

LYCEE PILOTE BEJA
ANNEE SCOLAIRE 2021/2022

REVISION BAC PRATIQUE 2022
MATIERE : INFORMATIQUE
ENSEIGNANT : ABDELHAMID GUIZANI

SOMMAIRE

Sujet	Page
Sujet N°1	02
Sujet N°1- Correction	05
Sujet N°2	04
Sujet N°2- Correction	05
Sujet N°3	07
Sujet N°3- Correction	09
Sujet N°4	10
Sujet N°4- Correction	12
Sujet N°5	13
Sujet N°5- Correction	15
Sujet N°6	16
Sujet N°6- Correction	18
Sujet N°7	19
Sujet N°7- Correction	21
Sujet N°8	22
Sujet N°8- Correction	24
Sujet N°9	25
Sujet N°9- Correction	27
Sujet N°10	28
Sujet N°10- Correction	30
Sujet N°11	31
Sujet N°11- Correction	33
Sujet N°12	34
Sujet N°12- Correction	36

Sujet n°1 : Jeu de chance

Important : Dans le répertoire *Bac2022*, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Dans le cadre d'une campagne publicitaire, une société commerciale a décidé d'organiser chaque semaine un jeu de chance pour ses clients.

Le principe du jeu consiste à calculer le nombre de chance à partir du numéro de téléphone du client donné et d'afficher le message "*Félicitations, vous avez gagné.*" dans le cas où ce nombre est **premier** ou le message "*Désolé, vous n'avez pas gagné.*" dans le cas contraire.

Sachant que :

- Le numéro de téléphone devrait commencer par 2, 4, 5 ou 9.
- Le nombre de chance est la somme de chaque chiffre du numéro de téléphone multiplié par son indice avec l'indice du premier chiffre est 0.
- Un nombre premier est un nombre qui est divisible par 1 et par lui-même.

Exemple :

Donner le numéro : **29234560**

Le programme affiche : ***Désolé, vous n'avez pas gagné.***

En effet, le nombre de chance est égal à 99 qui n'est pas un nombre premier.

$99 = 2 * 0 + 9 * 1 + 2 * 2 + 3 * 3 + 4 * 4 + 5 * 5 + 6 * 6 + 0 * 7$ c'est la somme de chaque chiffre du numéro de téléphone multiplié par son indice :

Numéro téléphone	2	9	2	3	4	5	6	0
Indice	0	1	2	3	4	5	6	7

Ci-après, un algorithme de la fonction "**Chance**" à exploiter pour résoudre le problème posé.

Fonction Chance (Ch : Chaîne) : Chaîne

DEBUT

Si NON (Estnum (Ch) **ET** long (Ch) = 8 **ET** Ch[0] ∈ ["2", "4", "5", "9"]) **Alors**
 msg ← "Vérifier le numéro de téléphone !"

Sinon

 msg ← "Désolé, vous n'avez pas gagné."

 s ← 0

Pour i de 0 à long (Ch) - 1 Faire

 s ← s + valeur (Ch [i]) * i

Fin Pour

Si premier (s) Alors

 msg ← "Félicitation, vous avez gagné."

FinSi


FinSi

Retourner msg

FIN

La société a décidé de créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le nom de la société.
- Un label demandant la saisie du numéro de téléphone.
- Une zone de saisie permettant la saisie du numéro de téléphone.
- Un bouton nommé "**Jouer**".
- Un label pour afficher un message.



Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_Jeu**".
- 2) Implémenter en Python la fonction "**Chance**" dans un programme et l'enregistrer sous le nom "**Jeu0**", dans votre dossier de travail.
- 3) Développer la fonction "**Premier**" permettant de vérifier si un nombre, passé comme paramètre, est premier ou non puis l'enregistrer dans votre dossier de travail sous le nom "**Jeu1**".
- 4) Dans le programme "**Jeu1**", ajouter les instructions permettant :
 - D'appeler l'interface graphique intitulée "**Interface_Jeu**" en exploitant l'annexe ci-dessous.
 - D'implémenter un module "**Play**", qui s'exécute à la suite d'un clic sur le bouton "**Jouer**", permettant de récupérer le numéro de téléphone saisi puis d'exploiter la fonction "**Chance**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_Jeu**".

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_()
```

Sujet n°1: Jeu de chance – Correction

Programme jeu0 :

```
def Chance (Ch) :
    if not (Ch.isdecimal() and len ( Ch ) == 8 and Ch[0] in ["2","4","5","9"] ) :
        msg = "Vérifier le numéro de téléphone !"
    else :
        msg = "Désolé, vous n'avez pas gagné."
        s = 0
        for i in range (len ( Ch )) :
            s = s + int ( Ch [i] ) * i
        if Premier (s) :
            msg = "Félicitation, vous avez gagné."
    return msg
```

Programme jeu1 :

Importations à faire pour réaliser une interface graphique

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

Fonction Premier qui permet de vérifier si un nombre n est premier ou non

```
def Premier (n) :
    div=0
    for i in range (1 , n+1) :
        if n % i == 0 :
            div = div + 1
    return div==2
```

Fonction Chance

```
def Chance (Ch) :
    if not (Ch.isdecimal() and len ( Ch ) == 8 and Ch[0] in
["2","4","5","9"] ) :
        msg = "Vérifier le numéro de téléphone !"
    else :
        msg = "Désolé, vous n'avez pas gagné."
        s = 0
        for i in range (len ( Ch )) :
            s = s + int ( Ch [i] ) * i
        if Premier (s) :
            msg = "Félicitation, vous avez gagné."
    return msg
```

#Module Play qui s'exécute à la suite à un clic sur le bouton "Jouer"

```
def Play() :
    Ch=windows.numtel.text()
    msg=Chance(Ch)
    windows.res.setText(msg)
```

```
app = QApplication([])
windows = loadUi ("Interface_Jeu.ui")
windows.show()
windows.Jouer.clicked.connect (Play)
app.exec_()
```

Sujet n° 2 : IBAN

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Un **IBAN** ou numéro international de compte bancaire est une variété de caractères alphanumériques qui identifie de façon distincte, le compte d'un client tenu dans une institution bancaire partout dans le monde.

Exemple: **TU3830004015870002601171** est un numéro **IBAN** où :

TU désigne les initiales du pays du client qui est la Tunisie,

38 est la **clé IBAN**

Le reste des chiffres représente le code **RIB** du client qui est de longueur constante relativement à un pays donné (entre 10 et 30 chiffres). Sachant que **RIB** désigne le **Relevé d'Identité Bancaire** qui permet au titulaire d'un compte bancaire de transmettre ses coordonnées bancaires pour des virements ou des prélèvements.

La clé **IBAN** est obtenue en utilisant le procédé suivant :

- Former une chaîne **ch** composée par les deux premières lettres en majuscules du nom du pays du client auxquelles on ajoute "00" à droite.
- Former un nombre à partir de la chaîne **ch** en remplaçant chaque lettre par le nombre qui lui correspond selon le tableau suivant :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

-Calculer le reste de la division du nombre obtenu par 97

-Soustraire de **98** le reste obtenu. Si le résultat comporte un seul chiffre, insérer un zéro à gauche. Le nombre ainsi obtenu est la clé **IBAN**.

Exemple : Pour un client de la Tunisie, on obtient la chaîne suivante : **TU00**

En remplaçant **T** par **29** et **U** par **30**, on obtient le nombre suivant **293000**.

La clé **IBAN** correspondante à ce client est **38** obtenu comme suit : le reste de la division de 293000 par 97 donne 60 en la retranchant de 98 on obtient $98-60=38$ qui est la **clé IBAN** de la Tunisie.

Notre objectif est de créer un programme qui reçoit en entrée les deux premières lettres du nom du pays d'un client et de son code **RIB** et qui doit générer et afficher le code **IBAN**.

Ci-après, un algorithme de la fonction "**IBAN**" à exploiter pour résoudre le problème posé.

Fonction IBAN (pays, RIB : chaîne): Chaîne

DEBUT

Si NON (long(pays) =2 et pays[0] ∈ ["A".."Z"] et pays [1] ∈ ["A".."Z"]) **Alors**

 msg ← "Vérifier le pays"

Sinon Si NON(Estnum(RIB) et long(RIB) ∈ [10..30]) **Alors**

 msg ← "Vérifier le RIB"

Sinon

 msg ← pays+ cle(pays)+ RIB

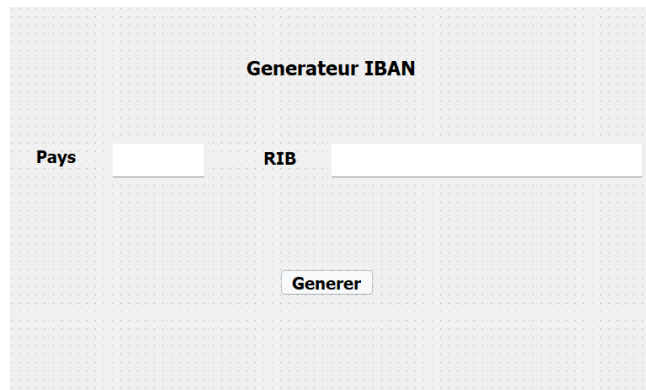
FinSi

Retourner msg

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Générateur IBAN**".
- Un label demandant la saisie le nom d'un pays.
- Une zone de saisie permettant la saisie les deux premières lettres du nom d'un pays.
- Un label demandant la saisie d'un RIB.
- Une zone de saisie permettant la saisie de RIB.
- Un bouton nommé "**Generer**".
- Un label pour afficher un message.



Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_IBAN**".
- 2) Implémenter en Python la fonction "**IBAN**" dans un programme et l'enregistrer sous le nom "**Générateur0**", dans votre dossier de travail.
- 3) Développer la fonction "**cle**" permettant de former la clé IBAN à partir de deux premières lettres du nom de pays et l'enregistrer dans votre dossier de travail sous le nom "**Générateur1**".
- 4) Dans le programme "**Générateur1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_IBAN**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Generation**", qui s'exécute à la suite d'un clic sur le bouton "**Generer**", permettant de récupérer les deux premières lettres du nom de pays et le RIB et puis d'exploiter la fonction "**IBAN**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_IBAN**".

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_()
```

Sujet n° 2 : IBAN – Correction

Programme Generateur1 :

<pre> # Importations à faire pour réaliser une interface graphique from PyQt5.uic import loadUi from PyQt5.QtWidgets import QApplication # Fonction cle qui permet de former la cle IBAN def cle(pays): ch= str(ord(pays[0])-ord("A")+10) + str(ord(pays[1])-ord("A")+10)+"00" n=int(ch) n=98-n%97 if n>=10: ch=str(n) else : ch="0" + str(n) return ch # Fonction IBAN def IBAN(pays, RIB): if not (len(pays)==2 and "A"<=pays[0]<="Z" and "A"<=pays[1]<="Z"): msg=' Verifier le pays' elif not(RIB.isdecimal() and 10<=len(RIB)<=30) : msg=' Verifier le RIB' else: msg=pays+cle(pays)+RIB return msg </pre>	<pre> #Module Generation qui s'exécute à la suite à un clic sur le bouton "generer" def Generation(): pays=windows.pays.text() RIB=windows.RIB.text() msg=IBAN(pays, RIB) windows.resultat.setText(msg) app = QApplication([]) windows = loadUi ("Interface_iban.ui") windows.show() windows.generer.clicked.connect (Generation) app.exec_() </pre>
---	--

Sujet n°3 : Salle de sport

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Un gérant d'une salle de sport veut récompenser les adhérents fidèles en leur offrant un bonus sous forme d'heures d'entraînement gratuites, calculé à partir de leurs numéros d'abonnement.

Le bonus est calculé en fonction de l'ancienneté de l'adhérent, exprimée en nombre de mois par rapport à la date du jour. En effet, une heure supplémentaire est offerte pour chaque mois d'ancienneté, sachant que le bonus ne sera pris en considération que si l'ancienneté dépasse 5 ans.

Un numéro d'abonnement est formé de 10 chiffres répartis comme suit :

- Les 4 premiers chiffres représentent l'année d'adhésion qui doit être comprise entre 2000 et 2022.
- Les 2 suivants représentent le mois d'adhésion dont la valeur doit être comprise entre 1 et 12.
- Les 4 derniers chiffres représentent le numéro d'adhésion qu'on suppose distinct pour tous les adhérents.

Exemple :

Pour le numéro d'abonnement 2016020110, l'adhérent est dont l'année d'adhésion est 2016, le mois d'adhésion est 02 (février) et son numéro d'adhésion est 0110. Le bonus accordé à cet adhérent est de 74 heures. En effet, son ancienneté est égale à six ans et deux mois par rapport à la date d'aujourd'hui (15/04/2022), en nombre de mois elle est égale à 74 ($12 \times 6 + 2$).

Notre objectif est de créer un programme qui reçoit en entrée numéro d'abonnement et qui doit afficher le bonus accordé à cet adhérent sachant que la date du jour à considérer est 15/04/2022.

Ci-après, un algorithme de la fonction **"Fidelite"** à exploiter pour résoudre le problème posé.

Fonction Fidelite (Ch : chaine): Chaine**DEBUT****Si NON Valide (ch)Alors**

msg ← "Vérifier le numéro d'abonnement"

SinonSi Anciennete (ch)<60 alors

msg ← "L'ancienneté de l'adhérent est inférieur à 5 ans"

Sinon

msg ← "L'adhérent à un bonus de " + convch(Anciennete (ch)) + " heures"

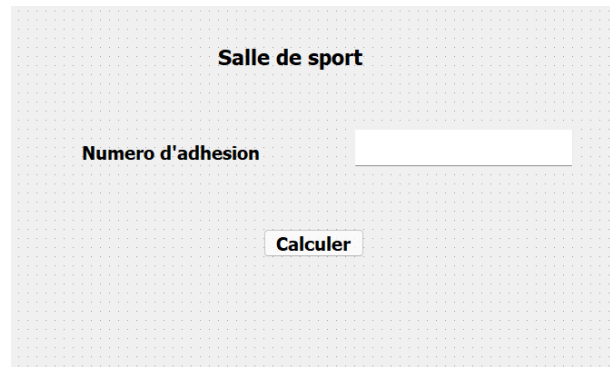
FinSi

Retourner msg

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Salle de sport**".
- Un label demandant la saisie d'un numéro d'adhésion.
- Une zone de saisie permettant la saisie de numéro d'adhésion.
- Un bouton nommé "**Calculer**".
- Un label pour afficher un message.



Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_adhesion**".
- 2) Implémenter en Python la fonction "**Fidelite**" dans un programme et l'enregistrer sous le nom "**Bonus0**", dans votre dossier de travail.
- 3) Développer la fonction **Valide** qui permet de vérifier si un numéro d'abonnement, passé comme paramètre, est valide ou non puis l'enregistrer dans votre dossier de travail sous le nom "**Bonus1**".
- 4) Dans le programme "**Bonus1**", développer la fonction "**Anciennete**" qui prend comme paramètre un numéro d'abonnement et qui permet de calculer le nombre de mois d'ancienneté de l'adhérent.
- 5) Dans le programme "**Bonus1**", ajouter les instructions permettant:
 - a. D'appeler l'interface graphique intitulée "**Interface_adhesion**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Bonus**", qui s'exécute à la suite d'un clic sur le bouton "**Calculer**", permettant de récupérer le numéro d'abonnement et puis d'exploiter la fonction "**Fidelite**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_adhesion**".

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_()
```

Sujet n°3 : Salle de sport- Correction

Programme Bonus1 :

Importations de bibliothèques

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

Calcul de nombre de mois d'ancienneté

```
def Anciennete(ch):
    a=int(ch[:4])
    m=int(ch[4:6])
    mm=4
    aa=2022
    an = aa-a;
    if (mm < m) and (an!=0):
        an = an-1
    if m<= mm :
        nb = an*12+(mm-m)
    else :
        nb = an*12+(12-(m-mm))
    return nb
```

Vérifier si un numéro d'abonnement est valide

def Valide(ch):

```
    a=int(ch[:4])
    m=int(ch[4:6])
    return ch.isdecimal() and len(ch)==10 and 2000<=a<=2022
        and 1<=m<=12
```

procedure fidelite

def Fidelite(ch):

```
    if not Valide(ch):
        msg= "Verifier le numéro d'abonnement"
    elif Anciennete (ch) < 60:
        msg ="L'ancienneté de l'adhérent est inferieur à 5 ans"
    else:
        msg = "L'adhérent à un bonus de " + str(Anciennete (ch))
            +" heures"

    return msg
```

Module qui sera exécuté à la suite d'un click sur le bouton calculer

def Bonus():

```
    ch=windows.n.text()
    msg=Fidelite(ch)
    windows.res.setText(msg)
```

```
app = QApplication([])
```

```
windows=
```

```
loadUi('interface_adhesion.ui')
```

```
windows.calculer.clicked.connect(Bonus)
```

```
windows.show()
```

```
app.exec_()
```

Sujet n°4 : Billet d'avion

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Sur les billets d'avion d'une Campanie aérienne, figure un code de **11** chiffres précédés d'une lettre majuscule. Exemple U19586900462.

Pour vérifier l'authenticité d'un billet, on remplace la lettre du code par son rang alphabétique pour obtenir un nombre de **12** ou de **13** chiffres.

Si le reste de la division par **9** de la somme des chiffres de ce nombre est égale à **8**, ce billet est authentique, sinon c'est un faux billet

Exemple :

Le billet ayant pour code "U19586900462" est authentique. En effet,

- La lettre "U" a pour rang alphabétique 21.
- Le nombre formé sera : "**21**19586900462".
- La somme des chiffres de ce nombre est $2+1+1+9+5+8+6+9+0+0+4+6+2 = 53$.
- Le reste de la division de **53** par **9** est égale à **8**.

Notre objectif est de créer un programme qui reçoit en entrée le code d'un billet et qui doit afficher si le billet est authentique ou non.

Ci-après, un algorithme de la fonction "**Authenticite**" à exploiter pour résoudre le problème posé.

Fonction Authenticite (Ch : chaîne): Chaîne**DEBUT**

Si NON (Estnum(sous-chaîne(ch, 1, long(ch))) et long(ch) =12 et ch[0] ∈["A".."Z"]) **Alors**

msg ← "Vérifier le code de billet"

Sinon

rang ← ord(ch[0]) - ord("A") + 1

Effacer (ch, 0, 1)

ch ← convch(rang) + ch

si valide(ch) **alors**

msg ← "Le billet est authentique"

Sinon

msg ← " C'est un faux billet"

FinSi**FinSi**

Retourner msg

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Billet d'avion**".
- Un label demandant la saisie d'un code de billet.
- Une zone de saisie permettant la saisie de code de billet.
- Un bouton nommé "**Verifier**".
- Un label pour afficher un message.



Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_billet**".
- 2) Implémenter en Python la fonction "**Authenticite**" dans un programme et l'enregistrer sous le nom "**Billet0**", dans votre dossier de travail.
- 3) Développer la fonction "**Valide**" permettant de vérifier si le billet est authentique ou non puis l'enregistrer dans votre dossier de travail sous le nom "**Billet1**".
- 4) Dans le programme "**Billet1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_billet**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Verification**", qui s'exécute à la suite d'un clic sur le bouton "**Verifier**", permettant de récupérer le code de billet et puis d'exploiter la fonction "**Authenticite**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_billet**".

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_()
```

Sujet n°4: Billet d'avion – Correction

Programme billet1 :

Importations à faire pour réaliser une interface graphique

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

Fonction Valide qui permet de vérifier la validité de code de billet

```
def Valide(ch):
    s=0
    for i in range(0, len(ch)):
        s=s+int(ch[i])
    return s%9==8
```

Fonction Authenticite

```
def Authenticite(ch):
    if not(ch[1:len(ch)].isdecimal() and len(ch)==12 and
"A"<=ch[0]<="Z"):
        msg="Verifier le code de billet"
    else:
        rang =ord(ch[0])- ord("A")+1
        ch=ch[1:len(ch)]
        ch=str(rang)+ch
        if valide(ch):
            msg ="Le billet est authentique"
        else:
            msg =" C'est un faux billet"
    return msg
```

#Module Verification qui s'exécute à la suite à un clic sur le bouton "verifier"

```
def Verification():
    ch=windows.code.text()
    msg=Authenticite(ch)
    windows.resultat.setText(msg)

app = QApplication([])
windows = loadUi ("Interface_billet.ui")
windows.show()
windows.verifier.clicked.connect (Verification)
app.exec_()
```

Sujet n° 5 : Décomposition en facteurs premiers

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

La décomposition d'un entier en produit de facteurs premiers consiste à écrire cet entier sous la forme d'un produit de ces diviseurs premiers.

On peut appliquer le principe suivant :

- Vérifier si n est divisible par 2, si oui continuer à le diviser par 2 et le remplacer par $n \div 2$ jusqu'à ce qu'il ne soit plus multiple de 2
- Refaire l'étape précédente pour 3, 4, ...

Exemple : pour $n = 140$, la décomposition en facteurs premier est $140 = 2 * 2 * 5 * 7$

Notre objectif est de créer un programme qui reçoit en entrée un entier n sachant que $n \geq 2$ et qui doit afficher la décomposition en produit de facteurs premiers de n .

Ci-après, un algorithme de la fonction "**Resultat**" à exploiter pour résoudre le problème posé.

Fonction Resultat (Ch : chaîne): Chaîne

DEBUT

Si NON (Estnum(Ch) et valeur(ch) ≥ 2) **Alors**

msg \leftarrow "Vérifier le nombre saisi"

Sinon

n \leftarrow valeur(ch)

msg \leftarrow **FactPremiers(n)**

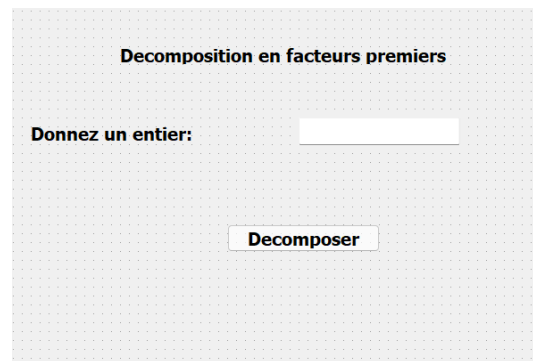
FinSi

Retourner msg

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Décomposition en facteurs premiers**".
- Un label demandant la saisie d'un entier.
- Une zone de saisie permettant la saisie d'un entier.
- Un bouton nommé "**Decomposer**".
- Un label pour afficher un message.

**Travail demandé :**

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_decomposition**".
- 2) Implémenter en Python la fonction "**Resultat**" dans un programme et l'enregistrer sous le nom "**Decomposition0**", dans votre dossier de travail.
- 3) Développer la fonction "**FacPremiers**" qui retourne une chaîne de caractères contenant la décomposition en facteurs premiers d'un entier passé comme paramètre selon le principe décrit ci-dessus puis l'enregistrer dans votre dossier de travail sous le nom "**Decomposition1**".
- 4) Dans le programme "**Decomposition1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_decomposition**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Decomposition**", qui s'exécute à la suite d'un clic sur le bouton "**Decomposer**", permettant de récupérer le nombre à décomposer et puis d'exploiter la fonction "**Resultat**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_decomposition**".

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_()
```

Sujet n°5 : Décomposition en facteurs premiers- Correction**Programme Decomposition1 :****# Importations de bibliothèques**

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

#Fonction FactPremiers qui permet de retourner un message contenant la décomposition en facteurs premiers de n**def FactPremiers(n):**

```
    res=str(n)+"="
    i=2
    while n!=1:
        if n%i==0:
            res=res+str(i)+"*"
            n=n//i
        else:
            i=i+1
    return res[0:len(res)-1]
```

Fonction Resultat**def Resultat(ch):**

```
    if not(ch.isdecimal()) and int(ch)>=2:
        msg= "Verifier le nombre "
    else:
        n=int(ch)
        msg=FactPremiers(n)
    return msg
```

#Module Decomposition qui s'exécute à la suite à un clic sur le bouton "decomposer"**def Decomposition():**

```
    ch= windows.n.text()
    msg=Resultat(ch)
    windows.res.setText(msg)
```

app = QApplication([])

```
windows= loadUi('interface_decomposition.ui')
```

```
windows.decomposer.clicked.connect(Decomposition)
```

```
windows.show()
```

```
app.exec_()
```


Sujet n°6 : Nombre heureux

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Un nombre heureux est un nombre entier qui, lorsqu'on ajoute les carrés de chacun de ses chiffres, puis les carrés des chiffres de ce résultat et ainsi de suite jusqu'à l'obtention d'un nombre à un seul chiffre égal à 1 (un).

Exemple :

N=7 est heureux, puisque :

- $7^2=49$
- $4^2+9^2=97$
- $9^2+7^2=130$
- $1^2+3^2+0^2=10$
- $1^2+0^2=1$

On est arrivé à un nombre d'un seul chiffre qui est égal à 1, donc N=7 est heureux

Entrez un nombre : 8

- $8^2=64$
- $6^2+4^2=52$
- $5^2+2^2=29$
- $2^2+9^2=85$
- $8^2+5^2=89$
- $8^2+9^2=145$
- $1^2+4^2+5^2=42$
- $4^2+2^2=20$
- $2^2+0^2=4$

On est arrivé à un nombre d'un seul chiffre (4) (<10 et $\neq 1$), donc N=8 n'est pas heureux.

(Idem pour tous les nombres non heureux)

Ci-après, un algorithme de la fonction **"Tester"** à exploiter pour résoudre le problème posé.

Fonction Tester (Ch : Chaîne) : Chaîne

DEBUT

Si NON (Estnum(Ch)) **Alors**

msg ← "Vérifier le nombre saisi"

Sinon

n ← valeur(ch)

Si Heureux (n) **Alors**

msg ← "Le nombre est heureux"

Sinon

msg ← "Le nombre n'est pas heureux"

FinSi

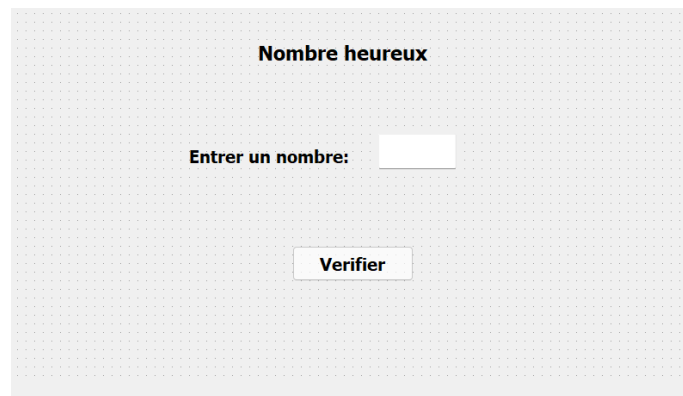
FinSi

Retourner msg

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant un titre "**Nombre heureux**".
- Un label demandant la saisie d'un nombre.
- Une zone de saisie permettant la saisie d'un nombre.
- Un bouton nommé "**Verifier**".
- Un label pour afficher un message.



Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface**".
- 2) Implémenter en Python la fonction "**Tester**" dans un programme et l'enregistrer sous le nom "**test0**", dans votre dossier de travail.
- 3) Développer la fonction "**Heureux**" permettant de vérifier si un nombre, passé comme paramètre, est heureux ou non puis l'enregistrer dans votre dossier de travail sous le nom "**test1**".
- 4) Dans le programme "**test1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Verification**", qui s'exécute à la suite d'un clic sur le bouton "**Verifier**", permettant de récupérer le nombre saisi puis d'exploiter la fonction "**Tester**" afin d'afficher le message retourné via un **label** de l'interface "**Interface**".

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_()
```

Sujet n°6 : Nombres heureux- Correction

Programme test1 :

Importations à faire pour réaliser une interface graphique

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

Fonction heureux qui permet de vérifier si un nombre est heureux

```
def heureux(n):
    ch=str(n)
    test=False
    limite=False
    while limite == False and test == False:
        s=0
        for i in range(len(ch)):
            s=s+int(ch[i])*int(ch[i])
        if s == 1:
            test=True
        if s < 10:
            limite=True
        ch = str(s)
    return test
```

Fonction tester:

```
def tester(ch):
    if not ch.isdecimal():
        msg=' Verifier le nombre saisi'
    else :
        n=int(ch)
        if heureux(n) :
            msg=' Le nombre est heureux'
        else:
            msg='Le nombre n'est pas heureux'
    return msg
```

#Module verification qui s'exécute à la suite à un clic sur le bouton "Verifier"

```
def verification():
    ch=windows.nombre.text()
    msg=tester(ch)
    windows.resultat.setText(msg)
```

```
app = QApplication([])
windows = loadUi ("Interface.ui")
windows.show()
windows.verifier.clicked.connect (verification)
app.exec_()
```

Sujet n°7 : Produit puissance

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Etant donné un entier N qui vérifie la propriété suivante :

"Le **produit** des **diviseurs** de N sauf lui-même est égal à une **puissance** de N avec un **exposant** strictement supérieur à 0".

Exemples :

- $N = 6$ vérifie cette propriété car le **produit** de ses diviseurs sauf lui-même est égal à 6 ($1 * 2 * 3 = 6$) qui est une **puissance** de 6, avec un **exposant** égal à 1 (car $6 = 6^1$).
- $N = 12$ vérifie cette propriété car le **produit** de ses diviseurs sauf lui-même est égal à 144 ($1 * 2 * 3 * 4 * 6 = 144$) qui est une **puissance** de 12, avec un **exposant** égal à 2 (car $144 = 12^2$).
- $N = 30$ vérifie cette propriété car le **produit** de ses diviseurs sauf lui-même est égal à 27000 ($1 * 2 * 3 * 5 * 6 * 10 * 15 = 27000$) qui est une **puissance** de 30, avec un **exposant** égal à 3 (car $27000 = 30^3$).
- $N = 9$ ne vérifie pas cette propriété car le **produit** de ses diviseurs sauf lui-même est égal à 3 ($1 * 3 = 3$) qui n'est pas une **puissance** de 9.
- $N = 11$ ne vérifie pas cette propriété car le **produit** de ses diviseurs sauf lui-même est égal à 1 qui est une **puissance** de 11, avec un **exposant** égal à 0.

Notre objectif est de créer un programme qui reçoit en entrée un entier N strictement supérieur à 1 et qui doit déterminer si cet entier vérifie la propriété décrite ci-dessus ou non.

Ci-après, un algorithme de la fonction "**Resultat**" à exploiter pour résoudre le problème posé.

Fonction Resultat (ch : chaîne): Chaîne

DEBUT

Si NON (Estnum(ch) et valeur(ch) > 1) **Alors**

msg ← "Vérifier le nombre"

Sinon

msg ← "Le nombre ne vérifie pas la propriété"

s ← valeur (ch)

p ← Produit(s)

puiss ← s

Tantque puiss < p **faire**

puiss ← puiss*s

FinTantque

Si puiss = p **Alors**

msg ← "Le nombre vérifie la propriété"

FinSi

FinSi

Retourner msg

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Produit Puissance**".
- Un label demandant la saisie d'un entier.
- Une zone de saisie permettant la saisie d'un entier.
- Un bouton nommé "**Verifier**".
- Un label pour afficher un message.



Travail demandé :

- 5) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_produit**".
- 6) Implémenter en Python la fonction "**Resultat**" dans un programme et l'enregistrer sous le nom "**test0**", dans votre dossier de travail.
- 7) Développer la fonction "**Produit**" permettant de calculer le produit des diviseurs, d'un entier passé comme paramètre, sauf lui-même et l'enregistrer dans votre dossier de travail sous le nom "**Test1**".
- 8) Dans le programme "**Test1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_produit**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Verification**", qui s'exécute à la suite d'un clic sur le bouton "**Verifier**", permettant de récupérer un nombre saisi et puis d'exploiter la fonction "**Resultat**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_produit**".

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_()
```

Sujet n°7: Produit puissance – Correction

Programme test1 :

Importations à faire pour réaliser une interface graphique

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

Fonction Produit qui permet de calculer le produit de diviseurs de s sauf lui meme

```
def produit(s):
    p=1
    for i in range(1, s//2+1):
        if s%i==0:
            p=p*i
    return p
```

Fonction Resulta

```
def Resultat(ch):
    if not (ch.isdecimal() and int(ch)>1):
        msg="Verifier le nombre"
    else:
        msg="le nombre ne verifie pas la propriété"
        s=int(ch)
        p=produit(s)
        puiss=s
        while puiss<p:
            puiss=puiss*s
        if puiss==p:
            msg="le nombre verifie la propriété"
    return msg
```

#Module Verification qui s'exécute à la suite à un clic sur le bouton "verifier"

```
def verification():
    ch=windows.n.text()
    msg=Resultat(ch)
    windows.res.setText(msg)
```

```
app = QApplication([])
windows= loadUi('interface_produit.ui')
windows.verifier.clicked.connect(verification)
windows.show()
app.exec_()
```

Sujet n°8 : Nombre unitairement Parfait

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Un entier N est dit **unitairement parfait** s'il est égal à la somme de ses **diviseurs unitaires** sauf lui-même.

On appelle **diviseur unitaire** d'un entier N , tout entier D qui vérifie les conditions suivantes :

- D est un diviseur de N .
- D et $(N \text{ Div } D)$ sont premiers entre eux.

NB : Deux nombres sont dits **premiers entre eux** si leur plus grand commun diviseur (PGCD) est égal à 1.

Exemple 1 : Pour $N = 36$,

N n'est pas un entier unitairement parfait car il n'est pas égal à la somme de ses diviseurs unitaires :

<i>Les diviseurs de 36</i>	1	2	3	4	6	9	12	18
<i>36 DIV diviseur</i>	36	18	12	9	6	4	3	2
<i>Test de primalité entre eux</i>	oui	non	non	oui	non	oui	non	non
<i>Les diviseurs unitaires de 36</i>	1			4		9		
<i>La somme des diviseurs unitaires de 36</i>	14 ($\neq 36$)							

Exemple 2 : Pour $N = 60$

N est un entier unitairement parfait car il est égal à la somme de ses diviseurs unitaires :

<i>Les diviseurs de 60</i>	1	2	3	4	5	6	10	12	15	20	30
<i>60 DIV diviseur</i>	60	30	20	15	12	10	6	5	4	3	2
<i>Test de primalité entre eux</i>	oui	non	oui	oui	oui	non	non	oui	oui	oui	non
<i>Les diviseurs unitaires de 60</i>	1		3	4	5			12	15	20	
<i>La somme des diviseurs unitaires de 60</i>	60										

Notre objectif est de créer un programme qui reçoit en entrée un entier N strictement supérieur à 1 et qui doit vérifier si cet entier est unitairement parfait.

Ci-après, un algorithme de la fonction "**Resultat**" à exploiter pour résoudre le problème posé.

```

Fonction Resultat (ch : chaîne): Chaîne
DEBUT
  Si NON( Estnum(ch) et valeur(ch) > 1) Alors
    msg ← "Vérifier le nombre"
  Sinon
    msg ← "Le nombre ne pas unitairement parfait"
    s ← valeur (ch)
    div ← 0
    Pour i de 1 à s div 2 faire
      Si (s mod i = 0) et Primalité(i, s div i) Alors
        div ← div + i
      FinSi
    FinPour
    Si s = div Alors
      msg ← "Le nombre est unitairement parfait"
    FinSi
FinSi
Retourner msg
  
```

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Nombre unitairement parfait**".
- Un label demandant la saisie d'un entier.
- Une zone de saisie permettant la saisie d'un entier.
- Un bouton nommé "**Vérifier**".
- Un label pour afficher un message.



Travail demandé :

- 9) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_ parfait**".
- 10) Implémenter en Python la fonction "**Resultat**" dans un programme et l'enregistrer sous le nom "**Parfait0**", dans votre dossier de travail.
- 11) Développer la fonction "**Primalité**" permettant de vérifier si deux entiers, passés comme paramètres sont premiers entre eux ou non et l'enregistrer dans votre dossier de travail sous le nom "**Parfait1**".
- 12) Dans le programme "**Parfait1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_parfait**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**UnitParfait**", qui s'exécute à la suite d'un clic sur le bouton "**Vérifier**", permettant de récupérer un nombre saisi et puis d'exploiter la fonction "**Resultat**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_parfait**".

Sujet n°8: Nombre unitairement parfait – Correction

Programme parfait1 :

<pre> # Importations à faire pour réaliser une interface graphique from PyQt5.uic import loadUi from PyQt5.QtWidgets import QApplication # Fonction Prim qui permet de vérifier si deux nombres sont premiers entre eux def prim(i,a): while a!=i: if a>i: a=a-i else: i=i-a return i==1 # Fonction Resultat def resultat(ch): if not(ch.isdecimal() and int(ch)>1): msg="verifier le nombre" else: s=int(ch) div=0 for i in range(1,s): if s%i==0 and prim(i,s/i): div=div+i if s==div: msg="le nombre est unitairement parfait" else: msg="le nombre pas unitairement parfait" return msg </pre>	<pre> #Module unitParfait qui s'exécute à la suite à un clic sur le bouton "verifier" def unitparfait(): ch=windows.nombre.text() msg=resultat(ch) windows.res.setText(msg) app = QApplication([]) windows = loadUi ("Interface_parfait.ui") windows.show() windows.verifier.clicked.connect (unitParfait) app.exec_() </pre>
--	---

Sujet n°9 : Code de César

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Le codage de César est une manière de crypter un message de manière simple : On choisit un nombre (appelé clé de codage) et on décale toutes les lettres de notre message du nombre choisi.

Par exemple : Si je choisis comme clé le nombre 2. Alors la lettre A deviendra C, le B deviendra D ... et le Z deviendra B. Ainsi, le mot MATHS deviendra une fois codé OCVJU.

Notre objectif est de créer un programme qui reçoit en entrée une clé de codage n (qui peut être négative) et un message en majuscule sans accent à coder et qui doit afficher le message codé correspondant.

Ci-après, un algorithme de la fonction "**Cesar**" à exploiter pour résoudre le problème posé.

```

Fonction Cesar (Ch, n : chaine): Chaine
DEBUT
    i ← 0
    alpha ← vrai
    Tantque i < long(Ch) et alpha = vrai faire
        alpha ← Ch[i] ∈ ["A".."Z"]
        i ← i + 1
    FinTantque
    Si alpha = faux Alors
        msg ← "Vérifier le message à crypter !"
    Sinon Si NON (Estnum(n)) Alors
        msg ← "Vérifier la clé de codage !"
    Sinon
        c ← valeur(n)
        msg ← Coder (Ch, c)
    FinSi
    Retourner msg
FIN

```

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Code de césar**".
- Un label demandant la saisie de la chaine à crypter.
- Une zone de saisie permettant la saisie de la chaine à crypter.
- Un label demandant la saisie de la clé de codage.
- Une zone de saisie permettant la saisie de la clé de codage.
- Un bouton nommé "**Crypter**".
- Un label pour afficher un message.



The image shows a graphical user interface for a Caesar cipher program. The title is "Code de César". There are two input fields: "Message" and "Clé". Below these fields is a button labeled "Crypter".

Travail demandé :

- 13) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_codage**".
- 14) Implémenter en Python la fonction "**Cesar**" dans un programme et l'enregistrer sous le nom "**Codage0**", dans votre dossier de travail.
- 15) Développer la fonction "**Coder**" qui prend comme paramètres une chaîne de caractères et une clé de codage et qui permet de coder cette chaîne selon le principe décrit ci-dessus puis l'enregistrer dans votre dossier de travail sous le nom "**Codage1**".
- 16) Dans le programme "**Codage1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_codage**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Cryptage**", qui s'exécute à la suite d'un clic sur le bouton "**Crypter**", permettant de récupérer le message à crypter et la clé de codage puis d'exploiter la fonction "**Cesar**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_codage**".

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_()
```

Sujet n°9 : Code de César- Correction

Programme Codage1 :

Importations de bibliothèques

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

#Fonction Coder qui permet de crypter un message

def Coder(ch,c):

```
    res=""
    for i in range(0, len(ch)):
        asc=ord(ch[i])+c
        if asc>ord("Z"):
            asc=asc-26
        if asc<ord("A"):
            asc=asc+26
        res=res+chr(asc)
    return res
```

Fonction Cesar

def Cesar(ch, n):

```
    alpha=True
    i=0
    while i<len(ch) and alpha :
        alpha="A"<=ch[i]<="Z"
        i=i+1
    if alpha==False:
        msg=" Verifier le message à crypté"
    elif not n.isdecimal():
        msg="Verifier la clé de codage"
    else:
        c=int(n)
        msg=Coder(ch, c)
    return msg
```

#Module Cryptage qui s'exécute à la suite à un clic sur le bouton "crypter"

def Cryptage():

```
    ch= windows.message.text()
    n= windows.n.text()
    msg=Cesar(ch, n)
    windows.res.setText(msg)
```

app = QApplication([])

```
windows= loadUi('interface_codage.ui')
```

```
windows.crypter.clicked.connect(Cryptage)
```

```
windows.show()
```

```
app.exec_()
```

Sujet n° 10 : Cryptage

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

Dans le but de sécuriser les messages à envoyer, on peut faire appel à une méthode de cryptage. Une des méthodes utilisées consiste à remplacer chaque lettre du message à crypter par celle qui la suit de **p** positions dans l'alphabet français, où **p** désigne le nombre de mots du message.

NB :

- On suppose que le caractère qui suit la lettre "Z" est le caractère "A" et celui qui suit la lettre "z" est le caractère "a".
- Le caractère espace ne subit aucune modification.

Exemple :

Pour le message "Examen Pratique En Informatique"

Etant donné que le message à crypter est formé de **4** mots, pour la lettre alphabétique "E" par exemple, elle sera remplacée par "I".

En continuant à appliquer ce principe de codage, le message crypté sera :

"Ibeqir Tvexmuyi Ir Mrjsvqexmuyi"

Ci-après, un algorithme de la fonction "CrypterMessage" à exploiter pour résoudre le problème posé.

Fonction CrypterMessage (ch : Chaîne) : Chaîne**DEBUT**

```

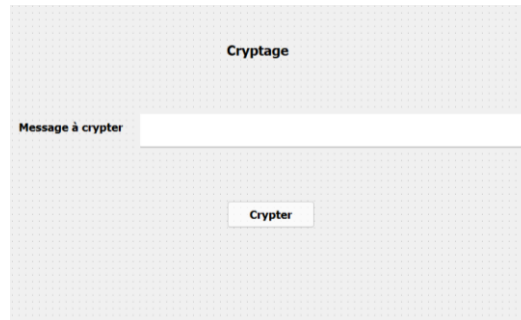
Si NON Valide(ch) alors
    msg ← " Vérifier votre message!"
Sinon
    msg ← ""
    p ← NombreMots(ch)
    Pour i de 0 à Long(ch)-1 Faire
        Si ch[i]="_" alors
            msg ← msg+"_"
        Sinon Si ord (Majus(ch[i]))+p ≤ ord("Z") alors
            msg ← msg+chr(ord(ch[i])+p)
        Sinon
            msg ← msg+chr(ord(ch[i])+p-26)
    Fin Si
    Fin Pour
    Fin Si
    Retourner msg

```

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Cryptage d'un message**"
- Un label demandant la saisie du message à crypter
- Une zone de saisie permettant la saisie du message à crypter
- Un bouton nommé "**Crypter**"
- Un label pour afficher un message.



Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_cryptage**".
- 2) Implémenter en Python la fonction "**CrypterMessage**" dans un programme et l'enregistrer sous le nom "**prog0**", dans votre dossier de travail.
- 3) Développer la fonction "**Valide**" qui permet de vérifier si une chaîne de caractères, passée comme paramètre, est non vide, formée uniquement par des lettres et des espaces, ne commence pas par un espace, ne se termine pas par un espace et entre deux mots il y a une seule espace. Enregistrer le programme dans votre dossier de travail sous le nom "**prog1**".
- 4) Dans le programme "**prog1**", développer la fonction "**NombreMots**" qui permet de retourner le nombre de mots dans une chaîne de caractères passée comme paramètre.
- 5) Dans le programme "**Prog1**" ajouter les instructions permettant :
 - D'appeler l'interface graphique intitulée "**Interface_cryptage**" en exploitant l'annexe ci-dessous.
 - D'implémenter un module "**Cryptage**" qui s'exécute à la suite d'un clic sur le bouton "**Crypter**", permettant de récupérer un message saisi puis d'exploiter la fonction "**CrypterMessage**" afin d'afficher le message retourné via un label de l'interface "**Interface_cryptage**".

ANNEXE

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

Sujet n°10 : Cryptage - Correction

Programme Prog1 :

Importations de bibliothèques

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
```

#Vérifier la validité de la chaîne ch :

def Valide(ch):

```
    test=True
    i=0
    while i<len(ch) and test :
        test="A"<=ch[i].upper()<="Z" or ch[i]==" "
        i=i+1
    return ch!=" " and test and ch[0]!=" " and ch[-1]!=" " and
           ch.find(" ")!=-1
```

Retourner le nombre de mots dans ch :

def NombreMots(ch):

```
    nb=1
    while ch.find(" ")!=-1:
        ch=ch[ch.find(" ")+1:]
        nb=nb+1
    return nb
```

Fonction CrypterMessage

def CrypterMessage(ch):

```
    if not Valide(ch):
        msg=" Verifier le message à crypté"
    else:
        msg=""
        p=NombreMots(ch)
        for i in range(0, len(ch)):
            if ch[i]==" ":
                msg=msg+" "
            elif ord(ch[i].upper())<=ord("Z"):
                msg=msg+ chr(ord(ch[i])+p)
            else:
                msg=msg+ chr(ord(ch[i])+p-26)
        return msg
```

#Module Cryptage qui s'exécute à la suite à un clic sur le bouton "crypter"

def Cryptage():

```
    ch= windows.message.text()
    msg=CrypterMessage(ch)
    windows.res.setText(msg)
```

app = QApplication([])

```
windows= loadUi('interface_cryptage.ui')
```

```
windows.crypter.clicked.connect(Cryptage)
```

```
windows.show()
```

```
app.exec_()
```

Sujet n° 11 : Décomposition en facteurs premiers avec Tableau

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

La décomposition d'un entier en produit de facteurs premiers consiste à écrire cet entier sous la forme d'un produit de ces diviseurs premiers.

On peut appliquer le principe suivant :

- Vérifier si n est divisible par 2, si oui continuer à le diviser par 2 et le remplacer par $n \div 2$ jusqu'à ce qu'il ne soit plus multiple de 2
- Refaire l'étape précédente pour 3, 4, ...

Exemple : pour $n = 140$, la décomposition en facteurs premier est : $140 = 2 * 2 * 5 * 7$

Notre objectif est de créer un programme qui reçoit en entrée un entier n sachant que $n \geq 2$ et qui doit afficher la décomposition en produit de facteurs premiers de n .

Ci-après, un algorithme de la procédure **"Resultat"** à exploiter pour résoudre le problème posé.

Procédure Resultat (Ch : chaîne, @Fact :tab, @msg :chaîne)

DEBUT

Si NON (Estnum(Ch) et valeur(Ch) ≥ 2) **Alors**

msg \leftarrow "Vérifier le nombre saisi"

Sinon

n \leftarrow valeur(Ch)

RemplirFact(Fact, f, n)

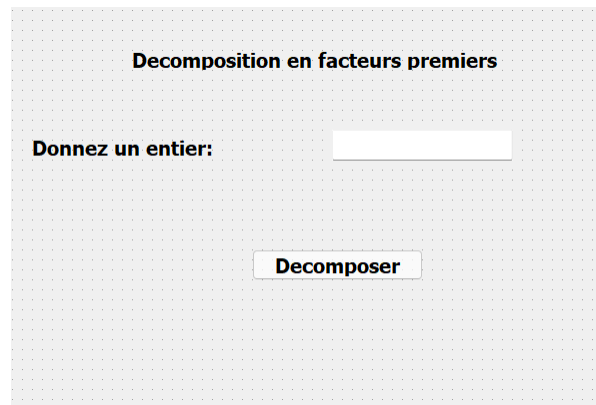
msg \leftarrow **FactPremiers(Fact, f, n)**

FinSi

FIN

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre **"Décomposition en facteurs premiers"**.
- Un label demandant la saisie d'un entier.
- Une zone de saisie permettant la saisie d'un entier.
- Un bouton nommé **"Decomposer"**.
- Un label pour afficher un message.

**Travail demandé :**

- 17) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_decomposition**".
- 18) Implémenter en Python la procédure "**Resultat**" dans un programme et l'enregistrer sous le nom "**Decomposition0**", dans votre dossier de travail.
- 19) Développer la procédure **RemplirFact** qui permet de chercher et ranger les facteurs premiers dans le tableau **Fact**. Le paramètre **f** contiendra le nombre de facteurs premiers. Enregistrer le programme dans votre dossier de travail sous le nom "**Decomposition1**".
- 20) Dans le programme "**Decomposition1**", développer la fonction "**FactPremiers**" qui prend comme paramètres le tableau **Fact** et le nombre de facteurs **f**, et qui retourne une chaîne de caractères contenant les facteurs premiers séparés par le caractère " *".
- 21) Dans le programme "**Decomposition1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_decomposition**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Decomposition**", qui s'exécute à la suite d'un clic sur le bouton "**Decomposer**", permettant de récupérer le nombre à décomposer et puis d'exploiter la procédure "**Resultat**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_decomposition**".

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_()

```

Sujet n° 11 : Décomposition en facteurs premiers avec Tableau - Correction**Programme Decomposition1 :****# Importations de bibliothèques**

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
from numpy import array
```

#Declaration de tableau Fact

```
Fact=array([int()]*30)
```

#Retourner un message contenant la décomposition en facteurs premiers**def FactPremiers(Fact,f, n):**

```
res= str(n)+"="
for i in range(0, f):
    res = res+str(Fact[i]) + "*"
return res[0:len(res)-1]
```

Remplir le tableau Fact par les facteurs premiers de n :**def RemplirFact(Fact, n):**

```
global f
i=2
f=0
while n!=1:
    if n%i==0:
        Fact[f]=i
        f=f+1
        n=n//i
    else:
        i=i+1
```

Procedure Resultat:**def Resultat(ch, Fact):**

```
global msg
if not(ch.isdecimal() and 2<=int(ch)):
    msg= "Verifier le nombre "
else:
    n=int(ch)
    RemplirFact(Fact, n)
    msg= FactPremiers(Fact, f, n)
```

#Module Decomposition qui s'exécute à la suite à un clic sur le bouton "decomposer"**def Decomposition():**

```
ch= windows.n.text()
Resultat(ch, Fact)
windows.res.setText(msg)
```

```
app = QApplication([])
windows= loadUi('interface_decomposition.ui')
windows.decomposer.clicked.connect(Decomposition)
windows.show()
app.exec_()
```

Sujet n° 12 : Multiplication Russe avec Tableaux

Important : Dans le répertoire **Bac2022**, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solutions de ce sujet.

La « **multiplication Russe** » est une méthode particulière permettant la multiplication de deux entiers A et B en utilisant seulement la multiplication par 2, la division par 2 et l'addition selon le principe suivant :

- Le premier nombre A est divisé par 2 (division entière) et le deuxième B est multiplié par 2.
- Le processus se répète jusqu'à avoir 1 comme valeur pour le premier nombre A.
- Les différentes valeurs de A et B seront arranger respectivement dans les tableaux D et M.
- Le résultat est la somme des éléments du tableau M qui sont en face des éléments impairs du tableau D.

Exemple : Pour A= 11 et B= 13 :

Les tableaux générés D et M:

D	11	5	2	1
	↓	↓		↓
M	13	26	52	104

Le résultat du produit de 11 par 13 est égal à : $13 + 26 + 104 = 143$

Notre objectif est de créer un programme qui reçoit en entrée deux entiers strictement positifs et qui doit calculer et afficher leur produit par la multiplication Russe.

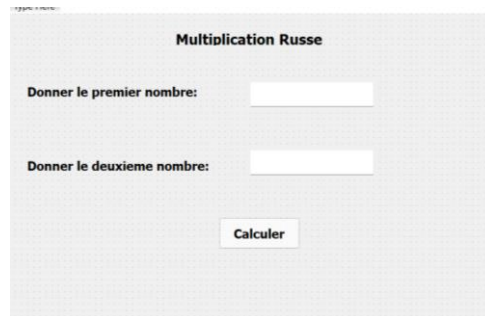
Ci-après, un algorithme de la fonction "**Produit**" à exploiter pour résoudre le problème posé.

```

Fonction Produit(ch1, ch2 : chaîne) : chaîne
DEBUT
    Si NON (Estnum(ch1) et valeur (ch1) > 0) Alors
        msg ← "Vérifier A"
    Sinon Si NON (Estnum(ch2) et valeur (ch2) > 0) Alors
        msg ← "Vérifier B"
    Sinon
        A ← valeur (ch1)
        B ← valeur (ch2)
        Generer (A, B, D, M, N)
        msg ← ch1 + "*" + ch2 + "=" + convch (Calculer (D, M, N))
    FinSi
    Retourner msg
FIN
  
```

On désire créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le titre "**Multiplication Russe**"
- Un label demandant la saisie de premier nombre.
- Une zone de saisie permettant la saisie de premier nombre.
- Un label demandant la saisie de deuxième nombre.
- Une zone de saisie permettant la saisie de deuxième nombre.
- Un bouton nommé "**Calculer**".
- Un label pour afficher un message.



Travail demandé :

- 22) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface_multiplication**".
- 23) Implémenter en Python la fonction "**Produit**" dans un programme et l'enregistrer sous le nom "**Multiplication0**", dans votre dossier de travail.
- 24) Développer la procédure **Generer** qui permet d'arranger les différentes valeurs de A et B respectivement dans les tableaux D et M selon le principe décrit ci-dessus. **Le paramètre N** contiendra la taille des tableaux D et M. Enregistrer le programme dans votre dossier de travail sous le nom "**Multiplication1**".
- 25) Dans le programme "**Multiplication1**", développer la fonction "**Calculer**" qui prend comme paramètres les tableaux D et M et leur taille N et qui retourne le produit de A et B selon le principe décrit ci-dessus.
- 26) Dans le programme "**Multiplication1**", ajouter les instructions permettant :
 - a. D'appeler l'interface graphique intitulée "**Interface_multiplication**" en exploitant l'annexe ci-dessous.
 - b. D'implémenter un module "**Multiplication**", qui s'exécute à la suite d'un clic sur le bouton "**Calculer**", permettant de récupérer les deux nombres à multiplier puis d'exploiter la fonction "**Produit**" afin d'afficher le message retourné via un **label** de l'interface "**Interface_multiplication**".

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_()
```

Sujet n° 12 : Multiplication Russe avec tableaux - Correction**Programme Multiplication1 :****# Importations de bibliothèques**

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
from numpy import array
```

#Déclaration des tableaux D et M

```
D=array([int()]*50)
M=array([int()]*50)
```

#Retourner le produit de A et B**def Calculer(D, M, N):**

```
s=0
for i in range(0, N):
    if D[i]%2 != 0:
        s = s+ M[i]
return s
```

Générer les tableaux D et M à partir de A et B**def Generer(A, B, D, M):**

```
global N
N=0
while A!=1:
    D[N]=A
    M[N]=B
    A=A//2
    B=B*2
    N=N+1
D[N]=A
M[N]=B
N=N+1
```

Procedure Produit:**def Produit (ch1, ch2):**

```
if not(ch1.isdecimal() and int(ch1)>0):
    msg= "Verifier A"
elif not(ch2.isdecimal() and int(ch2)>0):
    msg= "Verifier B"
else:
    A=int(ch1)
    B=int(ch2)
    Generer(A, B, D, M)
    msg=ch1+"*" +ch2+"="+str(Calculer(D,M,N))
return msg
```

#Module Multiplication qui s'exécute à la suite à un clic sur le bouton "calculer"**def Multiplication():**

```
ch1= windows.A.text()
ch2= windows.B.text()
msg=Produit(ch1, ch2)
windows.res.setText(msg)
```

app = QApplication([])

```
windows= loadUi('interface_Multiplication.ui')
windows.calculer.clicked.connect(Multiplication)
windows.show()
app.exec_()
```