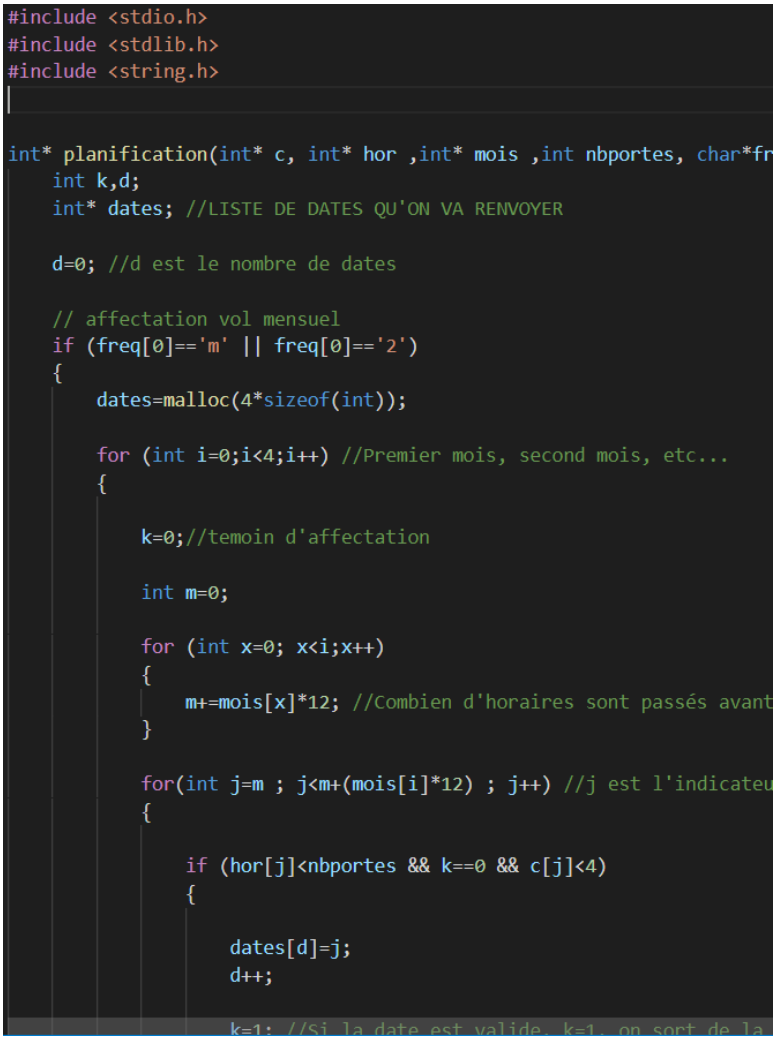


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int* planification(int* c, int* hor ,int* mois ,int nbportes, char*fr
    int k,d;
    int* dates; //LISTE DE DATES QU'ON VA RENVOYER

    d=0; //d est le nombre de dates

    // affectation vol mensuel
    if (freq[0]=='m' || freq[0]=='2')
    {
        dates=malloc(4*sizeof(int));

        for (int i=0;i<4;i++) //Premier mois, second mois, etc...
        {

            k=0;//temoin d'affectation

            int m=0;

            for (int x=0; x<i;x++)
            {
                m+=mois[x]*12; //Combien d'horaires sont passés avant
            }

            for(int j=m ; j<m+(mois[i]*12) ; j++) //j est l'indicateu
            {

                if (hor[j]<nbportes && k==0 && c[j]<4)
                {

                    dates[d]=j;
                    d++;

                    k=1; //Si la date est valide, k=1, on sort de la
```

Optimisation de la planification des vols

PYTHON & C

Ouyed Ryan, Bosle Clément | NF06 | 30/12/2021

Table des matières

Introduction

Le travail demandé était de créer

1. Une mise en place du planning de vols sur une période de 4 mois pour chacune des compagnies aériennes rentrées par l'utilisateur
2. Une fois le planning des vols mis à jour, il faut affecter chacune des requêtes de vol de chaque compagnie aérienne aux différentes portes de l'aéroport pour chaque jour (sur 4 mois).

La difficulté dans ce projet NF06 est de devoir mélanger du Python et du C. Chacun ayant son propre rôle.

Nous verrons dans un premier temps la partie sur Python avec la récupération des données de l'utilisateur ainsi que son affichage, puis dans un deuxième temps les 2 fonctions en C qui permettent la planification et l'affectation des vols.

Programmation sur Python

1. def creerhoraire

Nous travaillons sur 4 mois, il faut donc trouver un moyen efficace de pouvoir fractionner ces 4 mois en éléments sur lesquels il sera simple de travailler.

C'est l'utilité de la première fonction def creerhoraire() qui permet de créer un format horaire sur les 4 mois.

L'idée est de convertir ces 4 mois en « créneaux ». On part du principe qu'un vol occupe une porte pendant 2 heures. Donc sur une journée, on aurait 12 créneaux (24/2) et donc sur 4 mois (2 mois de 31 jours et 2 mois de 30 jours), cela fait un total de 1464 créneaux.

Ceci est le mode d'utilisation par défaut, mais la fonction permet également à l'utilisateur d'entrer manuellement la durée en jours de chacun des 4 mois.

2. def creerportes

L'utilisateur va renseigner :

- le nombre de porte de l'aéroport
- la taille de chaque porte (facultatif)

Une porte peut avoir une taille spécifique et donc compatible uniquement sur certains vols (en fonction du nombre de passagers de ce vol).

Sinon l'utilisateur pourra utiliser le mode par défaut sans avoir des portes de tailles standard, qui peuvent accueillir n'importe quel vol.

3. def creercompagnies

Les compagnies sont définies par :

- un nom (ex : AirFrance)
- un nombre minimum de vol sur les 4 mois
- une flotte d'avion définie par :
    - des modèles d'avion
    - un nombre d'avion pour chaque modèle
    - une capacité minimale et maximale pour chaque modèle d'avion

Exemple de répertoire de compagnie :

nom	min	flotte
airfrance	20	flotte1
qatar	65	flotte2
emirates	45	flotte3

Exemple de flotte pour une compagnie (ici AirFrance) :

modele	nombre	capaMin	capaMax
A380	3	400	500
A320	10	100	200
B777	5	200	300

L'utilisateur pourra renseigner manuellement toutes les compagnies et tous leurs détails ou alors fournir un fichier Excel (comme ceux en exemple) avec déjà toutes les informations à l'intérieur.

La fonction stocke alors ces compagnies dans une liste de dictionnaire python.

Ex : [{'nom': 'emirates', 'min': 45, 'flotte': [{'modele': 'A320', 'nombre': 3, 'capaMin': 100, 'capaMax': 200}]}]

4. def recupcompagnie

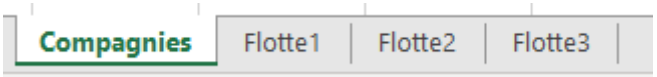
Ici la fonction va récupérer les informations des compagnies depuis un fichier excel à multiple feuilles (voir testcompagnie.xlsx). La fonction va extraire 2 types de dictionnaire (grâce à pandas) :

- un dictionnaire compagnie avec : nom, nombre min de vol et leur flotte.

Cependant chaque compagnie aérienne à sa propre flotte composée d'un nombre unique d'avions de modèle unique. Donc il est nécessaire de passer par la création d'un autre dictionnaire spécialement pour chaque flotte de chaque compagnies :

- dictionnaire flotte[i] avec i allant de 0 au nombre de compagnies, composé de : modèle de l'avion, nombre d'avion de ce modèle, capacité minimale et maximale de ce modèle d'avion.

Si nous avons 3 compagnies, le fichier Excel ressemble à ça :



5. def creerrequetes

Cette fonction permet à l'utilisateur de créer les requêtes de vol. Chaque compagnie peut avoir plusieurs requêtés de vol définie par :

- le nom de la compagnie qui fait la requête
- le numéro de vol
- les appareils pouvant effecteur le vol
- la ville de départ et d'arrivée
- la fréquence de vol de cette requête

Ex d'une fiche Excel de requêtes de vol :

compagnie	nom	appareils	depart	arrivee	freq
airfrance	AK7848	A380	Paris	Jakarta	hebdomadaire
airfrance	AJ2345	A320	Troyes	Paris	journalier
qatar	QT1018	A35K	Doha	Dubai	journalier
qatar	QT1188	B788, B777	Paris	Dubai	hebdomadaire
qatar	QT1768	B789	Barcelone	Londres	mensuel
emirates	UA1245	A388	Istanbul	Manchester	hebdomadaire
emirates	UA1924	B77W, A388	Lisbonne	Bruxelles	journalier

Chaque requête correspond à un dictionnaire qui seront ajoutés à la liste « requetes » et qui sera return vers le main.

```
requetes.append({"compagnie":compagnies[i]["nom"],"num":num,"appareil":
appareil,"depart":dep,"arrivee":arr,"freq":freq})
```

6. def recuprequetes

Cette fonction à la même fonction que celle d'avant, elle va juste aller chercher les informations dans un Excel (voir testrequetes.xlsx) et les stocker dans un dictionnaire puis ensuite les ajouter à la liste et return (récupéré comme requetes).

Ex d'une requete :

{'compagnie': 'emirates', 'nom': 'UA1245', 'appareils': 'A388', 'depart': 'Istanbul', 'arrivee': 'Manchester', 'freq': 'hebdomadaire'}

7. def trackcompagnie

Cette fonction renvoie un tableau avec le nombre de vol pour chaque jour de la compagnie prise en entrée.

8. def data_to_excel

Fonction très simple qui permet juste de créer un Excel à partir de données (dictionnaire et liste). Cette fonction n'a pas d'utilisation primaire dans notre programme mais a servi, au cours de la programmation, à tester si les dictionnaires sont viables.

Programmation en C

1. Fonction planification

La fonction planification prend en entrée le tableau horaire de l'aéroport, le nombre de portes de l'aéroport, la fréquence du vol demandé, un tableau renseignant la longueur des mois et un tableau renseignant le nombre de vols déjà attribués à la compagnie chaque jour.

Cette fonction renvoie une liste d'entiers, où chaque entier est un créneau qui a été attribué au vol demandé.

Pour attribuer un vol, l'algorithme examine d'abord la fréquence du vol (mensuelle, hebdomadaire ou journalière) puis vérifie pour chaque période à assigner (chaque mois, chaque semaine ou chaque jour) si le tableau horaire de l'aéroport, qui renseigne l'occupation des portes, indique une occupation inférieure au nombre de portes à un horaire donné. Si c'est le cas, un vol est assigné (on ajoute l'horaire sélectionné au tableau de sortie), on incrémente l'horaire concerné dans le tableau horaire et on passe à la période suivante. Si ce n'est pas le cas, on ajoute l'horaire "-1" qui indique que le vol sur cette période n'a pas pu être assigné.

2. Fonction affectation

La fonction affectation prend en entrée le nombre de portes, le tableau renseignant le statut des portes (occupée ou libre) à l'horaire du vol, le tableau contenant la taille des portes, et le nombre de passagers que l'avion peut accueillir.

L'algorithme renvoie le numéro de la porte auquel l'avion a été assigné.

Pour affecter le vol à une porte, l'algorithme examine toutes les portes. Si l'une d'elle a le statut "libre" et une taille supérieure ou égale à celle de l'avion, le vol est assigné et on renvoie le numéro de cette porte. Si aucune porte ne correspond, on renvoie l'entier correspondant à une erreur, "-1".

Problèmes rencontrés

1. Problème de dictionnaire dans le def recupcompagnie

Etant donné que nous avons 2 dictionnaires :

- Un dictionnaire général pour les compagnies
- Un dictionnaire pour chaque flotte de chaque compagnie

Il faut donc les assembler afin que nous puissions avoir le dictionnaire compagnies (avec les flottes associées à chaque compagnie) dans un grand dictionnaire.

Cependant si on fait juste un « append », les 2 dictionnaires seront juste placés l'un a la suite de l'autre.

On a donc utilisé la fonction .update de pandas qui va regarder si 2 clés de 2 dictionnaires différents sont similaires et si oui, il va combiner les valeurs (merge values) des 2 dictionnaires ensemble dans le premier dictionnaire au niveau de leur clé compatible.

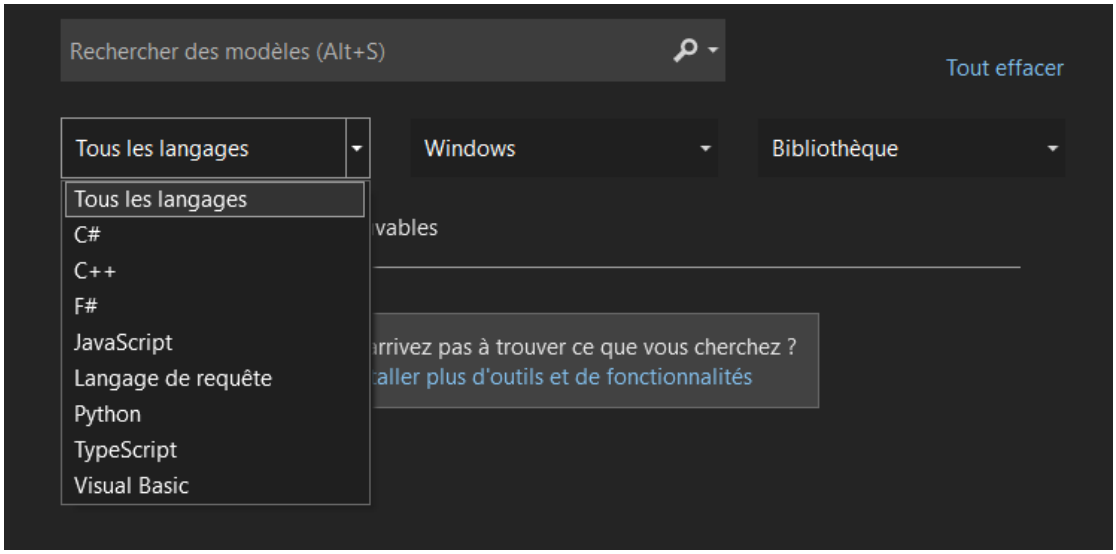
```
        cd[i-1].update(dict0)
```

Cela nous permet d'avoir ceci :

{'nom': 'qatar', 'min': 65, 'flotte': [{'modele': 'A350', 'nombre': 5, 'capaMin': 200, 'capaMax': 300}, {'modele': 'A320', 'nombre': 6, 'capaMin': 100, 'capaMax': 200}]}

2. Problème lors de la création du shared library

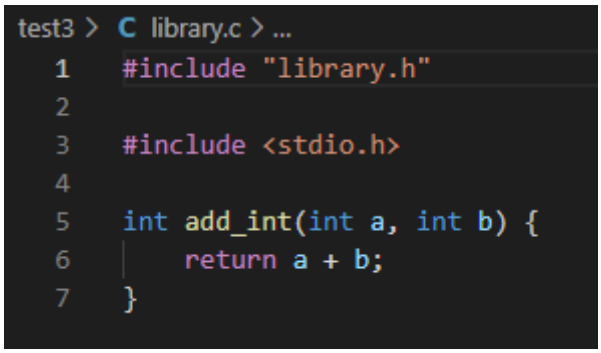
On a téléchargé Visual Studio 2022 (version community), mais cependant il fut très compliqué de savoir comment créer une shared librairy car le C n'était même pas disponible comme language.



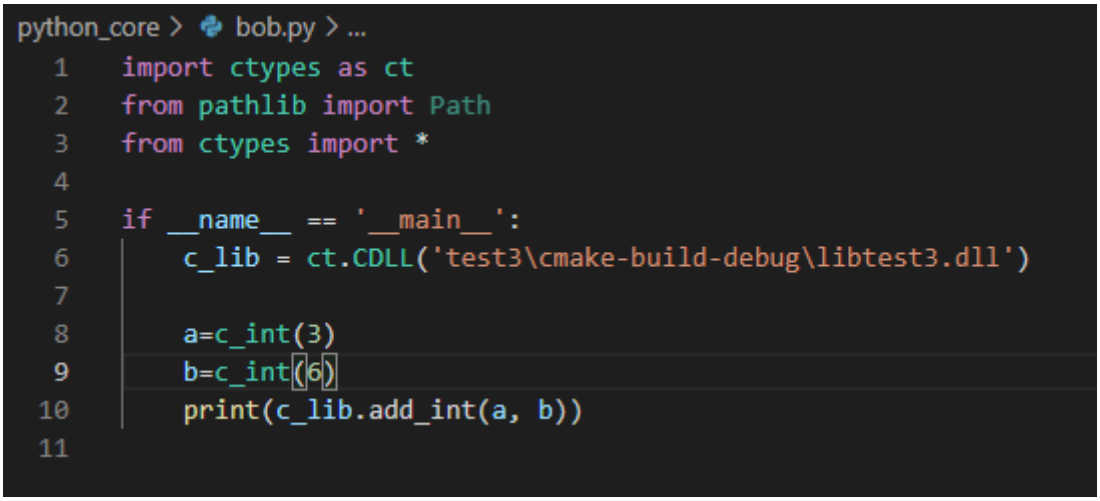
On est alors aller voir du côté de Codeblocks, mais celui-ci n'imprime pas le .dll même après une bonne compilation.

On s'est alors tourné vers Clion et étonnament celui-ci a marché. Nous l'avons donc utilisé pour compiler le code C en un shared library et utilisé vscode pour utiliser le shared librairy avec Python.

Test simple du ctypes :



En C



En Python

Output : 9

3. Problème de conversion des arguments de Python vers la fonction C

La ligne qui posait problème fut celle-ci :

```
    v["porte"]=c_lib.affectation(ct.c_int(nbportes),  ct.c_int*len(disp
oportes[v["date"]])(dispoportes[v["date"]]),  ct.c_int*len(tailleportes
)(tailleportes), ct.c_int(capaMax))
```

Malgré les conversions faites avec ct.c_int ou meme juste c_int avec la variable dans la parenthèse, Python retournait toujours la même erreur :

“an integer is required (got type list)”

Le programme en C est :

```
123   int affectation(int nbportes, int* statutportes, int* tailleportes, int capaMax)
124   {
125       int i;
126       for (i=0;i<nbportes;i++) //On vérifie qu'il n'y ait pas une porte déjà libre
127       {
128           if (statutportes[i]==0 && tailleportes[i]>=capaMax)
129           {
130               return i+1;
131           }
132       }
133       return -1;
134   }
135
136
```

Nous avions pensé que la conversion de la liste python s’est alors mal faite et donc il faudrait utiliser ctypes.POINTER() :

Ctypes.POINTER() permet de récupérer le pointeur de l’item C à l’intérieur. Cependant cet item doit être un item en C. Donc nous devons convertir nos liste python en tableau C.

Avec un remplissage test de valeurs :

```
py_a=[1,2]
py_b=[3,4]

seq_a = ct.c_int * len(py_a)
a=seq_a(*py_a)

seq_b = ct.c_int * len(py_b)
b=seq_b(*py_b)

c=c_int(3)
d=c_int(500)
portes=c_lib.affectation(c_int(3), ct.POINTER(a), ct.POINTER(b), c_int(500))
```

Output : TypeError: unhashable type

A ce point-là, nous sommes un peu bloqués.

Mode d'emploi du programme

Lancer le main.py du Python (dans le folder sharedlib/python_core) :

Voulez-vous utiliser le mode par défaut ? Répondez 0 pour Non, 1 pour Oui

Si vous faites 1, le mode par défaut sera pris : 2 mois de 30 jours et 2 mois de 31 jours

Sinon en faisant 0, vous pouvez entrer vos propres durées de jours pour chacun des 4 mois.

Le programme affiche le nombre de créneaux disponibles.

Combien de portes dispos dans l'aéroport ?

Entrer le nombre de portes de l'aéroport

les portes ont-elles une taille spécifique? Si oui, entrer 1, 0 sinon

Si 1, la taille standard sera utilisée où il n'y a qu'une seule taille fixe.

Sinon en mettant 1, vous pouvez donner une taille spécifique pour chacune des portes allant de petite, moyenne et grande.

Comment voulez-vous entrer les compagnies ? Pour les entrer manuellement, taper 0 Pour les extraire depuis un fichier, taper 1

Si 0, vous allez les rentrer une par une avec tous les détails.

Si 1, le programme va chercher les infos dans le Excel testcompagnie.xlsx à l'intérieur du folder parent.

Comment voulez-vous entrer les requêtes? Pour les entrer manuellement, taper 0 Pour les extraire depuis un fichier, taper 1

Si 0, même chose qu'avant, il faut manuellement taper chaque requêtes et ses infos.

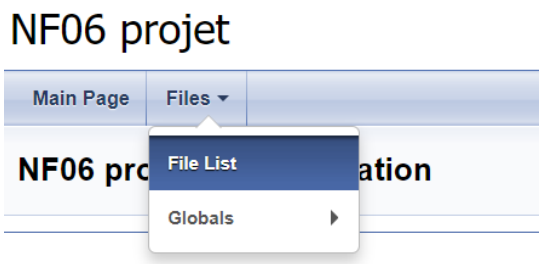
Si 1, le programme va chercher les requêtes dans le Excel testrequetes2.xlsx à l'intérieur du folder parent.

Le programme par la suite renvoie un Excel avec toute la liste des vols, leur date et les portes auxquelles ils sont assignés.

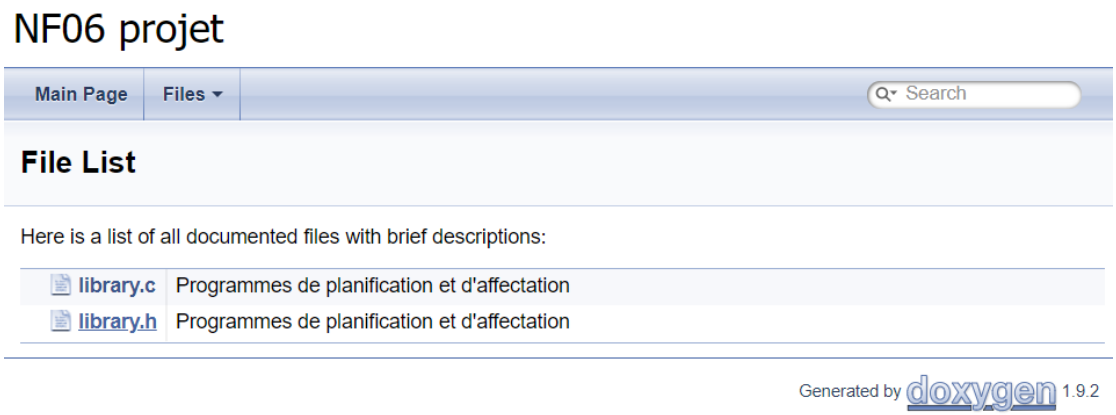
Mode d'emploi DOXYGEN

Aller dans le folder Doxygen, cliquer sur le fichier index.html.

Appuyer sur files -> File List



Choisir la librairie à afficher en cliquant dessus :



Conclusion et perspectives

Ce projet est assez intense dans le sens où il faut vraiment réfléchir avant de s'y lancer. Nous avions fait une session de brainstorming ou nous avions trié les informations, écrit ce que nous devions faire et structuré tout cela afin de bien comprendre par où nous devions commencer et dans quelle direction nous allions.

Il a fallu réfléchir sur l'interprétation du sujet et l'implémentation. Je peux prendre par exemple le système de créneaux. Le sujet nous précise uniquement que nous devons planifier, c'est alors à nous de décider comment incorporer ceci dans une optimisation de vols, que ce soit par le nombre d'heures passées à l'aéroport ou le format horaire (heure-jour-mois, heure-jour-semaine-mois ou des créneaux sans structure comme nous l'avons fait).

Nos connaissances de python, du c et de pandas pendant ce projet. Pour trouver une solution à un problème, nous avons dû nous débrouiller nous-même et aller chercher les informations sur internet.

Le programme peut être encore grandement amélioré. Dû fait que nous soyons restés bloqué dans la communication entre Python et C, nous n'avons pas pu nous concentrer sur d'autre détails d'optimisation du programme, le débogage des parties ultérieures, et la création de fonctions de replanification et de réaffectation qui auraient elles faits une priorisation et une sélection parmi les vols. En l'état, notre programme ne fait que remplir les créneaux et les portes jusqu'à ce qu'ils soient pleins.

Durée estimée de travail combiné : Python : 40h, C : 10h, débogage du ctypes : 20h.

Références

Ctypes made easy – C/C++ Dublin User Group – Youtube https://www.youtube.com/watch?v=p_LUzwylf-Y&t=298s

how to create and use shared library in codeblocks - https://forums.codeblocks.org/index.php?topic=22538.0

Using Pandas to pd.read_excel() for multiple worksheets of the same workbook - https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multiple-worksheets-of-the-same-workbook

Python | Pandas Dataframe.to_dict() - https://www.geeksforgeeks.org/python-pandas-dataframe-to_dict/

Python | Split dictionary keys and values into separate lists - https://www.geeksforgeeks.org/python-split-dictionary-keys-and-values-into-separate-lists/

How to count the total number of sheets in an Excel file using Python - https://stackoverflow.com/questions/50950359/how-to-count-the-total-number-of-sheets-in-an-excel-file-using-python

itertools not defined when used inside module - https://stackoverflow.com/questions/4125231l/itertools-not-defined-when-used-inside-module

Python enumerate(): Simplify Looping With Counters - https://realpython.com/python-enumerate/

Slicing a dictionary - https://stackoverflow.com/questions/29216889/slicing-a-dictionary

Python Dict and File - https://developers.google.com/edu/python/dict-files

Safe method to get value of nested dictionary - https://stackoverflow.com/questions/25833613/safe-method-to-get-value-of-nested-dictionary

Python Nested Dictionary - https://www.programiz.com/python-programming/nested-dictionary

Python Dictionary fromkeys() Method - https://www.w3schools.com/python/ref_dictionary_fromkeys.asp

Python – Pandas – Converting XLSX to Dictionary (key and values) - https://www.heelpbook.net/2020/python-pandas-converting-xlsx-to-dictionary-key-and-values/

Python TypeError: 'builtin_function_or_method' object is not subscriptable Solution - https://careerkarma.com/blog/python-builtin-function-or-method-is-not-subscriptable/

pandas.DataFrame.to_dict - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_dict.html

TypeError: 'int' object is not callable - https://stackoverflow.com/questions/9767391/typeerror-int-object-is-not-callable

Python ctypes error - TypeError: an integer is required (got type LP_c_long) - https://stackoverflow.com/questions/65654144/python-ctypes-error-typeerror-an-integer-is-required-got-type-lp-c-long

Questions tagged [ctypes] - https://stackoverflow.com/questions/tagged/ctypes

*What's the difference between ctypes.POINTER(ctypes.c_int) and ctypes.c_int * 5?* - https://stackoverflow.com/questions/57027124/whats-the-difference-between-ctypes-pointerctypes-c-int-and-ctypes-c-int-5

ctypes — A foreign function library for Python - https://docs.python.org/3/library/ctypes.html#ctypes.c_char_p

"input expected at most 1 arguments, got 2"- https://stackoverflow.com/questions/43243627/input-expected-at-most-1-arguments-got-2

Pointers and arrays in Python ctypes - https://stackoverflow.com/questions/1363163/pointers-and-arrays-in-python-ctypes