## University of Science and Technology HOUARI BOUMEDIENE Faculty of Electronics and Computer Science

**Département d'informatique Examen Python –L2 S4**

USTHB/F.E.I/Département Informatique/L2 ACAD : A, B et C/Examen : PYTHON - 2019/2020

Fait par l’etudiant :

NOM : KHITER.

PRENOM : Mohamed Achraf.

Matricule : 171731032520.

Section : B.

Groupe : 03.

**Exercice 01 :**

1. Expliquation de chaque ligne de code du script suivant, permettant de lire tout le contenu d’un répertoire :

le script suivant permet de lister tout le contenu d'un répertoire recursivement jusqu'à trouver le dernier sous fichier (condition d'arrêt)

* import os,sys : ça veut dire importer les deux modules (bibliotheques) standards du python , pour qu'on puisse profiter de l'utilisation de ses fonctions et sous modules . Les modules os et sys fournissent de nombreux outils pour gérer les noms de fichiers, les chemins et les répertoires. Ces modules sont des wrappers (emballages)pour des modules spécifiques à la plate-forme, donc des fonctions comme :

os.path.isdir(i) , sys.argv[i] , sous UNIX, Windows, Mac OS et toute autre plate-forme prise en charge par Python.

* def R(d): c'est la définition d'une fonction qui s'appelle (R) avec une variable (d) comme paramètre d'entrée. (d) est du type (string) fait la référence au nom du répertoire / fichier.
* for i in os.listdir(d): le debut de la boucle (for) en utilisant une variable (i) aussi du type (string) qui fait référence au nom du répertoires / fichiers situe sous le repertoire courant (d) un par un dans chaque iteration.

l'utilisation de la fonction standard os.listdir(d) permet de lister le contenu du repertoire (d) de A a Z : fichiers, sous-repertoires

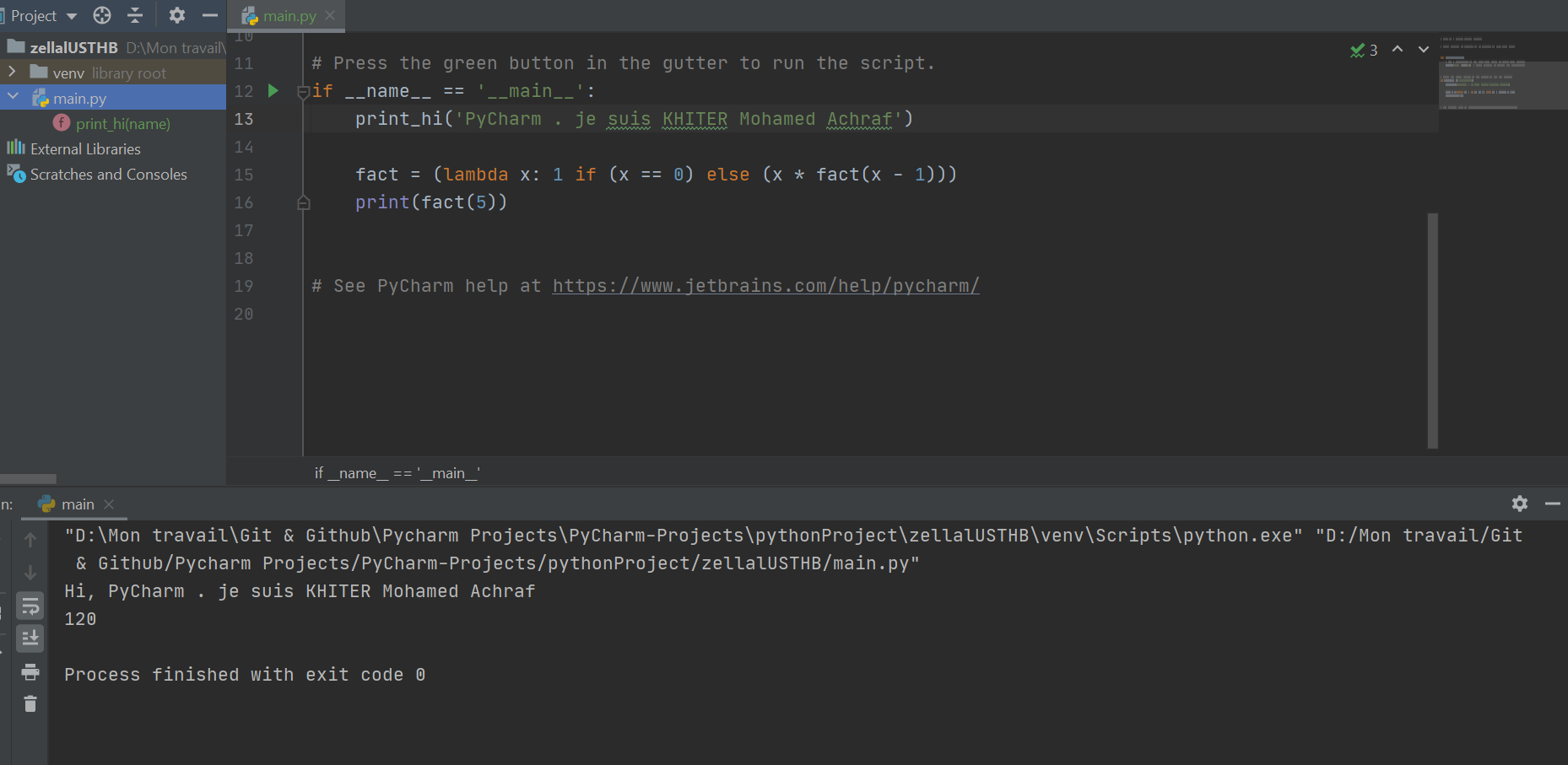
* i=d+"/"+i : la variable (i) recoit la concaténation du 3 chaines de caractere (d,i et "/") .tel que (d) : repertoire courant , (i):un objet quelquonque situe sous le rep (d) . et le "/" pour generer le nouveau chemain relatif a tester .
* if os.path.isdir(i): commencer une condition(combinaison de if et de else) pour tester a l'aide de la fonction standard isdir(i) du module OS si l'objet (i) est un sous-repertoire ou pas .

si oui on fait:

* R(i),print(i) : cette ligne contient deux instructions différentes : la premiere celle de R(i) : c est un appel recursif a la fonction predifinie (R) avec l argument (i) . qui est dans ce cas un sous repertoire. pour la deuxième ligne print(i) . c est facile c est la fonction standard print qui nous aide a afficher un message dans la console .dans ce cas est le dernier sous-repertoire empiler dans la pile durant les appels recursifs,precisementle sous-repertoire qui ne contient plus des sous-repertoires, apres ca on commence a depiler .
* Else : sinon on fait ca :
* print(i) : afficher la variable (i) qui est une chaine de caractere du nom du fichiers testes sous un chemain relatif au repertoire principale .
* R(sys.argv[1]) : c est un appel a la fonction (R) avec un parametre d entree qui est un argument (I/O) numero 1 . il presente le repertoir de depart .

**Exercice 03 :**

1. reprendre le script ci-dessous, en supprimant les deux boucles « for » dans la partie en gras (partie du calcul de factorielle). Le calcul de factorielle doit passer de 7 à 2 lignes de code suivant :

* factorielle = (lambda x: 1 if (x == 0) else (x \* fact(x - 1)))
*  [print(factorielle(i)) for i in [x-1 , y-1 ]] # x et y : sont des nombres donnes .

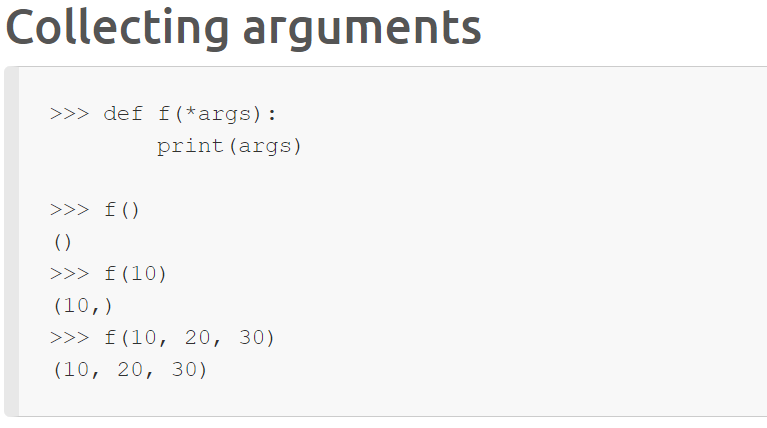
1. Expliquer à quoi sert l’étoile \* placée avant le « x » dans la ligne de code 10 :

Mettre (\* args) comme dernier élément dans la liste d'arguments de notre définition de fonction permet à cette fonction d'accepter un nombre arbitraire d'arguments anonymes et / ou de mots-clés.

Ces arguments sont appelés arguments de mots-clés. En fait, ce sont des espaces réservés pour plusieurs arguments, et ils sont particulièrement utiles lorsque nous devons passer un nombre différent d'arguments à chaque fois que nous appelons la fonction.

{

et voilà l'ensemble des arguments entres se forment un tuple (tableu des valeurs ) comme parametre d'entree . la meme chose avec la sortie . ca sera un tuple de resultats a parcourir.

}

**Exercice 02 :**

1. expliquer l’expression régulière, qui se trouve entre guillemets en gras (ligne de code 5)

* x=re.search("[.!]??(.$)",i,re.S)
* [.!]??(.$)

[.!] :

Faites correspondre un seul caractère présent dans la liste ci-dessous.

?? :

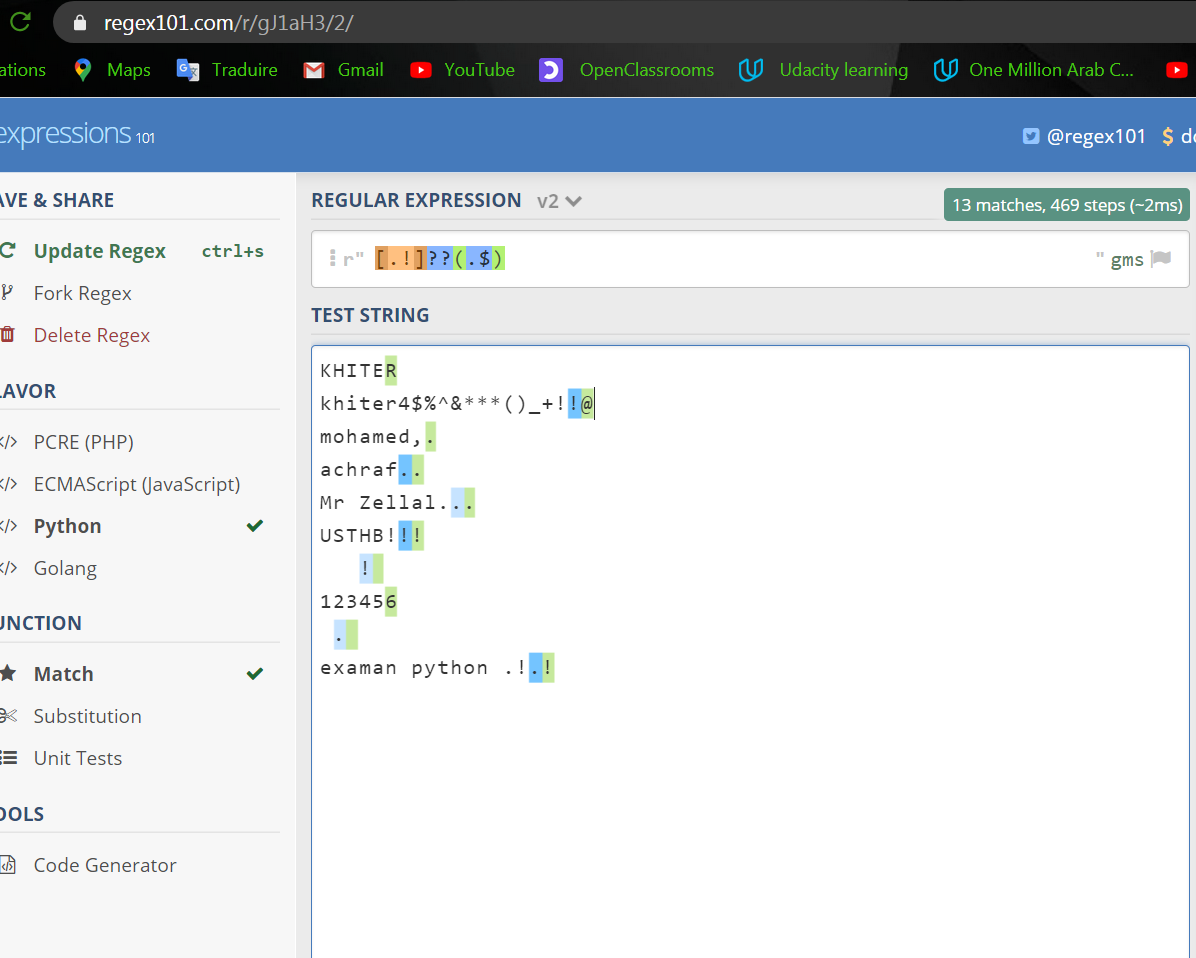
Quantificateur - Correspond entre zéro(0) et une fois(1), aussi peu de fois que possible, en s'étendant au besoin (paresseux) ,

1st Capturing Group  (.$) :

. correspond à n'importe quel caractère

$ affirme sa position à la fin d'une ligne

**REGEX FLAGS : s (re.S):** **faire (.) correspond également à la nouvelle ligne.**

* ****en générale cette expression reguliere capture toujours le dernier caractère de chauque lignes(espace blanc " " inclu), et aussi capture l'avant dernier caractère si il est équivalent a (.) ou (!) comme on peut voir dans l'exemple dessous : # quand il capte un caractere il fait pas de saut de ligne dans l’affichage . si il capte rien (ligne vide) il fait un saut de ligne(\n)

**Exercice 04 :**

1. Expliquer la ligne de code 3 en gras du script ci-dessous.

* for j in zip(\*[re.split("\W+",open(sys.argv[1],'r',encoding="utf-8").read().lower())[i:] for i in range(int(sys.argv[2]))]):
* d[j]=d.get(j,0)+1

Cette ligne dans ce scripte elle va prendre a l’entree un fichier ex : (test.txt)

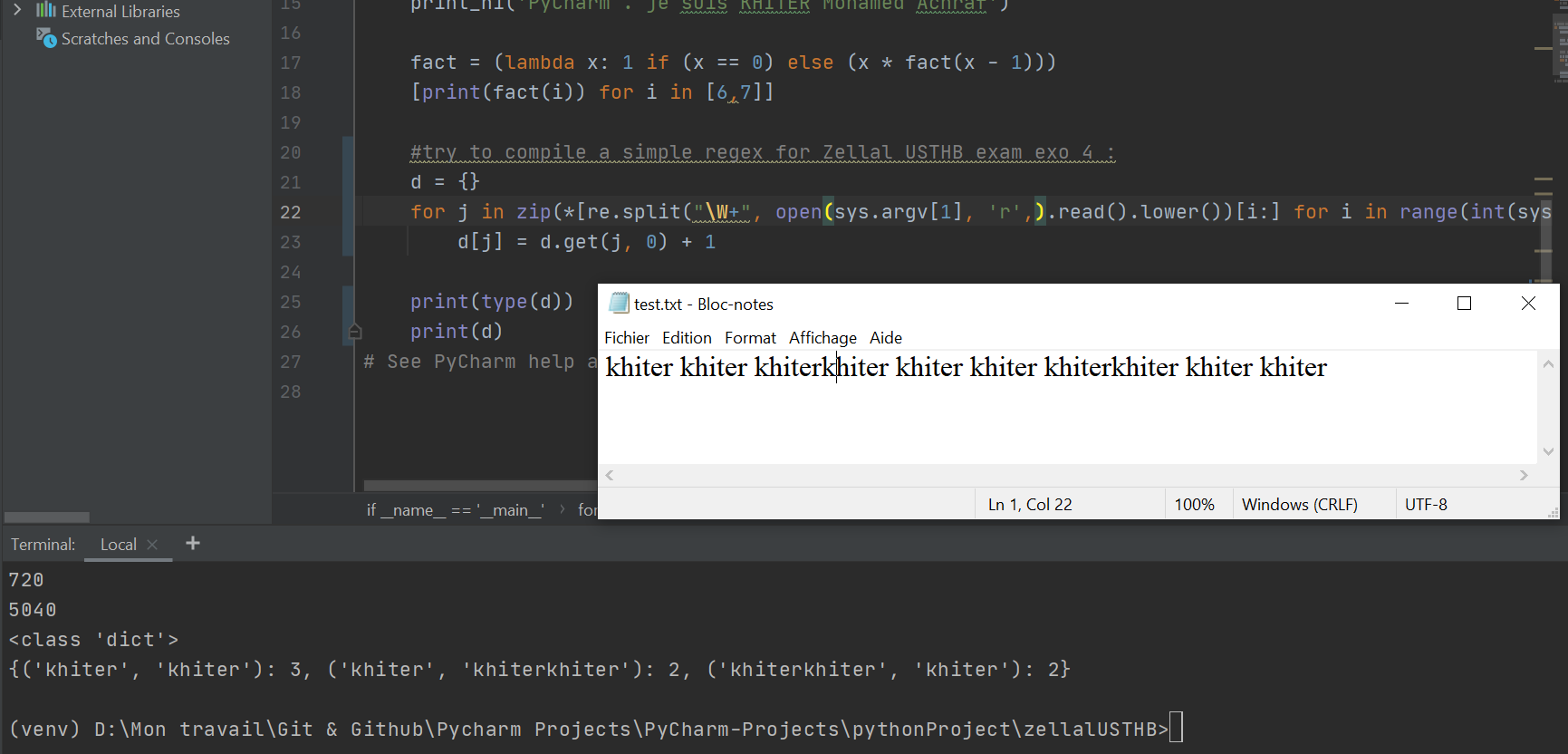
Comme (sys.argv[1]) et un nombre entier comme (sys.argv[2]) qui represente

longueur de co-occurrence pr ex (2 : [‘khiter’,’achraf’]) avec un espace blan entre eux , alors cette portion de code genere un dictionnaire (cle , valeure)

la cle represente un tuple des mots separes par un espaces blanc dans le text principale . le nombre du mots dans le tuple depends a l’entier d’entre (sys.argv[2]) pr exemple si (sys.argv[2]) = 3 alors le tuple soit comme ca :

(‘khiter’,’mohamed’,’achraf’) il prend les trois premier mots separee par un espace et calculer la fréquence de cette cooccurrence dans tous le fichier

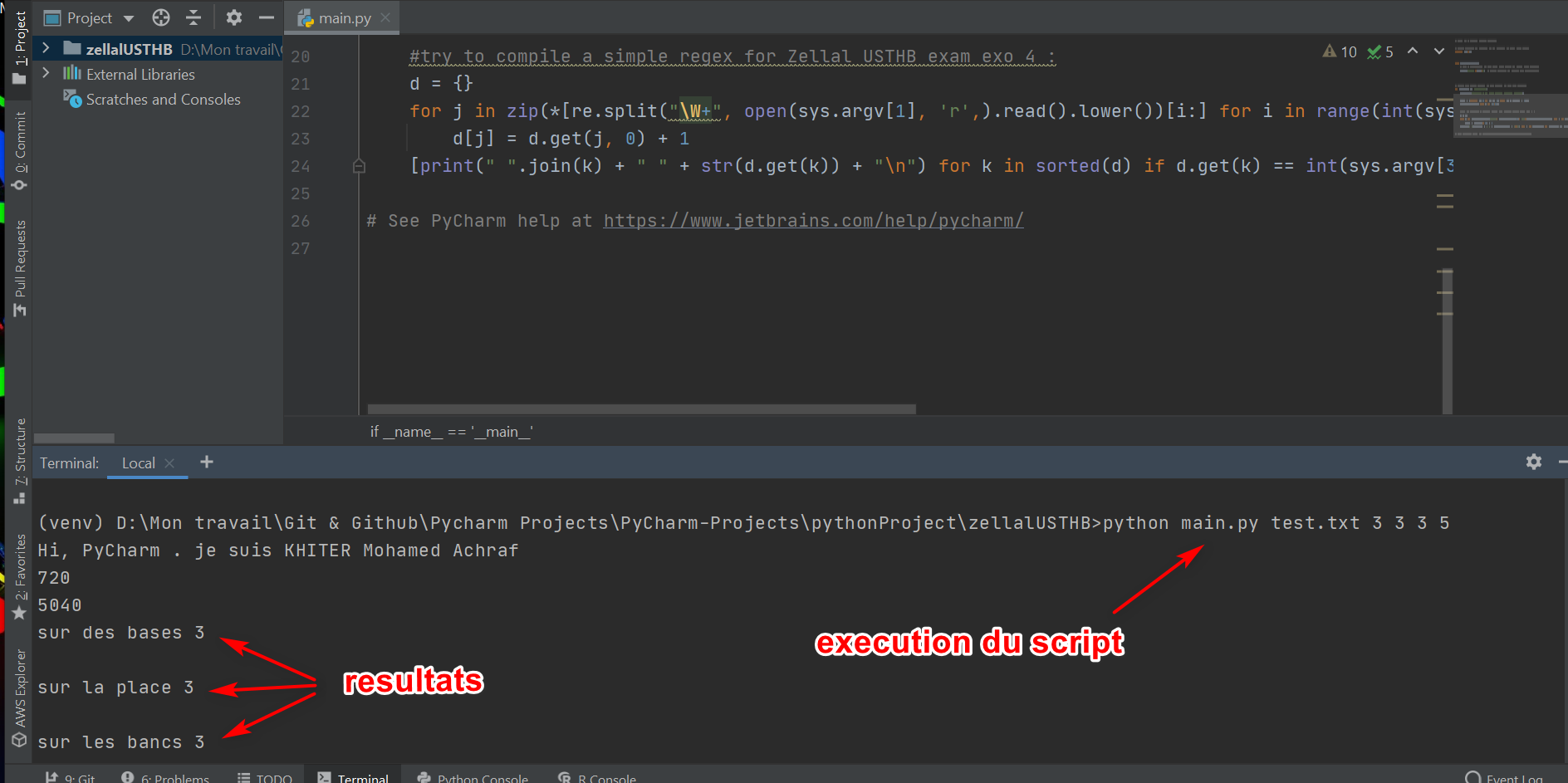
Et retourner cette fréquence comme la valeure de la cle dans le dictionnaire .

Apres revenir au debut er prendre la deuxieme cooccurrence trouvee en avancant le cursur avec un mot . c pour ca on va trouver plusieures cooccurrence . voici un exemple aui explique ca :

1. expliquer l’expression régulière, qui se trouve entre guillemets en gras (ligne de code 5).

* "^\w{"+sys.argv[4]+"} (.+ )?\w{"+sys.argv[5]+"}$" :  
  Remarque : sys.argv[4] = longueur du 1er token d’une cooccurrence et sys.argv[5] = longueur du dernier token d’une cooccurrence.

L’expression régulière va extraire : une chaîne de caractères qui commence par une sous chaîne (un mot)de longueur argv[4] et se termine par une autre sous chaîne (mot)de longueur argv[5] et entre ses 2 chaînes on peut trouver toures autres caractères tout sauf l espace blanc ou le saut de ligne

1. Modifier ce script, en supprimant la ligne de code 2 (création du dictionnaire vide « d »). Après modification, ce script doit passer de 5 à 4 lignes de code, tout en conservant les 3 boucles « for » et la condition « if ». :

Le nouveau code est :

* import sys,re
* for j in zip(\*[re.split("\W+",open(sys.argv[1],'r',encoding="utf-8").read().lower())[i:] for i in range(int(sys.argv[2]))]):
* d[j]=d.get(j,0)+1
* [print(" ".join(k)+" "+str(d.get(k))+"\n") for k in sorted(d) if d.get(k)==int(sys.argv[3]) and re.search("^\w{"+sys.argv[4]+"} (.+ )?\w{"+sys.argv[5]+"}$"," ".join(k))] .

FIN .

MERCI POUR LA

Correction.