Écriture du code Projet 5 développement Python

Création de la base de données

```sql
CREATE DATABASE P5;

USE P5;

CREATE TABLE CATEGORIES
(ID INT UNSIGNED AUTO_INCREMENT NOT NULL,
NOM VARCHAR(80),
LINK_OFF VARCHAR(80),
PRIMARY KEY (ID)
)ENGINE=INNODB;

CREATE TABLE PRODUIT(
PRODUIT_ID INT UNSIGNED AUTO_INCREMENT NOT NULL,
NOM VARCHAR(200),
PRODUIT_URL VARCHAR(150),
STORE VARCHAR(300),
NUTRISCORE INT,
CATEGORIE_ID INT UNSIGNED,
PRIMARY KEY (PRODUIT_ID),
CONSTRAINT fk_ok
FOREIGN KEY (CATEGORIE_ID) REFERENCES CATEGORIES (ID)
)ENGINE=INNODB;


CREATE TABLE SUBSTITUT(
SUBSTITUT_ID INT UNSIGNED AUTO_INCREMENT NOT NULL,
NOM VARCHAR(80),
STORE VARCHAR(80),
SUBSTITUT_URL VARCHAR(100),
DESCRIPTION VARCHAR(200),
PRIMARY KEY (SUBSTITUT_ID)
)ENGINE=INNODB;
```

Connexion à la base de données et initialisation

P5.py × constantes.py ×

```python
# coding: utf-8
import requests
import json
import pymysql
import pymysql.cursors
import math
import copy
from constantes import *
"""CONNECT TO THE DATABASE"""
try :
    connection = pymysql.connect(host=HOST, #variable in file constantes.py
                                 user=USER,
                                 password=PASSWORD,
                                 db=DB,
                                 charset='utf8mb4',
                                 port = PORT,
                                 cursorclass=pymysql.cursors.DictCursor)
except :
        print("erreur de connexion")

```

```python
def cleaning_tables(NAME_TABLE = CATEGORIES):

    """CLEANING TABLES"""

    with connection.cursor() as cursor:
        sql = "DELETE FROM %s;" %(NAME_TABLE)
        cursor.execute(sql, ())
        connection.commit()
def reset_counter(NAME_TABLE = CATEGORIES):
    """RESET THE COUNTERS"""

    with connection.cursor() as cursor:
        sql = "ALTER TABLE %s AUTO_INCREMENT=0;" % (NAME_TABLE)
        cursor.execute(sql, ())
        connection.commit()

```

Création d’un menu utilisateur 1/2 : Boucle principal

```
"""MAIN LOOP"""

continu = 0
while continu == 0 :
    try :
        print(transition)
        terminal_mode = input("\n1 - Quel aliment souhaitez-vous remplacer ? \n2 - Retrouver mes aliments substitués. \n3 - Sortir du programme ? \n4 - Nettoyer la Base de données")
        terminal_mode = int(terminal_mode)

        """mode remplacement"""
        while terminal_mode == 1 : ...

        """Consult database"""

        while terminal_mode == 2 : ...

        """exit mode"""

        if terminal_mode == 3 : ...

        if terminal_mode == 4 : ...

        elif terminal_mode > 4 :
            print("{} n'est pas dans les numéros proposés\n".format(terminal_mode))

    except ValueError :
        if len(terminal_mode) > 1 :
            print("\nOops! {} est un mot, veuillez recommencer : \n".format(terminal_mode))
        else :
            print("\nOops! {} est une lettre, veuillez recommencer : \n".format(terminal_mode))
```

Création d’un menu utilisateur 2/2 choix d’une catégorie

```
"""mode remplacement"""
while terminal_mode == 1 :
    print(transition)
    """condition for mode_substitut catégorie"""
    print(mode_substitut_categorie)
    temp_categorie = int
    try :
        keyboard_input = input ("choissisez votre catégorie de produit : " + propostion)
        keyboard_input = int(keyboard_input)
        print(type(keyboard_input))
        choice = str
        name_categories = str

        if keyboard_input in nb_series :
            choice = categories[keyboard_input - 1]
            name_categories = categories[keyboard_input - 1]
            link_categories = link[l = keyboard_input - 1)
            temp_categorie = keyboard_input
            print(temp_categorie)
            print("model")
        elif keyboard_input not in nb_series :
            print("{} n'est pas dans les numéros proposés\n".format(keyboard_input))
            break

        if keyboard_input in mode_substitut_categorie :
            print("catégories {} existante dans la BDD".format(name_categories))
            select_and_substitut()
        else :
            with connection.cursor() as cursor:
                sql = "INSERT INTO `CATEGORIES` (`NOM`,`LINK_OFF`) VALUES (%s, %s)"
                cursor.execute(sql, (name_categories, link_categories))
            mode_substitut_categorie.append(temp_categorie)
            connection.commit()
            Add_product()
            print(transition)
            select_and_substitut()

    except ValueError :
        print("\nOops! {} est une lettre, veuillez recommencer : \n".format(keyboard_input))

    terminal_mode = 0

"""Consult database"""

while terminal_mode == 2 :
    print(transition)
    print("Voici vos produits : \n ")
    with connection.cursor() as cursor:
        sql = "SELECT `NOM`,`STORE`,`SUBSTITUT_ID` FROM `SUBSTITUT`"
        cursor.execute(sql, ())
        my_product = cursor.fetchall()
        print(my_product)
```

Utilisation du module json pour récupérer les données d'une API *1/2*

```
68
69  """PROGRAM FUNCTIONS"""
70  def Add_product():
71      def AlimenterListeProduit():
72
73          for data in product ["products"]:
74              """nutriscore = (data["nutrition_grades"])"""
75              url.append((data["url"]))
76
77              name.append((data["product_name"]))
78
79              ns.append((data["nutrition_grades_tags"]))
80      """creation des listes"""
81
82      url = []
83      store = []
84      name = []
85      ns = []
86
87      """ adaptation liens en fonctions du choix"""
88      link_product = choice
89
90      """requêtes get"""
91      r = requests.get('https://fr.openfoodfacts.org/categorie/' + link_product + '/1.json')
92      """dynamic link"""
93      dynamic_link = 'https://fr.openfoodfacts.org/categorie/' + link_product + '/1.json'
94      print(dynamic_link)
95      """liens catégorie figé"""
96      l = str
97      l = r
98      """lecture json"""
99      product = r.json()
100     """on remplie la liste des produits"""
101     AlimenterListeProduit()
102     """calcul du nombre de produits"""
103     count = product['count'] # Le nombre total de produits
104
105     page_size = product['page_size'] # La taille d'une page
106
107     nbPages = int( math.floor(count / page_size) + 1) # On déduit le nombre de pages
108     print("nombre de pages = " +str(nbPages))
109     i = 2
110     while i <= nbPages:
111
112         r = requests.get('https://fr.openfoodfacts.org/categorie/' + link_product + "/" + str(i)+ '.json')
113         dynamic_link = 'https://fr.openfoodfacts.org/categorie/' + link_product + "/" + str(i)+ '.json'
114         print(dynamic_link)
115         product = r.json()
116         AlimenterListeProduit()
117         print("Page " + str(i) + " récupérée !")
118         print(len(url))
119         i += 1
```

Utilisation du module json pour récupérer les données d'une API 2/2

```
    i += 1

nb_product = (len(url))
print(nb_product)
ns_number = []
ns = str(ns)
ns_simple = []
ns_number = []


for x in ns :
    if x in ("a","b","c","d","e") :
        ns_simple.append(x)


def convert(score = ns_simple):
    for i in score :
        if i == 'a' :
            ns_number.append(1)
        elif i == "b" :
            ns_number.append(2)
        elif i == "c" :
            ns_number.append(3)
        elif i == "d" :
            ns_number.append(4)
        elif i == "e" :
            ns_number.append(5)
    return ns_number
convert()
"""récupération de l'ID"""
```

Utilisation de la Fonction INSERT en MYSQL

```
145                 ns_number.append(5)
146         return ns_number
147     convert()
148     """récupération de l'ID"""
149     with connection.cursor() as cursor: ...
154
155     print(ID)
156     ID = str(ID)
157     N_ID = ""
158     for x in ID : ...
161
162     print(N_ID)
163     N_ID = int(N_ID)
164
165     """INSERT TO BDD"""
166
167     xy = 0
168     try :
169         while xy < nb_product :
170             with connection.cursor() as cursor:
171                 sql = "INSERT INTO `PRODUIT` (`NOM`,`PRODUIT_URL`,`NUTRISCORE`, `CATEGORIE_ID`) VALUES (%s, %s, %s, %s)"
172                 cursor.execute(sql, (name[xy], url[xy], ns_number[xy], N_ID))
173
174             connection.commit()
175             xy = xy + 1
176             print(str(xy)+"produits récupérés sur " + str(nb_product))
177     except :
178         pass
179
180
181 def select_and_substitut(): ...
249
250 """MAIN LOOP"""
251
252 continu = 0
253 while continu == 0 : ...)
```

Utilisation des opérations de jointures en MYSQL

```
 69  """PROGRAM FUNCTIONS"""
 70  def Add_product():...
178  def select_and_substitut():
179      with connection.cursor() as cursor:
180
181          sql = "SELECT PRODUIT.NOM, PRODUIT.PRODUIT_ID FROM `PRODUIT` INNER JOIN `CATEGORIES` ON PRODUIT.CATEGORIE_ID = CATEGORIES.ID WHERE CATEGORIES.NOM = %s AND NUTRISCORE >= 3"
182          cursor.execute(sql, (name_categories))
183          result = cursor.fetchall()
184          print(result)
185
186      choice_produit = input("\n Indiquer le numéro du produit que vous souhaitez remplacer : ")
187
188      with connection.cursor() as cursor:
189
190          sql = "SELECT PRODUIT.NOM, PRODUIT.PRODUIT_ID FROM `PRODUIT` INNER JOIN `CATEGORIES` ON PRODUIT.CATEGORIE_ID = CATEGORIES.ID WHERE CATEGORIES.NOM = %s AND NUTRISCORE < 3"
191          cursor.execute(sql, (name_categories))
192          result = cursor.fetchall()
193          print(result)
194
```

Utilisation de la Fonction SELECT en MYSQL

```
250 """MAIN LOOP"""
251
252 continu = 0
253 while continu == 0 :
254     try :
255         mode_terminal = input("1 - Quel aliment souhaitez-vous remplacer ? \n2 - Retrouver mes aliments substitués. \
256         mode_terminal = int(mode_terminal)
257
258         """mode remplacement"""
259         while mode_terminal == 1 :…
311         """mode BDD"""
312
313         while mode_terminal == 2 :
314             print("Voici vos produits : \n ")
315             with connection.cursor() as cursor:
316                 sql = "SELECT * FROM `SUBSTITUT`"
317                 cursor.execute(sql, ())
318                 mes_produits = cursor.fetchall()
319                 print(mes_produits)
320
321             mode_terminal = 0
322
323         """Mode sortie"""
324
325         if mode_terminal == 3 :…
328
329         elif mode_terminal >= 3 :
330             print("{} n'est pas dans les numéros proposés\n".format(mode_terminal))
331
332     except ValueError :…)
```