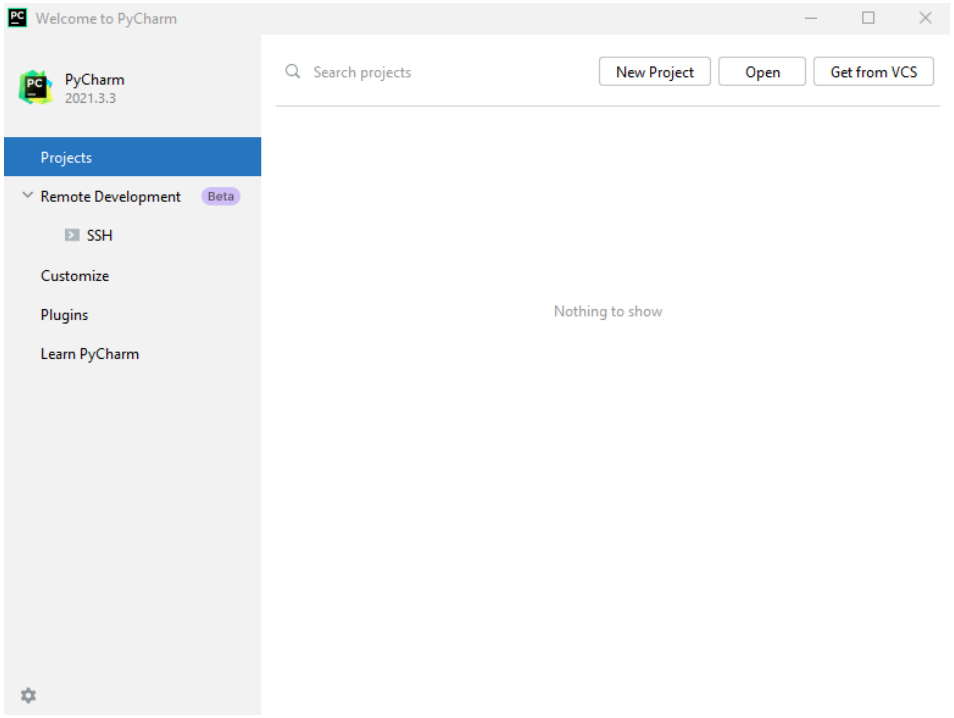
TD 1 : Introduction à Python

Projet Programmation - Geipi 2A IT

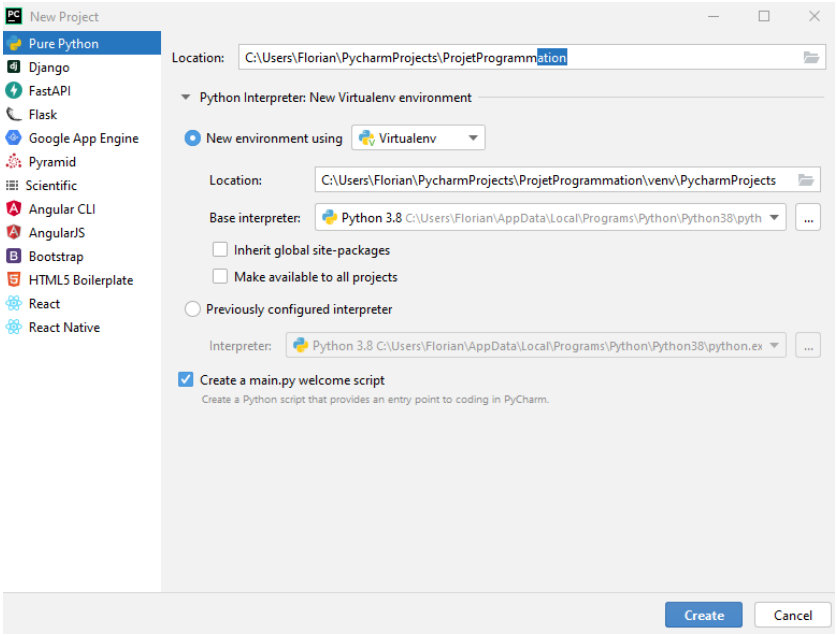
ESIREM GEIPI2 IT - Avril 2022

1 Découverte de l'IDE PyCharm

1.1 Configuration d'un nouveau projet

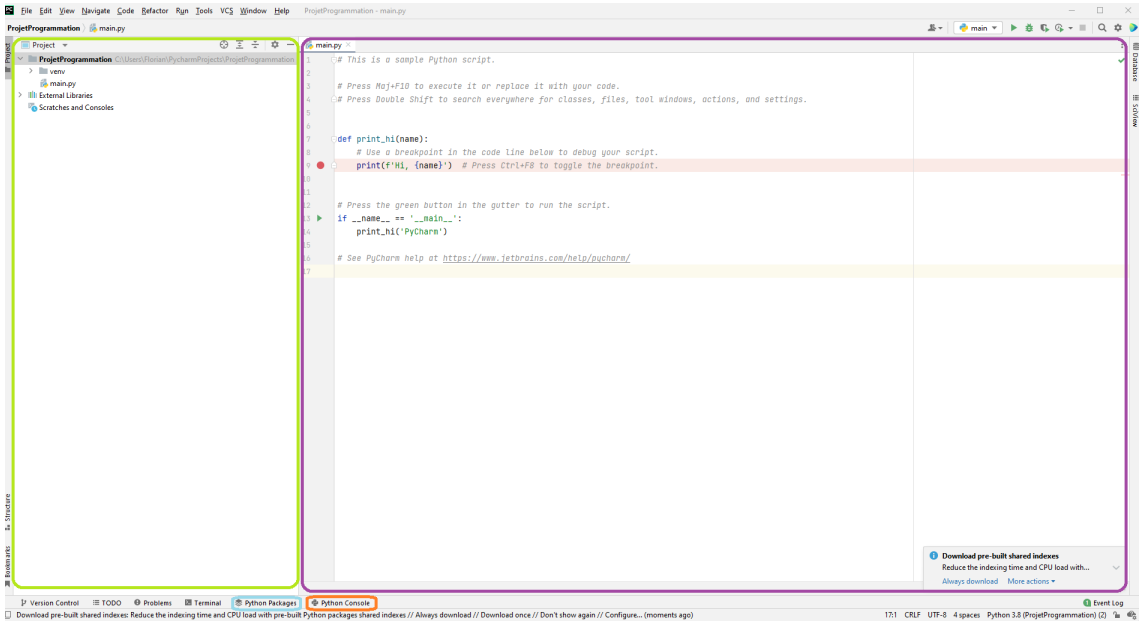


Ouvrir l'IDE de programmation PyCharm et créer un nouveau projet via le bouton New Project (Figure 1) puis créer un nouvel environnement virtuel en spécifiant Python en version 3.X.. Un environnement virtuel permet de créer un espace de travail séparé avec des versions de dépendances différentes des autres espaces de travail.



Une fois, l'environnement crée, l'IDE s'affiche. Pour finaliser l'initialisation du projet, incluez, pour la suite des TDs, à votre environnement virtuel les packages élémentaires numpy et pygame via le gestionnaire de package (Figure 3 - Zone bleu.). Il vous reste plus qu'a importé les packages Python à votre fichier Python avec les instructions suivantes :

```
import numpy
import pygame
```



2 Langage Python

Le langage Python, créé en 1991 par Guido van Rossum, est un langage orienté-objet, interprété (opposé aux langages compilés comme le C/C++, Java, ...) et multi-plateforme. Le haut niveau d'abstraction du langage ainsi qu'une myriade de package proposée par la communauté de programmeur font du langage Python l'un des plus utilisés au monde. Son usage s'est accru dans le domaine de l'intelligence artificielle avec l'apparition de librairie spécialisée comme Tensorflow ou Pytorch proposée respectivement par Google et Facebook.

2.1 Variable & Manipulation des chaîne de caractère

Une variable définie par un nom, est une zone mémoire de l'ordinateur dans laquelle une valeur est stockée. Contrairement à d'autres langages de programmation, l'espace mémoire de la variable est dynamiquement alloué, par l'interpréteur Python, en fonction de la valeur assignée. Le langage Python étant un langage orienté objet, il est possible de connaître le type d'une variable avec la fonction **type(*var*)**. Il se peut que l'on souhaite spécifier ou le modifier. Cela peut être fait avec le casting d'une variable vers un type donné.

- **int(*var*)** Créez un objet de type entier à partir d'une valeur flottante, ou d'une chaîne de caractère numérique.
- **float(*var*)** Créez un objet de type flottant à partir d'une valeur entière, ou d'une chaîne de caractère numérique.
- **str(*var*)** Créez un objet chaîne de caractère à partir d'un nombre.

La plupart des programmes informatiques requièrent, à un moment donné, une interaction avec l'utilisateur soit par une entrée clavier ou pour visualiser un résultat ou l'état d'une variable.

La fonction **print(*str*, *str2*, ...)** permet d'afficher une ou plusieurs chaînes de caractère sur la console tandis qu'une entrée clavier est capturée par la fonction **input(*texte*)**.

Le **Chiffre de César** est l'un des plus anciens algorithmes de Cryptographie où chaque lettre est substituée par une autre lettre suivant un décalage de K lettres dans l'alphabet.
Le chiffrement est régi par l'équation suivante :

$$E = (C + K) \, mod \, 26 \tag{1}$$

Équation du déchiffrement:

$$C = (E - K) \, mod \, 26 \tag{2}$$

1. Proposez un script de chiffrage et un de déchiffrage où l'utilisateur entre un caractère et un nombre correspondant au décalage, puis afficher le résultat. **Help :** La fonction **ord(*var*)** retourne la valeur ASCII d'un caractère. La méthode **chr(*var*)** réalise l'opération inverse.

2.2 Syntaxe des structure de base

2.2.1 If - Elif - else

Listing 1: Exemple d'une structure If - Elif - else

```
import datetime
birthOfYear = 2002
currentYear = datetime.datetime.now().year
diff = currentYear - birthOfYear
if diff >= 18:
    print("La personne est majeure")
elif diff < 18 and diff >= 0 :
    print("La personne est mineure")
else:
    print("Back to the futur")
```

2.2.2 Boucle For

La boucle For est une méthode d'itération, sur un nombre fini d'opération, puissante en Python. Similairement, au C++, la boucle For permet d'itérer entre deux nombres suivant un certain pas en utilisant la fonction **range(start, end, incrément)** mais également permet d'itérer sur une chaîne de caractère ou une énumération sans que le programmeur ait besoin de spécifier les dimensions des éléments. Ci-dessous, vous trouverez un exemple de deux boucles For.

Listing 2: Exemple de deux boucles For

```
# Chaine de caractere:
a = "Hello World"
# Pour i allant de 0 a la longueur de la chaine de car
for i in range(len(a)):
    print(f"Le caractere {a[i]} est a la position {i}")
i = 0
for c in a:
    print(f"Le caractere {c} est a la position {i}")
    i = i+1
```

2. Modifiez les scripts de la question 1 afin qu'ils puissent maintenant chiffrer et déchiffrer une chaîne de caractère.

2.2.3 Boucle While

La boucle While est une méthode d'iteration sur un nombre infini d'opération classique des langages de programmation. Contrairement à d'autres langages, il n'y a pas d'équivalence en Python à **do...while**. La syntaxe d'une boucle While est la suivante:

Listing 3: Exemple d'une boucle While

```
i = 0
while input("Entree N pour quitter\n") != 'N':
    print(f"La valeur de i est: {i}")
    i += 1
```

3. Créez un script python en utilisant une boucle While qui retourne si un nombre est premier ou non. Un nombre premier si ses diviseurs sont 1 et lui-même. **Help:** Une optimisation élémentaire consiste à vérifier uniquement les diviseurs nombres impairs à partir de 3 inférieurs ou égaux à sa racine carrée. La fonction racine carrée est un élément du package *numpy*.

2.3 Syntaxe des structure de base

Un interpréteur Python retranscrit ligne par ligne de code en La syntaxe et la structure des éléments du langage interprété est strict. En effet, l'interpréteur retranscrit en code machine ligne par ligne.

2.4 Fonction

L'organisation du script Python en fonction permet comme les autres langages de Programmation d'organiser son code, ne pas réécrire inutilement les mêmes lignes de codes.
Le mot-clé **def** suivi du nom de la fonction et d'éventuel(s) paramètre(s) définit que le bloc d'instruction suivant est une fonction. L'avantage du Python comparer à certains langages est la possibilité d'avoir un ou plusieurs de paramètre de sortie ainsi que le type en sortie diffèrent en fonction des paramètres d'entrée. Dans l'exemple ci-dessous d'une fonction la variable de sortie peut-être de type *None*, *float* ou *tuple*. Un **tuple** est une séquence non-modifiable de données ordonnées.

Listing 4: Exemple de fonction

```
def resolveEquation(x2=0, x1=0, x0=0):
    rSolve = None
    discriminant = np.power(x1,2) - 4 * x2 * x0
    if discriminant==0:
        rSolve = -x1 / (2.0*a)
    elif discriminant > 0:
        rSolve = ((-x1 - np.sqrt(discriminant))/ (2.0 * a),
                  (-x1 + np.sqrt(discriminant))/ (2.0 * a))
    return rSolve
```

4. Réécrirez le script de la question 3 afin de l'intégrer au sein d'une fonction.
5. Écrirez une fonction de calcul du PGCD entre deux nombres **Help:** Pour calculer le PGCD entre deux nombres, vous pouvez utiliser l'algorithme d'Euclide.

Algorithme d'Euclide: Entre A et B avec A >= B

- Division Euclidienne de A par B.
- A prend la valeur de B, B la valeur du reste de la division
- Réitérer tant que le reste est différents de 0

2.5 Classe

Le langage Python est un langage Orienté Objet (POO) où l'ensemble des variables, même basique comme un entier ou un flottant, correspondent à un objet (Dans la section 2.1, vous avez découvert la fonction type qui retourne le nom de classe d'un objet). Une classe en Python est composée de méthode classique et d'attributs comme en C++, mais également de méthode spéciale décrite sous la syntaxe **__name__**. Ces méthodes sont réservées par Python pour décrire le fonctionnement de l'objet lors de l'initialisation, d'une comparaison entre objets... Les principales méthodes spéciales sont :

- **__init__**: Initialise une instance
- **__repr__**: Retourne une chaîne de caractère lors de l'affichage de l'objet via la fonction Print.
- **__del__**: Similaire au destructeur en C++. La méthode est exécuté lorsque l'objet est supprimé par **del *var***.

Listing 5: caption

```
class Complex:
    # Initialisation de la classse
    def __init__(self, real=0, im=0):
        self.r = real
        self.i = im

    # Representation de la classe
    def __repr__(self):
        return "Nombre complexe: " + str(self.r) + " + " + str(self.i) + "i."

    # Addition entre nombre complexe
    def __add__(self, other):
        return Complex(self.r + other.r, self.i + other.i)

    # Retourne le module du nombre complexe
    def module(self):
        return np.sqrt(np.power(self.i,2) + np.power(self.r, 2))
```

La classe Complex, ci-dessus, permet d'initialiser un objet correspondant à un nombre complexe. Cette classe implémente une fonction calculant le module du nombre complexe ainsi qu'une méthode spéciale décrivant le comportement de l'objet lors d'une addition entre 2 objets Complex.

Listing 6: Instanciation d'objet

```
varA = Complex(2,2)
varB = Complex(real=4, im=3)
print(varA + varB)
print(f"Le module du nombre complexe est {varA.module()}")
```

6. Testez les lignes d'instruction permettant d'initialiser et de manipuler 2 objets Complexe et compléter la classe Complex avec l'implémentation d'une méthode de calcul de l'argument et d'une multiplication entre deux nombres complexes avec la méthode spéciale **__mul__**.

2.6 Exercice de synthèse

L'exercice de synthèse consiste à mettre au point un script de chiffrement de texte basé sur l'algorithme RSA. L'algorithme RSA, correspondant aux initiales des 3 créateurs, est une méthode créée en 1977 de chiffrement répandu au niveau mondial pour chiffrer les échanges de données sur Internet. Contrairement au chiffre de César où la clé pour chiffrer et déchiffrer sont identiques (méthode symétrique) , l'algorithme RSA utilise une clé publique pour chiffrer et une clé privée différente (méthode asymétrique). Il n'est pas nécessaire de connaître la clé publique pour déchiffrer un texte et inversement.

Le script est composé une classe CryptoRSA avec les attributs clé privés et clé publique qui sont deux tuples avec respectivement les valeurs **(n, e)** et **(n, d)**. Cette classe intègre également une méthode de génération des clés nommée generateKey, une méthode de chiffrement et une méthode de déchiffrement. Le morceau de code ci-dessous reprend les consignes précédentes. Compléter ce code avec les instructions de fonctionnement des algorithmes de génération de chiffrement en utilisant les fonctions crées à la question 5 et 6.

Listing 7: Classe CryptoRSA à completer

```
class CryptoRSA():
    def __init__(self):
        self.privateKey = None
        self.publicKey = None
        ...

    def generateKey(self):
        ...
        self.privateKey = (e,d)
        self.publicKey = (n,d)

    def chiffrement(self, text):
        ...
        return crypText

    def dechiffrement(self, crypText):
        text = ""
        numList = "";
        crypText = crypText.split(" ")
        for idx in crypText:
            value = (int(idx)** self.privateKey[1]) % self.privateKey[0]
            numList += str(value)
        for idx in range(int(len(numList) / 2)):
            asciiVal = ""
            for j in range(2):
                asciiVal += numList[idx*2 + j]
            text += chr(int(asciiVal))
        return text
```

2.6.1 Description de l'algorithme

Algorithme de Génération des clés

- Entrer un nombre premier P et Q
- Calculer n = P * Q et ϕ(n) = (P - 1) * (Q - 1)
- Entrer e tel que PGCD(e,ϕ(n)) = 1

- Obtenir d tel que $de \equiv 1$ (modulo ϕ(n)). Cette formule équivalent au snippet de programme suivant

Listing 8: Equivalent en langage Python de la formule $de \equiv 1$ (modulo ϕ(n))

```
tmp = 1
d = 1
while (((d * e) % phi) != 1):
    tmp = tmp + 1
    d = round(tmp * f / e)
d = int(d)
```

Algorithme de chiffrement

- Transcrire la chaîne de caractère en une suite de numériques (utilisation du code ASCII).
- Séparer la chaîne numérique en une suite de nombre de n-1 chiffres.
- Coder chaque bloc numérique avec la formule :

$$C = pow(Nombre, e) \, mod \, n \tag{3}$$

- Concaténer la suite de bloc numérique codé séparé par un espace.

Algorithme de déchiffrement

Méthode inverse à l'algorithme de chiffrement avec la formule de déchiffrement suivante :

$$M = pow(C, d) \, mod \, n \tag{4}$$