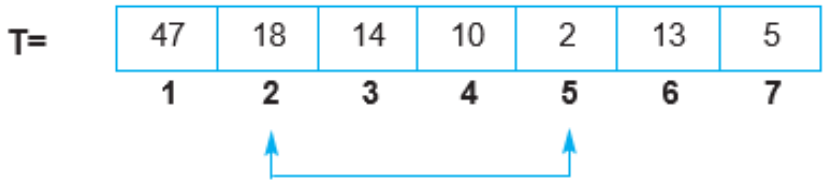
Résultat :

T=	47	2	14	10	18	13	5
	1	2	3	4	5	6	7

Etape 2

- Cherchez la valeur maximale dans T(2 → 7) : Cette valeur est 18 et son indice est 5.
- Permutez cette case avec la case n°2

Résultat :



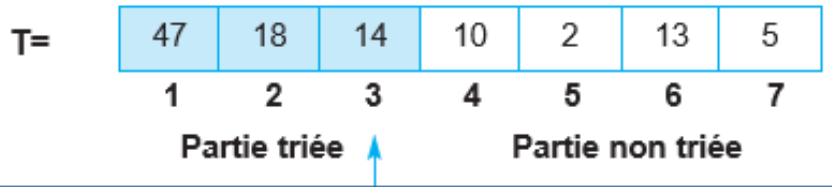
Remarque :

Le tableau comporte une partie triée composée des deux premiers éléments et une partie non encore triée qui comporte le reste des éléments de T.

Etape 3

- Cherchez la valeur maximale dans T(3 → 7) : Cette valeur étant 14 et son indice est 3.
- Puisque la valeur maximale correspond à la première case de la partie non triée du tableau T, vous n'avez pas besoin de réaliser l'action de permutation.

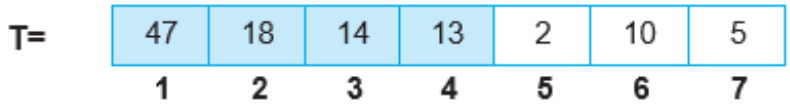
Résultat :



Etape 4

- Cherchez la valeur maximale dans T(4 → 7) : Cette valeur étant 13 et son indice est 6.
- Permutez cette case avec la case n°4

Résultat :

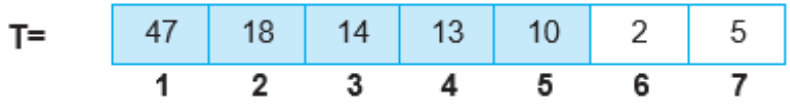


Partie triée **Partie non triée**

Etape 5

- Cherchez la valeur maximale dans T(5 ← 7) : Cette valeur est 10 et son indice est 6.
- Permutez cette case avec la case n°5

Résultat :

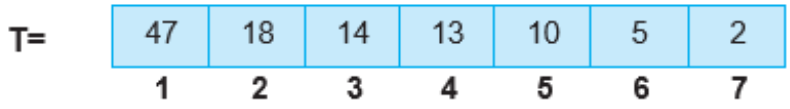


Partie triée **Partie non triée**

Etape 6

- Cherchez la valeur maximale dans T(6 → 7) : Cette valeur est 5 et son indice est 7.
- Permutez cette case avec la case n°6

Résultat :



Remarque :

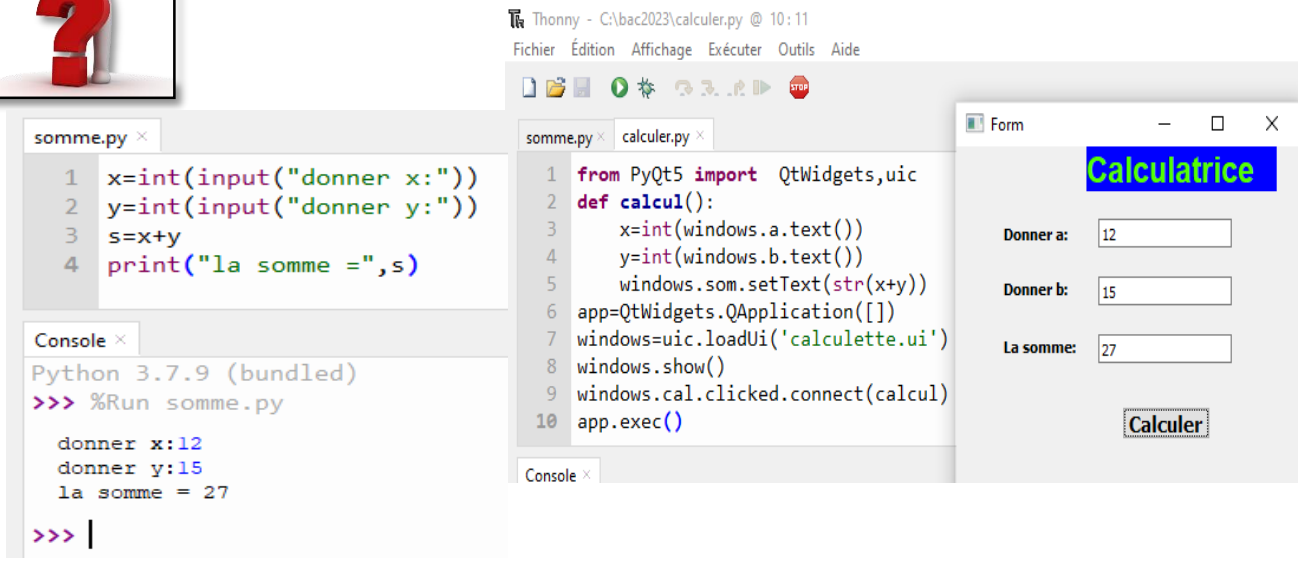
Nous n'avons pas besoin de traiter le dernier élément du tableau T, puisque si les 6 premiers éléments sont ordonnés alors automatiquement le dernier sera le plus petit et par conséquent il se trouve à sa bonne position.

Retenons :

La recherche d'un élément dans un tableau ou dans une liste de valeurs est un traitement très utilisé en informatique. Parmi les méthodes de recherche, il existe :

- La recherche séquentielle qui consiste à parcourir une liste de valeurs jusqu'à trouver la valeur cherchée ou atteindre la fin de la liste.
- La recherche dichotomique qui consiste à chercher en subdivisant la série ordonnée en deux parties égales et vérifier dans quelle partie figurerait la valeur recherchée, puis répéter ce processus.

Situation 9



Programmation Textuelle **Programmation graphique**

Présentation :

Les interfaces graphiques (ou interfaces homme/machine) sont appelées GUI (Graphical User Interface). Elles permettent à l'utilisateur d'interagir avec un programme informatique, grâce aux différents objets graphiques (zone de texte, case à cocher, bouton radio, bouton poussoir, menu, …). Ces objets graphiques sont généralement actionnés avec un dispositif de pointage, le plus souvent la souris.

Comment créer une interface graphique ?

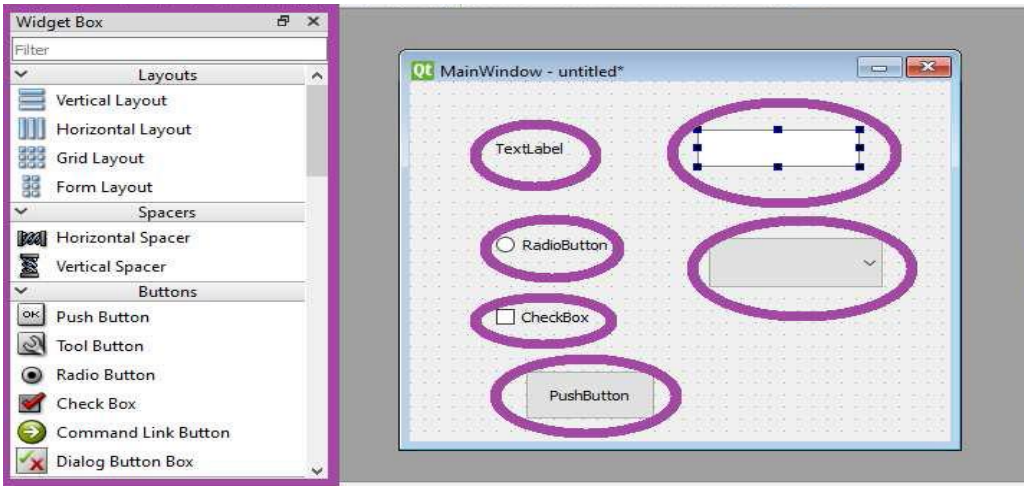
On va créer l'interface avec l'outil de création d'interface graphique GUI **(Qt Designer**) puis on va appeler le fichier (**Nom_fichier.ui**) dans le code python (**Nom_fichier.py**) crée par l'éditeur Python (**Thonny**).

Présentation de Qt Designer

Qt Designer est un générateur d'interface graphique, elle permet de créer vos fenêtres visuellement, elle vous permet de créer des objets graphiques (**widgets**) et modifier ses propriétés (Nom, Taille, couleur, …), etc..

Notion de Widget

Le Widget est un composant d'interface graphique (Label, zone de texte, bouton radio, case à côcher, bouton poussoir, menu, liste déroulante, barre de défilement, …)



Remarque :

- Le fichier créé par Qt Designer (**l'interface graphique**) possède l'extension ".ui" (User Interface).
- Après avoir réalisé l'interface graphique de votre application (**Nom_Fichier.ui**), on doit passer à la partie programmation (écriture du code) (**Nom_Fichier.py**).
- Pour cela, il faut lancer un éditeur Python (Environnement de développement) et écrire le code correspondant à votre application.
- De préférence, il faut mettre les deux fichiers **Nom_Fichier.ui** et **Nom_Fichier.py** dans le même répertoire.

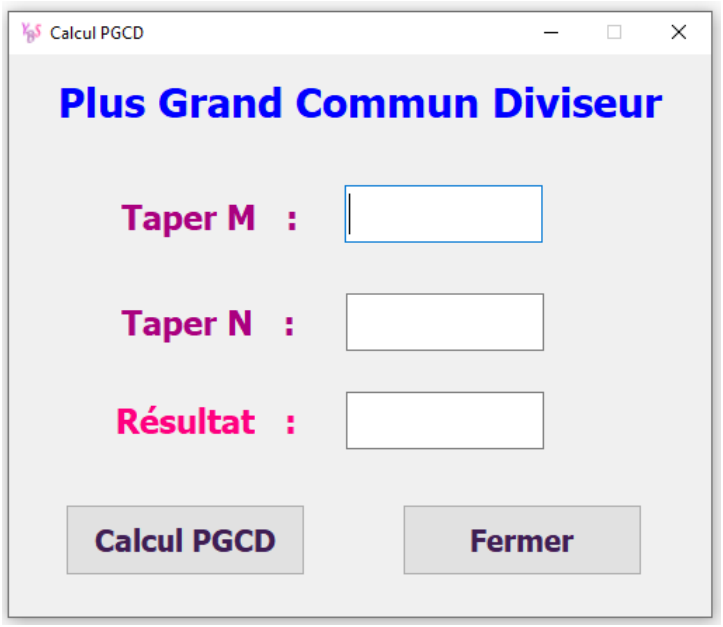
Nom_Fichier.py

```
# Importations à faire pour réaliser une interface graphique
from PyQt5 import QtWidgets , uic
# Définir une fonction
def nom_fonction( ) :
< Traitements >
# Création d'une application
Nom_objet_App = QtWidgets.QApplication([])
# charger le fichier crée par Qt Designer
Nom_objet_Fen = uic.loadUi( "nom_fichier.ui" )
# Visualiser la fenêtre
Nom_objet_Fen.show()
# Signal (événement) : Exemple clicked : Appel du fonction
Nom_objet_Fen.Nom_Widget.clicked.connect(nom_fonction)
# Exécution de l'application
app.exec_()
```

Application1 : PGCD avec interface graphique

Notre mini projet consiste à créer une interface graphique avec l'outil de création Qt Designer qui permet de saisir deux entiers M et N positifs non nuls et afficher le Plus Grand Commun Diviseur (**PGCD**) en utilisant la méthode des différences.

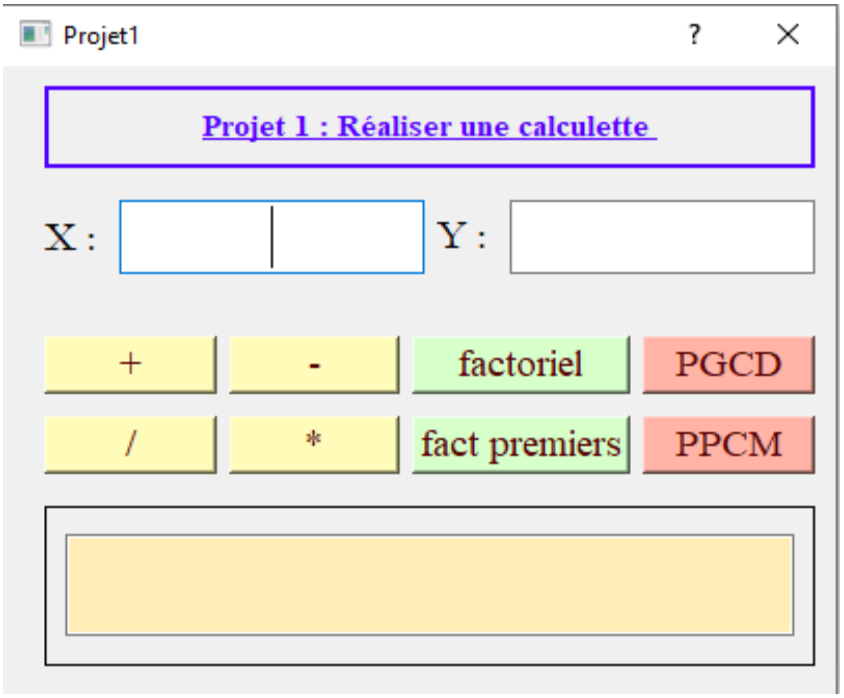
PGCD (27,90) = PGCD (27, 63)
= PGCD (27, 36)
= PGCD (27, 9)
= PGCD (18,9)
= PGCD (9,9) = 9



Application2 : Réalisation d'une calculette

On se propose de créer une calculatrice graphique permettant de saisir deux entiers dans deux zones de textes et de réaliser une des opérations ci-dessous et d'afficher le résultat dans la zone associée :

- Opérations arithmétiques : +,-,*,/
- Calcul du factoriel du premier nombre
- Décomposition en facteurs premiers du premier nombre
- Calcul du PGCD et PPCM des deux entiers



Extrait Bac2011

Exercice1

Répondre par **Vrai** si la proposition est correcte ou par **Faux** dans le cas contraire.

Proposition	Réponse
Un tableau de réels peut être rempli par des entiers.	
Le compteur d'une structure répétitive complète doit être de type scalaire.	
Les opérateurs **DIV** et **MOD** peuvent être appliqués sur les nombres réels.	
Efface (ch,longueur(ch)-1,2) efface les deux derniers caractères de la chaîne **ch**.	

Exercice2

*Un entier **n** de **4** chiffres est dit **valable**, si ses trois derniers chiffres sont des multiples de son chiffre des milliers.*

Exemple : L'entier **2648** est **valable** car son chiffre des milliers est **2** et il est suivi par les chiffres **6**, **4** et **8** qui sont tous multiples de **2.**

On se propose d'écrire un programme qui permet de lire un entier positif **n** composé de **4** chiffres puis d'afficher s'il est **valable** ou non.

Exemple 1 : Si **n= 2888** alors le programme affichera : ***Cet entier est valable.***

Exemple 2 : Si **n= 2179** alors le programme affichera : ***Cet entier n'est pas valable.***

Exercice3

```
r ← 0
Répéter
   r ← r + n MOD 10
   n ← n DIV 10
Jusqu'à (n = 0)
```

Questions

1) Quelle est la valeur retournée par la fonction "**Traitement**" pour **n = 125 ?**

.....

2) Quelle est la valeur retournée par la fonction "**Traitement**" pour **n = 458 ?**

.....

3) Donner le rôle de cette fonction.

.....

.....

Extrait Bac2012

Exercice1

Compléter le tableau suivant par les valeurs des variables indiquées sachant que toutes les instructions sont correctes.

Instructions	Valeurs
X ← Tronc (11.8) Y← Arrondi (11.8)	X= ..... Y=.....
Valeur ("138.25" , N , E)	N= ..... E=.....
Convch (138.25 , Ch)	Ch=.....
Ch1←"information" Efface (ch1 , 3 , 6)	Ch1=.....
Ch1←"information" Ch2← sous_chaine (ch1 , 3 , 6)	Ch1=..... Ch2=.....

Exercice2

On se propose d'écrire un programme qui saisit un entier naturel **N (2 ≤ N ≤ 20)**, puis remplit un tableau **T** par **N** nombres complexes de la forme **a+bi** avec **a** et **b** deux entiers naturels non nuls. Chaque suite d'éléments du tableau T qui ont le même module sera affiché sur une ligne à part. On rappelle que le module d'un nombre complexe de la forme **a+bi** est $\sqrt{a^2+b^2}$.

Pour réaliser le traitement demandé on suivra les étapes suivantes :

- Remplir un tableau **M** par les modules des éléments de **T** de façon à ce que **M[i]** soit le module du nombre complexe **T[i]**.
- Trier simultanément les deux tableaux **T** et **M** selon l'ordre décroissant des valeurs du tableau M.
- Afficher chaque suite d'éléments du tableau T qui ont le même module sur une ligne à part.

Exemple :

Pour **N=6** et pour le tableau **T** suivant :

T	2+3i	2+15i	2+17i	23+3i	17+2i	15+2i
	1	2	3	4	5	6

- Le remplissage de **M** donne le tableau suivant :

M	3. 60	15.13	17.12	23.19	17.12	15.13
	1	2	3	4	5	6

- Après le tri on obtient les deux tableaux suivants :

T	23+3i	2+17i	17+2i	2+15i	15+2i	2+3i
	1	2	3	4	5	6
M	23.19	17.12	17.12	15.13	15.13	3.60
	1	2	3	4	5	6

- Le programme affiche les lignes suivantes :
  23+3i
  2+17i 17+2i
  2+15i 15+2i
  2+3i

Travail demandé:

1. Analyser le problème en le décomposant en modules.
2. Analyser chacun des modules proposés.

Extrait Bac2013

Exercice1

Pour chacune des instructions suivantes, valider chaque proposition en mettant dans la case correspondante la lettre **V** si elle est correcte ou **F** dans le cas contraire.

a. Soit l'instruction **C ← Sous_chaîne ("Baccalauréat",4,1)**.

Elle permet d'affecter le caractère **"c"** à la variable **C**. ☐

La variable **C** doit être déclarée de type **caractère**. ☐

La variable **C** doit être déclarée de type **Chaîne**. ☐

b. L'instruction **X ← Aléatoire (6) + 4** permet d'affecter à la variable **X** une valeur aléatoire de l'intervalle

[4,6] ☐

[4,10] ☐

[4,9] ☐

c. L'instruction **R ← Arrondi (12.5)** permet d'affecter à la variable **R**

l'entier **12** ☐

l'entier **13** ☐

le réel **13.0** ☐

d. Soit l'affectation suivante **C ← Majus("?")**.

Elle permet d'affecter à la variable **C** le caractère "?" en gras. ☐

Elle permet d'affecter à la variable **C** le caractère "?". ☐

La variable **C** doit être de type **Caractère**. ☐

Extrait Bac2014

Problème : (12 points)

On se propose de crypter un message composé par des mots séparés par un seul espace et ne contenant aucun signe de ponctuation (, ; . : ! ?) en utilisant le principe suivant :

1) Placer chaque mot du message initial dans une case d'un tableau **T**. On suppose que le message est composé d'au maximum 20 mots.
2) Pour chaque élément du tableau **T**, ajouter autant de fois le caractère "*" pour que sa longueur sera égale à celle du mot le plus long dans le tableau **T**.
3) Dans un nouveau tableau **T1** de taille **N1** (**N1**=longeur du mot le plus long), répartir les lettres du mot se trouvant dans la case **T[1]** de façon à placer la lettre d'indice **i** du mot dans la case d'indice **i** du tableau **T1**.
4) Répartir de la même façon les lettres du mot contenu dans la case **T[2]** en concaténant à chaque fois la lettre d'indice **i** avec le contenu de la case **i** du tableau **T1**
5) Répartir de la même façon le reste des mots de **T** dans **T1**.
6) Concaténer les mots obtenus dans **T1** en les séparant par un espace pour obtenir le message crypté.

Exemple : *Si le message à crypter est "Bonjour Sami j'ai fini mon travail", les étapes de cryptage sont :*

Etape 1 : *Répartir les mots du message dans le tableau T :*

T=	Bonjour	Sami	j'ai	fini	mon	travail

Etape 2 : *Ajouter le caractère "*" autant de fois pour obtenir des mots dont la longueur de chacun est égale à celle du mot le plus long.*

Etant donné que "Bonjour" est le mot le plus long du message (7 caractères), on obtient le tableau **T** suivant :

T=	Bonjour	Sami***	j'ai***	fini***	mon****	travail

Etape 3 : *Répartir les lettres de **T[1]** dans **T1***

T=	B	o	n	j	o	u	r

Etape 4 : *Répartir les lettres de **T[2]** dans **T1***

T1 =	BS	oa	nm	ji	o*	u*	r*

Etapes suivantes : *Répartir le reste des mots de **T** dans **T1***

T1 =	BSjfmt	oa'ior	nmanna	jiii*v	o****a	u****i	r****l

Le message crypté sera alors "BSjfmt oa'ior nmanna jiii*v o****a u****i r****l"

Travail demandé :

1. Analyser le problème en le décomposant en modules.
2. Analyser chacun des modules proposés.

Extrait Bac2015

Problème (11 points)

Un nombre M est dit « nombre premier sûr », s'il est un nombre premier de la forme 2*p + 1 avec p un nombre premier.

Exemples :

- ✓ Si M = 11, alors M est un nombre premier sûr. En effet, 11 est premier et il peut s'écrire sous la forme 2*p+1 où p = 5 qui est un nombre premier.
- ✓ Si M = 31, alors M n'est pas un nombre premier sûr. En effet, 31 est premier et il peut s'écrire sous la forme 2*p+1 où p = 15 qui n'est pas un nombre premier.

NB : Un nombre entier supérieur à 1 est dit premier s'il n'est divisible que par 1 et par lui-même.

On se propose d'écrire un programme qui permet de :

1. Remplir un tableau T par N entiers strictement supérieurs à 1 (avec $10 \leq N < 45$).
2. Trier dans l'ordre croissant les éléments premiers sûrs du tableau T suivis du reste des éléments sans tri.
3. Afficher le tableau T résultant.

Exemple : Pour N = 10 et le tableau T suivant :

le contenu du tableau suivant :

T	5	25	59	23	13	47	31	100	7	107
	1	2	3	4	5	6	7	8	9	10

Le programme affichera

T	5	7	23	47	59	107	25	13	31	100
	1	2	3	4	5	6	7	8	9	10

Eléments premiers sûrs triés dans un ordre croissant — Eléments non premiers sûrs

Travail demandé :

1) Analyser le problème en le décomposant en modules.

2) Analyser chacun des modules envisagés.

Succession parfaite (Bac Pratique 2023)

On se propose de concevoir une interface graphique permettant de saisir deux nombres **positifs M** et **N** puis de vérifier s'ils forment une **succession parfaite** ou non.

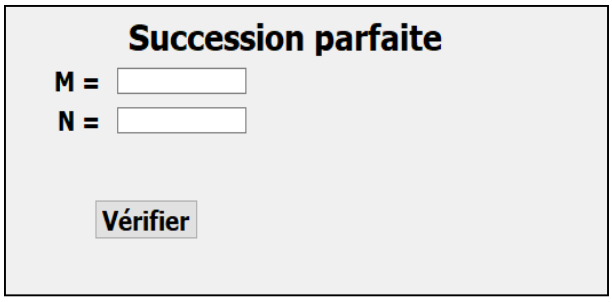
Une succession parfaite de deux nombres positifs **M** et **N** est une chaîne de caractères **ch** formée par une succession de chiffres consécutifs distincts où le **pas** de la succession est égal à **1**. Cette chaîne est obtenue enconcaténant les chiffres de **M** et **N** puis en les triant dans l'ordre croissant.

Exemples :

- Pour M=2748 et N=365, **ch** = "2345678". Les chiffres de **ch** forment une **succession parfaite**.En effet, le **pas** de la succession est égal à **1** entre tous les chiffres de **ch**.
- Pour M=8473 et N=546, **ch** = "3**44**5678". Les chiffres de **ch** ne forment pas une **succession parfaite**.En effet, le **pas** de la succession est différent de **1** entre le deuxième et le troisième chiffre.
- Pour M=2748 et N=956, **ch** = "**24**56789". Les chiffres de **ch** ne forment pas une **succession parfaite**.En effet, le **pas** de la succession est différent de **1** entre le premier et le deuxième chiffre.

L'interface graphique à concevoir contient les éléments suivants, comme l'illustre la capture d'écranci-dessous :

- Un label contenant le texte " **Succession parfaite**"
- Un label contenant le texte "**M =** "
- Une zone de saisie pour la saisie du nombre **M**
- Un label contenant le texte " **N=** "
- Une zone de saisie pour la saisie du nombre **N**
- Un label pour afficher le message adéquat
- Un bouton intitulé "**Vérifier**"



Travail demandé :

1) Concevoir l'interface graphique présentée précédemment et l'enregistrer sous le nom **InterfaceSuccession**.

2) Créer un programme Python et l'enregistrer sous le nom **Succession**, dans lequel, il est demandé :

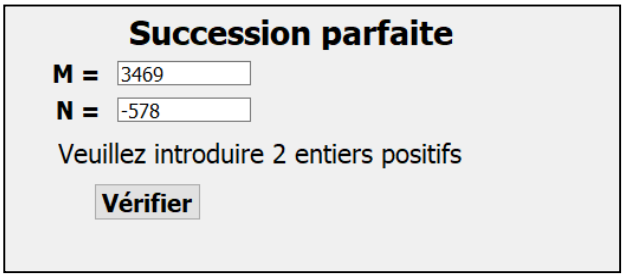
   a) de développer une fonction nommée **Verifier (M,N)** qui permet de vérifier si **M** et **N** forment unesuccession parfaite ou non.

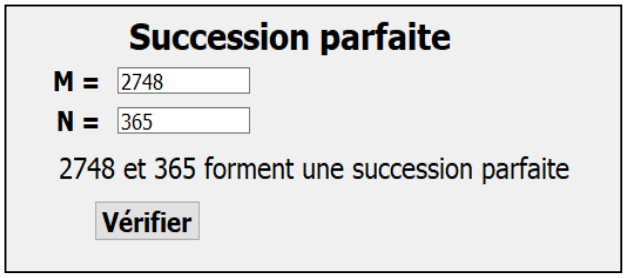
b) de développer un module **Play** qui s'exécute suite à un clic sur le bouton "**Vérifier**", permettant :
   - de récupérer les deux entiers **M** et **N** saisis qui doivent être positifs.
   - d'exploiter la fonction **Verifier (M,N)** afin d'afficher le message adéquat via le **label** dédié àl'affichage dans l'interface graphique **InterfaceSuccession**.

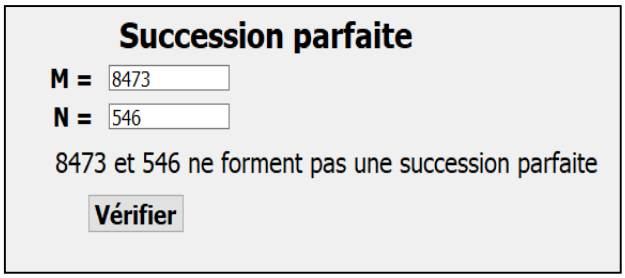
c) d'ajouter les instructions permettant d'exploiter l'interface graphique intitulée **InterfaceSuccession** en se référant à l'annexe ci-après.

N.B. : *l'affichage doit être conforme aux exemples d'exécutions suivants* :

Exemples d'exécutions :







Annexe

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
……
app = QApplication([])
windows = loadUi ("Nom_Interface.ui")
windows.show()
windows.Nom_Bouton.clicked.connect
(Nom_Module)app.exec_()
```

Grille d'évaluation

Tâches	Nombre de points
Conception de l'interface **InterfaceSuccession**	**4 pts**
Création et enregistrement du programme **Succession**	**1 pt**
Développement de la fonction **Verifier**	**6 pts**
Développement du module **Play**	**4 pts**
Ajout des instructions de l'exploitation de l'interface	**3 pts**
Modularité et cohérence	**2 pts**

Miroirs de mots (Bac Pratique 2023)

On se propose de concevoir une interface graphique permettant de saisir une chaîne de caractères **ch** et de la crypter en formant une nouvelle chaîne par les miroirs des mots de **ch** dans leur ordre d'apparition. On rappelle que le miroir d'un mot consiste à permuter le premier caractère avec le dernier, le deuxièmecaractère avec l'avant dernier et ainsi de suite.

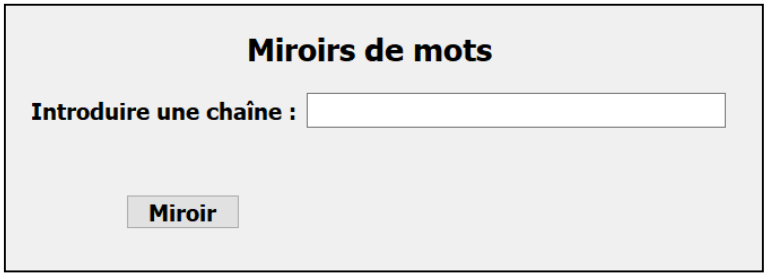
Exemple :

Pour **ch** = "**La vie est une aventure merveilleuse**"

On obtient la chaîne cryptée : "**aL eiv tse enu erutneva esuellievrem**"

L'interface graphique à concevoir contient les éléments suivants, comme l'illustre la capture d'écranci-dessous :

- Un label contenant le texte "**Miroirs de mots**"
- Un label contenant le texte "**Introduire une chaîne :** "
- Une zone de saisie pour la saisie d'une chaîne
- Un label pour afficher le résultat
- Un bouton intitulé "**Miroir**"



Travail demandé :

1) Concevoir l'interface graphique présentée précédemment et l'enregistrer sous le nom **InterfaceMiroirsMots**.

2) Créer un programme Python et l'enregistrer sous le nom **MiroirsMots**, dans lequel, il est demandé :

a) de développer une fonction nommée **Miroir (M)** qui permet de retourner le miroir d'un mot **M**.

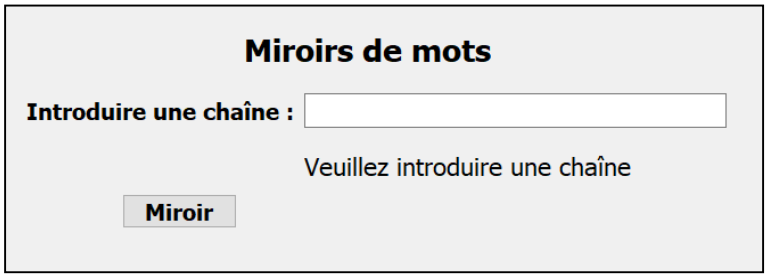
b) de développer un module **Play**, qui s'exécute suite à un clic sur le bouton "**Miroir**", permettant :

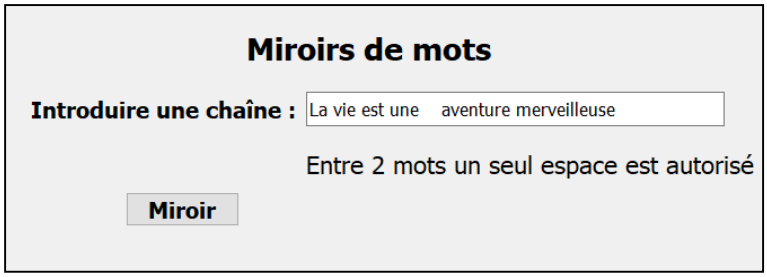
- de récupérer la chaîne **ch** saisie. La chaîne **ch** doit être non vide et de longueur inférieure à**50**, contient seulement des lettres alphabétiques en minuscule et chaque deux mots consécutifs sont séparés par un seul espace.
- de déterminer la chaîne cryptée en utilisant la fonction **Miroir (M)** et d'afficher le résultat via le **label** dédié à l'affichage dans l'interface graphique **InterfaceMiroirsMots**.

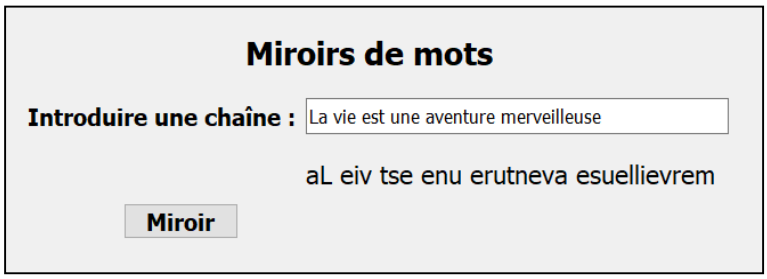
c) d'ajouter les instructions permettant d'exploiter l'interface graphique intitulée **InterfaceMiroirsMots** en se référant à l'annexe ci-après.

N.B. : *l'affichage doit être conforme aux exemples d'exécutions suivants :*

Exemples d'exécutions :







Annexe

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
……
app = QApplication([])
windows = loadUi ("Nom_Interface.ui") windows.show()
windows.Nom_Bouton.clicked.connect (Nom_Module)
app.exec_()
```

Grille d'évaluation

Tâches	Nombre de points
Conception de l'interface "**InterfaceMiroirsMots**"	**4 pts**
Création et enregistrement du programme " **MiroirsMots** "	**1 pt**
Développement de la fonction "**Miroir**"	**4 pts**
Développement du module "**Play**"	**6 pts**
Ajout des instructions de l'exploitation de l'interface	**3 pts**
Modularité et cohérence	**2 pts**