redis_script

November 17, 2020

1 Redis project

Fait par @ChloéTellier, @OcéaneGuitton, @FlavieThévenard et @AntoineLucas

1.1 Livrables :

1.2 - Ce notebook Jupyter avec le projet réalisé et quelques explications.

1.3 - Un fichier *projet.py* avec le code commenté pour exécuter facilement le programme.

2 Un peu de contexte : c'est quoi Redis ?

Redis, qui sert d'acronyme pour **RE**mote **DI**ctionary **S**erver, est système de gestion de base de données structure de stockage de données en mémoire, sous licence BSD, lancée en 2009 par Salvatore Sanfilippo.

L'une des grandes différences entre Redis et les autres bases de données NoSQL réside dans les structures de données que Redis fournit. Les utilisateurs de Redis peuvent exploiter les structures de données comme les chaînes, les hachages, les listes, les ensembles et les ensembles triés en utilisant des commandes similaires aux opérations de requête présente dans la plupart des langages de programmation.

Pour mieux comprendre les différentes structures nous vous invitons à parcourir rapidement le lien ci-contre. Et plus généralement la documentation et le site de Redis qui ont des documents complets et plutôt clairs.

Bonus tutoriel rigolo

3 Installation

3.1 Linux

Il suffit de se rendre à cette adresse et de suivre les consignes indiquées dans la partie installation.

Vous serez ensuite prêt en lançant dans un terminal la commande `shell  src/redis-server`

Voici un exemple de terminal indiquant le fonctionnement d'un serveur Redis : serveur redis *un exemple de serveur redis ouvert*

Vous pouvez ensuite lancer des requêtes en lançant le client redis : `shell src/redis-cli`

Ensuite il suffit de réaliser les requêtes sous python ! On verra plus bas pour établir la connexion avec le serveur redis généré plus haut.

3.2 Installation des librairies utiles dans python :

On doit installer les librairies python suivantes pour exécutez notre code :

`conda/pip install redis conda/pip install pandas conda/pip install numpy conda/pip install scipy conda/pip install seaborn`

Si vous avez des difficultés à installer les modules au sein de votre IDE python (**pyzo**,**spyder**,etc..) vous pouvez vous référez à l'aide suivante : Stackoverflow

Sinon via un terminal de commandes vous devez exécuter les commandes suivantes si vous utilisez Anaconda.

`shell conda install redis conda install pandas conda install numpy conda install scipy conda install seaborn`

Ou bien celle-ci pour l'utilisation de pip `shell pip install redis pip install pandas pip install numpy pip install scipy pip install seaborn`

3.3 Importation des librairies dans python :

[44]:
```
import redis
import numpy
import scipy
import random
import pprint
import json
import pandas as pd
import seaborn as sns
import time
```

3.4 Connexion

Commes les autres SGBD, il faut établir une connexion avec Redis pour communiquer avec le SGBD lors de nos requêtes python.

Ici on fixe les paramètres de connexion (commun pour tous) pour le nom de l'hôte `redis_host` et le numéro de port serveur `redis_port`.

[45]:
```
redis_host = "127.0.0.1"
redis_port = 6379
```

On peux ainsi établir la connexion avec la commande `redis.Redis` en précisant le numéro de la database que l'on veux utiliser (par défaut 16 database disponible sur un serveur Redis).

[46]:
```
r = redis.Redis(host=redis_host, port=redis_port, db = 0)
```

3.5 Nous sommes prêts pour faire quelques requêtes et explorer Redis!

Nous avons choisi un exemple pour illustrer la force de Redis : la gestion de stock de sushi **(miam)**.

Voici une photo d'un échantillon de sushis variés sushi
source:https://www.nippon.com/fr/ncommon/contents/japan-data/174999/174999.jpg

4 Création de notre base de données

Nous avons choisi de créer notre base de données de sushi en indiquant : - la composition d'un sushi selon les ingrédients qui le composent : 73 ingrédients différents, qui peuvent prendre la modalité 'Oui' (le sushi contient cet ingrédient) ou 'Non' (le sushi ne contient pas cet ingrédient) - le stock de ce sushi disponible : généré aléatoirement entre 10 et 10000 - le nombre de sushi acheté : initialisé à 0

[47]:
```
## Création de la database sushis

random.seed(444)

# Liste des attributs avec 2 modalités : les ingrédients
saumon = ['Oui', 'Non']
saumon_teriyaki = ['Oui', 'Non']
daurade = ['Oui', 'Non']
thon = ['Oui', 'Non']
crevette = ['Oui', 'Non']
poulet = ['Oui', 'Non']
thon_cuit = ['Oui', 'Non']
foie_gras = ['Oui', 'Non']
tofu = ['Oui', 'Non']
truite = ['Oui', 'Non']
hareng = ['Oui', 'Non']
poulpe = ['Oui', 'Non']
boeuf = ['Oui', 'Non']
chair_de_crabe = ['Oui', 'Non']
oeufs_de_saumon = ['Oui', 'Non']
avocat = ['Oui', 'Non']
fromage = ['Oui', 'Non']
oeuf = ['Oui', 'Non']
gingembre = ['Oui', 'Non']
wasabi = ['Oui', 'Non']
sauce_salee = ['Oui', 'Non']
sauce_sucree = ['Oui', 'Non']
sauce_sucreesalee = ['Oui', 'Non']
mayonnaise_classique = ['Oui', 'Non']
mayonnaise_teriyaki = ['Oui', 'Non']
mayonnaise_japonaise = ['Oui', 'Non']
mayonnaise_spicy = ['Oui', 'Non']
mayonnaise_ponzu = ['Oui', 'Non']
```

```
sauce_teriyaki = ['Oui', 'Non']
sauce_satay_aux_cacahuetes = ['Oui', 'Non']
sauce_epicee = ['Oui', 'Non']
mangue = ['Oui', 'Non']
carotte = ['Oui', 'Non']
anis = ['Oui', 'Non']
poivre_rose = ['Oui', 'Non']
cannelle = ['Oui', 'Non']
cardamome = ['Oui', 'Non']
curcuma = ['Oui', 'Non']
feve = ['Oui', 'Non']
edamame = ['Oui', 'Non']
macis = ['Oui', 'Non']
maniguette = ['Oui', 'Non']
paprika = ['Oui', 'Non']
piment = ['Oui', 'Non']
poivre = ['Oui', 'Non']
safran = ['Oui', 'Non']
sumac = ['Oui', 'Non']
persil = ['Oui', 'Non']
herbe_de_provence = ['Oui', 'Non']
sesame = ['Oui', 'Non']
feuilles_de_riz = ['Oui', 'Non']
menthe = ['Oui', 'Non']
coriandre = ['Oui', 'Non']
chou = ['Oui', 'Non']
ciboulette = ['Oui', 'Non']
pomme = ['Oui', 'Non']
celeri_rave = ['Oui', 'Non']
aneth = ['Oui', 'Non']
baies_roses = ['Oui', 'Non']
prune = ['Oui', 'Non']
betterave = ['Oui', 'Non']
noix_de_coco = ['Oui', 'Non']
citron_vert = ['Oui', 'Non']
citron_jaune = ['Oui', 'Non']
dattes = ['Oui', 'Non']
laitue = ['Oui', 'Non']
roquette = ['Oui', 'Non']
concombre = ['Oui', 'Non']
poivrons = ['Oui', 'Non']
asperge = ['Oui', 'Non']
oignons_crus = ['Oui', 'Non']
oignons_caramelises = ['Oui', 'Non']
oignons_frits = ['Oui', 'Non']

# 2**73 = 9.5*(10**21) combinaisons possibles de sushis
```

```
sushis = []

tps1 = time.perf_counter()

for i in range(0, 1000):
    saumon_choice = random.choice(saumon)
    saumon_teriyaki_choice = random.choice(saumon_teriyaki)
    daurade_choice = random.choice(daurade)
    thon_choice = random.choice(thon)
    crevette_choice = random.choice(crevette)
    poulet_choice = random.choice(poulet)
    thon_cuit_choice = random.choice(thon_cuit)
    foie_gras_choice = random.choice(foie_gras)
    tofu_choice = random.choice(tofu)
    truite_choice = random.choice(truite)
    hareng_choice = random.choice(hareng)
    poulpe_choice = random.choice(poulpe)
    boeuf_choice = random.choice(boeuf)
    chair_de_crabe_choice = random.choice(chair_de_crabe)
    oeufs_de_saumon_choice = random.choice(oeufs_de_saumon)
    avocat_choice = random.choice(avocat)
    fromage_choice = random.choice(fromage)
    oeuf_choice = random.choice(oeuf)
    gingembre_choice = random.choice(gingembre)
    wasabi_choice = random.choice(wasabi)
    sauce_salee_choice = random.choice(sauce_salee)
    sauce_sucree_choice = random.choice(sauce_sucree)
    sauce_sucreesalee_choice = random.choice(sauce_sucreesalee)
    mayonnaise_classique_choice = random.choice(mayonnaise_classique)
    mayonnaise_teriyaki_choice = random.choice(mayonnaise_teriyaki)
    mayonnaise_japonaise_choice = random.choice(mayonnaise_japonaise)
    mayonnaise_spicy_choice = random.choice(mayonnaise_spicy)
    mayonnaise_ponzu_choice = random.choice(mayonnaise_ponzu)
    sauce_teriyaki_choice = random.choice(sauce_teriyaki)
    sauce_satay_aux_cacahuetes_choice = random.
↪choice(sauce_satay_aux_cacahuetes)
    sauce_epicee_choice = random.choice(sauce_epicee)
    mangue_choice = random.choice(mangue)
    carotte_choice = random.choice(carotte)
    anis_choice = random.choice(anis)
    poivre_rose_choice = random.choice(poivre_rose)
    cannelle_choice = random.choice(cannelle)
    cardamome_choice = random.choice(cardamome)
    curcuma_choice = random.choice(curcuma)
    feve_choice = random.choice(feve)
    edamame_choice = random.choice(edamame)
```

```
    macis_choice = random.choice(macis)
    maniguette_choice = random.choice(maniguette)
    paprika_choice = random.choice(paprika)
    piment_choice = random.choice(piment)
    poivre_choice = random.choice(poivre)
    safran_choice = random.choice(safran)
    sumac_choice = random.choice(sumac)
    persil_choice = random.choice(persil)
    herbe_de_provence_choice = random.choice(herbe_de_provence)
    sesame_choice = random.choice(sesame)
    feuilles_de_riz_choice = random.choice(feuilles_de_riz)
    menthe_choice = random.choice(menthe)
    coriandre_choice = random.choice(coriandre)
    chou_choice = random.choice(chou)
    ciboulette_choice = random.choice(ciboulette)
    pomme_choice = random.choice(pomme)
    celeri_rave_choice = random.choice(celeri_rave)
    aneth_choice = random.choice(aneth)
    baies_roses_choice = random.choice(baies_roses)
    prune_choice = random.choice(prune)
    betterave_choice = random.choice(betterave)
    noix_de_coco_choice = random.choice(noix_de_coco)
    citron_vert_choice = random.choice(citron_vert)
    citron_jaune_choice = random.choice(citron_jaune)
    dattes_choice = random.choice(dattes)
    laitue_choice = random.choice(laitue)
    roquette_choice = random.choice(roquette)
    concombre_choice = random.choice(concombre)
    poivrons_choice = random.choice(poivrons)
    asperge_choice = random.choice(asperge)
    oignons_crus_choice = random.choice(oignons_crus)
    oignons_caramelises_choice = random.choice(oignons_caramelises)
    oignons_frits_choice = random.choice(oignons_frits)
    stock = random.randint(10,10000)
    nb_achat = 0
    sushi = {'id':i, 'saumon':saumon_choice, 'saumon_teriyaki':
↪saumon_teriyaki_choice, 'daurade':daurade_choice,
             'thon':thon_choice, 'crevette':crevette_choice, 'poulet':
↪poulet_choice, 'thon_cuit':thon_cuit_choice,
             'foie_gras':foie_gras_choice,'tofu':tofu_choice, 'truite':
↪truite_choice, 'hareng':hareng_choice,
             'poulpe':poulpe_choice, 'boeuf':boeuf_choice, 'chair_de_crabe':
↪chair_de_crabe_choice,
             'oeufs_de_saumon':oeufs_de_saumon_choice, 'avocat':avocat_choice,␣
↪'fromage':fromage_choice,
             'oeuf':oeuf_choice, 'gingembre':gingembre_choice,'wasabi':
↪wasabi_choice, 'sauce_salee':sauce_salee_choice,
```

```
            'sauce_sucree':sauce_sucree_choice,'sauce_sucreesalee':
↪sauce_sucreesalee_choice,
            'mayonnaise_classique':mayonnaise_classique_choice,␣
↪'mayonnaise_teriyaki':mayonnaise_teriyaki_choice,
            'mayonnaise_japonaise':mayonnaise_japonaise_choice,␣
↪'mayonnaise_spicy':mayonnaise_japonaise_choice,
            'mayonnaise_ponzu':mayonnaise_ponzu_choice, 'sauce_teriyaki':
↪sauce_teriyaki_choice,
            'sauce_satay_aux_cacahuetes':sauce_satay_aux_cacahuetes_choice,␣
↪'sauce_epicee':sauce_epicee_choice,
            'mangue': mangue_choice, 'carotte':carotte_choice, 'anis':
↪anis_choice,'poivre_rose':poivre_rose_choice,
            'cannelle':cannelle_choice, 'cardamome':cardamome_choice,␣
↪'curcuma':curcuma_choice,
            'feve':feve_choice, 'edamame':edamame_choice, 'macis':
↪macis_choice, 'maniguette':maniguette_choice,
            'paprika':paprika_choice, 'piment':piment_choice, 'poivre':
↪poivre_choice, 'safran':safran_choice,
            'sumac': sumac_choice, 'persil':persil_choice, 'herbe_de_provence':
↪herbe_de_provence_choice,
            'sesame':sesame_choice, 'feuilles_de_riz':feuilles_de_riz_choice,␣
↪'menthe':menthe_choice,
            'coriandre':coriandre_choice, 'chou':chou_choice, 'ciboulette':
↪ciboulette_choice, 'pomme':pomme_choice,
            'celeri_rave':celeri_rave_choice, 'aneth':aneth_choice,␣
↪'baies_roses':baies_roses_choice,
            'prune':prune_choice, 'betterave':betterave_choice, 'noix_de_coco':
↪noix_de_coco_choice,
            'citron_vert':citron_vert_choice, 'citron_jaune':
↪citron_jaune_choice, 'dattes':dattes_choice,
            'laitue':laitue_choice, 'roquette':roquette_choice, 'concombre':
↪concombre_choice,
            'poivrons':poivrons_choice, 'asperge':asperge_choice,␣
↪'oignons_crus':oignons_crus_choice,
            'oignons_caramelises':oignons_caramelises_choice, 'oignons_frits':
↪oignons_frits_choice,'stock':stock, 'nb_achat':nb_achat
           }

    sushis.append(sushi)

tps2 = time.perf_counter()
print(tps2 - tps1)
```

```
0.044649485999798344
```

Le temps pour créer une liste de 1 million de sushis différents est d'environ A COMPLETER

4.0.1 On ajoute ensuite nos données générées dans redis

Une des forces de redis est l'utilisation de pipeline pour réaliser les requêtes, nous y reviendrons en fin de rapport. Pour plus d'informations, rendez-vous ici.

[48]:
```
with r.pipeline() as pipe:
    for i in range(len(sushis)):
        name = str("sushi:")+str(i)
        r.hmset(name,sushis[i])
```

```
<ipython-input-48-4b3353b65ed5>:4: DeprecationWarning: Redis.hmset() is
deprecated. Use Redis.hset() instead.
  r.hmset(name,sushis[i])
```

On peux ensuite effectuer quelques requêtes simples avec les objets **hashs** que nous utiliserons au cours de notre projet. Pour plus d'informations sur les requêtes sur les types d'objets **hashs**, rendez-vous ici.

Ainsi, comme exemple ici, nous avons nos clés avec la dénomination *sushi:i* avec i le numéro du sushi allant de 1 à 1000000. Nous pouvons alors récupérer au sein de chaque clés différentes informations stockés sous une forme similaire à un dictionnaire.

[49]:
```
stock_exemple_propre = int(r.hget('sushi:3','stock').decode())
stock_exemple_bytes = r.hget('sushi:3','stock')
print("Propre:",stock_exemple_propre,"  ","Bytes:",stock_exemple_bytes)
```

```
Propre: 7786    Bytes: b'7786'
```

Ainsi, on récupère le stock (désigné par `stock` du *sushi:3*. Notez l'utilisation de `int()` et `.decode()` pour récupérez sous un format plus "agréable" les requêtes de redis qui sont retournées par défaut au format **bytes**.

Nous allons ensuite définir quelques fonctions utiles pour réaliser quelques requêtes. Pour cela nous aurons besoin de définir certains messages d'erreurs.

[50]:
```
# on définit notre message d'erreur qu'on utilisera pour prévenir l'utilisateur␣
↪de la fin du stock d'un objet

class OutOfStockError(Exception):
    """Raised when sushi.com is out of stock for the desired sushi"""

class TooMuchStockError(Exception):
    """Raised when sushi.com is full of the desired sushi"""

class TooMuchDemandError(Exception):
    """Raised when sushi.com has not enough sushi asked"""

class NoPlaceAvailableError(Exception):
    """Raised when sushi.com has restocké les 10000 sushis maximum."""
```

Nous allons maintenant créer la fonction `buyitem`. Celle-ci permet d'acheter un certain nombre de sushis (argument `count`) d'un certain type (argument `sushi_id`) : cela diminue-donc le stock de sushis, mais augmente le nombre de sushis achetés.

[51]:
```
def buyitem(r, sushi_id,count):

    with r.pipeline() as pipe:

        while True:

            pipe.watch(sushi_id)
            nleft = int(r.hget(sushi_id,'stock').decode())
            # On utilise le int et decode pour convertir l'output bytes de␣
↪redis en nombre int

            # On réalise l'opération d'achat normalement s'il y a assez de␣
↪sushis en stock
            if nleft > count :
                pipe.multi() # On débute l'enregistrement des étapes qu'on veux␣
↪réaliser sur le serveur redis
                pipe.hincrby(sushi_id, 'stock', -count) # Le stock diminue ...
                pipe.hincrby(sushi_id, 'nb_achat', count) # ... Et le nombre de␣
↪sushis achetés augmente
                pipe.execute() # On transfert l'ensemble des étapes réalisée␣
↪depuis multi sur notre serveur redis
                break

            # Si le nombre de sushis n'est pas suffisant pour satisfaire la␣
↪demande, on achète tout ce qu'il reste
            elif nleft < count and nleft > 0:
                pipe.multi()
                pipe.hset(sushi_id,'stock',0)
                pipe.hincrby(sushi_id,'nb_achat',nleft)
                pipe.execute()

                raise TooMuchDemandError(f'Désolé, vous avez acheté tout les␣
↪{sushi_id} restants mais il n''y en avait pas autant que vous le vouliez')

            # S'il n'y a plus du tout de ce type de sushis ...
            else:
                pipe.unwatch()
                raise OutOfStockError(f"Sorry, {sushi_id} is out of stock!")

    return None
```

Testons maintenant cette fonction : et si on achetait 60 sushis de type 5 ?

[52]:
```
buyitem(r,"sushi:5",60)
r.hmget("sushi:5","stock","nb_achat")
```

[52]:
```
[b'6165', b'60']
```

Nous allons maintenant créer la fonction `restock`. Celle-ci permet au magasin de se réapprovisionner d'un certain nombre (argument `stock_count`) de sushis d'un certain type (argument `sushi_id`) : le stock de ce type de sushis augmente ainsi. Arbitrairement, nous avons choisi qu'un magasin ne pouvait stocker que 10 000 sushis d'un même type (c'est déjà pas mal).

[53]:
```
def restock(r,sushi_id,stock_count):

    with r.pipeline() as pipe:

        while True:

            pipe.watch(sushi_id)
            nleft_restock = int(r.hget(sushi_id,'stock').decode())

            # S'il y a suffisamment de places pour stocker les sushis, on␣
↪achète le nombre demandé
            if nleft_restock + stock_count < 10000:
                pipe.multi()
                pipe.hincrby(sushi_id, 'stock', stock_count)
                pipe.execute()
                break

            # S'il n'y a pas suffisamment de place pour stocker tous les␣
↪nouveaux sushis, on achète seulement ce que l'on peut
            elif nleft_restock < 10000 and nleft_restock + stock_count > 10000:
                pipe.multi()
                pipe.hset(sushi_id,'stock',10000)
                pipe.execute()
                raise NoPlaceAvailableError(f"Désolé, nous avons déjà remis le␣
↪stock de {sushi_id} au maximum")

            # Si le stock est déjà plein, cela ne sert à rien d'acheter␣
↪davantage de sushis !
            # (et de toute façon on ne peut pas les stocker)
            else:
                pipe.unwatch()
                raise TooMuchStockError(f"Sorry, {sushi_id} is already full of␣
↪stock!")

    return None
```

Testons maintenant cette fonction : et si on augmentait le stock du sushi 9 de 400 ?

[54]:
```
restock(r,'sushi:9', 400)
r.hmget("sushi:4",'stock','nb_achat')
```

[54]:
```
[b'9207', b'0']
```

Nous allons maintenant créer la fonction `item_info`. Celle-ci permet de récupérer toutes les informations relatives à nos ventes de sushis : stock, vente. Ces informations seront ensuite stockées dans un Dataframe Pandas.

[55]:
```
def item_info():

    info_get = [] # Tableau qui contiendra les données récupérées

    for i in range(len(sushis)):
        name = str("sushi:")+str(i)
        stock = int(r.hget(name,'stock').decode())
        nb_achat = int(r.hget(name,'nb_achat').decode())
        info_add = [stock, nb_achat]
        info_get.append(info_add)

    # On convertit notre liste de liste en dataframe
    item_df = pd.DataFrame(info_get,
                           columns=['stock','nb_achat'],
                           index = [str("sushi:")+str(i) for i in␣
↪range(len(sushis))])

    return(item_df)
```

[56]:
```
liste_sushi = item_info()
liste_sushi
```

[56]:

	stock	nb_achat
sushi:0	4846	0
sushi:1	1596	0
sushi:2	6646	0
sushi:3	7786	0
sushi:4	9207	0
…	…	…
sushi:995	7978	0
sushi:996	4278	0
sushi:997	833	0
sushi:998	4246	0
sushi:999	8563	0

[1000 rows x 2 columns]

[57]:
```
# A ENLEVER

#sns.set()
#sns.barplot(data=liste_sushi,x=liste_sushi.index,y='stock')
#sns.barplot(data=liste_sushi,x=liste_sushi.index,y='nb_achat')
```

La fonction `item_ingredients`, elle, permet de récupérer les ingrédients de nos différents types de sushis (73 variables donc, correspondant à l'absence ou à la présence de chacun des ingrédients). Ces informations seront ensuite stockées dans un Dataframe Pandas.

[58]:
```
def item_ingredient():

    ingredient_get = []

    for i in range(len(sushis)):

        name = str("sushi:")+str(i)
        ingredient_add = []

        name_ingredient = ['saumon', 'saumon_teriyaki', 'daurade', 'thon',␣
↪'crevette',
                                                'poulet', 'thon_cuit', 'foie_gras',␣
↪'tofu', 'truite', 'hareng',
                                                'poulpe', 'boeuf',␣
↪'chair_de_crabe', 'oeufs_de_saumon', 'avocat',
                                                'fromage', 'oeuf', 'gingembre',␣
↪'wasabi', 'sauce_salee',
                                                'sauce_sucree',␣
↪'sauce_sucreesalee', 'mayonnaise_classique',
                                                'mayonnaise_teriyaki',␣
↪'mayonnaise_japonaise', 'mayonnaise_spicy',
                                                'mayonnaise_ponzu',␣
↪'sauce_teriyaki', 'sauce_satay_aux_cacahuetes',
                                                'sauce_epicee', 'mangue',␣
↪'carotte', 'anis', 'poivre_rose',
                                                'cannelle', 'cardamome', 'curcuma',␣
↪'feve', 'edamame', 'macis',
                                                'maniguette', 'paprika', 'piment',␣
↪'poivre', 'safran', 'sumac',
                                                'persil', 'herbe_de_provence',␣
↪'sesame', 'feuilles_de_riz',
                                                'menthe', 'coriandre', 'chou',␣
↪'ciboulette', 'pomme',
                                                'celeri_rave', 'aneth',␣
↪'baies_roses', 'prune', 'betterave',
```

```
                                                'noix_de_coco', 'citron_vert',␣
↪'citron_jaune', 'dattes',
                                                'laitue', 'roquette', 'concombre',␣
↪'poivrons', 'asperge',
                                                'oignons_crus',␣
↪'oignons_caramelises', 'oignons_frits']

        # La boucle permet de rentrer l'ensemble des ingrédients dans le tableau
        for i in range(len(name_ingredient)):
            ingredient = r.hget(name, name_ingredient[i]).decode()
            ingredient_add.append(ingredient)

        ingredient_get.append(ingredient_add)

    ingredient_df = pd.DataFrame(ingredient_get,
                         columns=name_ingredient,
                         index = [str("sushi:")+str(i) for i in␣
↪range(len(sushis))])

    return(ingredient_df)
```

[59]:
```
ingredient_df = item_ingredient()
ingredient_df
```

[59]:

	saumon	saumon_teriyaki	daurade	thon	crevette	poulet	thon_cuit	\
sushi:0	Non	Non	Oui	Non	Non	Non	Non	
sushi:1	Oui	Oui	Non	Oui	Non	Oui	Non	
sushi:2	Non	Oui	Oui	Oui	Non	Oui	Non	
sushi:3	Oui	Non	Non	Non	Non	Oui	Non	
sushi:4	Oui	Non	Non	Oui	Non	Non	Non	
…	…	…	…	…	…	…	…	
sushi:995	Oui	Oui	Oui	Non	Non	Non	Non	
sushi:996	Non	Oui	Non	Oui	Oui	Oui	Oui	
sushi:997	Non	Oui	Oui	Non	Non	Oui	Oui	
sushi:998	Oui	Oui	Oui	Oui	Non	Non	Oui	
sushi:999	Oui	Non	Oui	Oui	Non	Oui	Non	

	foie_gras	tofu	truite	…	citron_jaune	dattes	laitue	roquette	\
sushi:0	Non	Non	Non	…	Oui	Non	Non	Oui	
sushi:1	Oui	Non	Non	…	Non	Non	Non	Non	
sushi:2	Non	Non	Non	…	Oui	Non	Oui	Non	
sushi:3	Non	Non	Non	…	Oui	Oui	Oui	Non	
sushi:4	Non	Non	Non	…	Oui	Oui	Non	Oui	
…	…	…	…	…	…	…	…	…	
sushi:995	Oui	Oui	Oui	…	Oui	Oui	Oui	Oui	
sushi:996	Oui	Oui	Oui	…	Oui	Oui	Oui	Oui	
sushi:997	Oui	Non	Oui	…	Oui	Oui	Non	Non	

```
sushi:998       Non  Oui    Oui  …          Non    Oui    Non      Oui
sushi:999       Non  Oui    Non  …          Non    Oui    Non      Oui

          concombre poivrons asperge oignons_crus oignons_caramelises  \
sushi:0         Non      Non     Oui          Oui                  Oui
sushi:1         Non      Oui     Oui          Non                  Oui
sushi:2         Non      Non     Oui          Non                  Non
sushi:3         Oui      Non     Oui          Oui                  Oui
sushi:4         Oui      Oui     Non          Oui                  Non
…             …      …     …          …                  …
sushi:995       Oui      Oui     Oui          Oui                  Non
sushi:996       Non      Oui     Oui          Non                  Oui
sushi:997       Non      Oui     Oui          Non                  Non
sushi:998       Oui      Non     Non          Oui                  Oui
sushi:999       Oui      Non     Oui          Oui                  Non

          oignons_frits
sushi:0             Oui
sushi:1             Oui
sushi:2             Non
sushi:3             Non
sushi:4             Non
…                 …
sushi:995           Oui
sushi:996           Oui
sushi:997           Oui
sushi:998           Oui
sushi:999           Oui

[1000 rows x 73 columns]
```

La fonction `sushi_interest`, définie ci-dessous, est très pratique pour les clients. Ceux-ci peuvent entrer en arguments une liste des ingrédients qu'ils aiment dans les sushis, et la fonction leur retourne tous les sushis de la carte qui contiennent l'ensemble de ces ingrédients.

[60]:
```python
def sushi_interet(r,liste_ingredients,ingredient_df):

    sushi_interessant = [] # Liste qui stockera les numéros de tous les sushis␣
↪contenant tous les ingrédients de liste_ingredients
    for j in range(len(ingredient_df)): # On prend tous les sushis␣
↪successivement ...
        if all(ingredient_df.iloc[j][liste_ingredients]=="Oui"): # ... et on␣
↪regarde s'ils contiennent tous les ingrédients demandés
            sushi_interessant.append(j)

    phrase = "Vous pourriez apprécier les sushis suivants : "
    for i in range(len(sushi_interessant)):
```

```
        phrase += "sushi:"+str(sushi_interessant[i])+" "

    print(phrase)
    return(sushi_interessant)
```

Faisons un essai : je suis un client qui aime le saumon, l'avocat, le thon, la roquette, le concombre et la cannelle, et souhaiterait un sushi qui mélange toutes ces saveurs.

[62]:
```
liste_ingredients = ['saumon','avocat','thon','roquette','concombre','cannelle']
sushi_interet(r,liste_ingredients,ingredient_df)
```

```
Vous pourriez apprécier les sushis suivants : sushi:158 sushi:315 sushi:371
sushi:520 sushi:586 sushi:661 sushi:683 sushi:731 sushi:769 sushi:929 sushi:998
```

[62]:
```
[158, 315, 371, 520, 586, 661, 683, 731, 769, 929, 998]
```

4.0.2 Eventuellement inclure :

- rajouter des options de monitoring du stock des objets et complexifier un peu l'offre donc voir modifier les chsoes au début pour faire genre de la gestion de maxi database de vetements
- une visu type barchat du stock pour chaque produit avec les requetes sur la db? à voir comment faire je maitrise pas trop ça
- des visus complémentaires informant sur l'état des stocks, évolution des achats etc… à voir

faire des graphs "big data"

- genre faire une représentation de l'évolution du stock/nb_achat (total de sushi?) au cours du temps
- faire le camembert de proportion de sushi qui contient tel ingredient

on s'intéresse à l'expiration et à comment on pourrait l'utiliser en pratique

[375]:
```
from datetime import timedelta
r.setex("runner",timedelta(minutes=1.5), value =" careful! ")
r.ttl("runner")
```

[375]:
```
90
```

4.0.3 on va voir comment on peux ajouter des sushis temporaires ! avec de superbes recettes miam !

- pour cela on se sert de la fonctionnalité EXPIRE présente dans redis, pour en savoir plus allez ici
- **ce serait bien d'avoir un système de surveillance des perfs de notre server redis**

[397]:
```
# sert a fixer la durée de vie d'un sushi ! par ex simuler une vente temporaire␣
↪!

r.expire('sushi:3',15)
```

```
# tant que la clé est encore en vie on peux la manipuler comme bon nous semble␣
 ↪et récupérer les informations que l'on veux
r.hgetall('sushi:3')
```

[397]: True

[ ]:
```
def life_expire(r,life_count)
```

[ ]:
```

```

4.0.4 Qualitative approach

We are looking for differences in : - grammar - ease of use and prise en main - setting up process (manipulation of our first database) - access of ressources

4.0.5 on parle de redis ici de son usage

- prise ne main très facile et super documentation tout est renseigné on comprend assez vite son fonctionnement
- difficile de se projeter dans l'usage de ce sgbd car différents des autres et assez spécifique même si on peux l'utilsier pour tout usage il sera surtout performant dans des tâches spé.
- open source et accessible facilement mais l'environnement autour non car le proejt à été repris par une entreprise redislabs et typiquement un redis-GUI (logiciel pour manipuler des db redis sans lignes de commandes) n'est pas accessible facilement
- une bonne communauté mais pas très large donc accès ressources limités

4.0.6 Quantitative approach

We are looking for differences in : - time computation of requests - memore/cache use for same data

ex : système pour gérer els incrémentation lors de connexion sur srveeur/page web simultanée etc.. gère bien ça

gere automatiquement l'expiration des informations7

système de persistance : - RDB : on crée un dump file en .rdb qui est sauvegardé et qui peux etre rechargé - AOF : ??? a voir

Expliquer histoire de redis les points clés et différences par rapport aux autres sql : - cache et session bonne gestion - systeme cle-valeur avancé incr, decincr etc.. pas mal puissant - L'une des raisons pour lesquelles Redis est si rapide dans les opérations de lecture et d'écriture est que la base de données est conservée en mémoire (RAM) sur le serveur.