FIT9136 Assignment 1

1. Get user input function

This function has one positional argument “requirement” which is str type. The variable “requirement” can only be the value of (“letter”, “number”, “letter_or_number_or_underscore”, “email”). According to the “requirement” value, this function asks the user to input a corresponding value and return this input.

- If “requirement” is “letter”, user input can only be letters from [a-zA-Z].
- If “requirement” is “number”, user input can only be [0-9].
- If “requirement” is “letter_or_number_or_underscore”, user input can only be [a-zA-z0-9_].
- If “requirement” is “email”, the user input must contain “@” and “.com”.
- If user input cannot match the “requirement”, a loop should be applied to keep asking user to input until a valid result is obtained. Finally, the valid value should be returned.

For example, when calling this function and giving “requirement” value “letter”, user input like “abc123” will receive an error message printed out. Then, your system should print out messages to ask user re-input until an all letter input is made like “abcde”.

In [1]:
```
"""
name : JUNTAO YU
student ID : 30358809
start date : March 26th 2022
last modified date : March 31st 2022

purpose of this code : collect user information
"""

#define the function
def get_user_input_function(requirement):
    """
    Based on the requirement,verify the user input and return it.

    This function  requires the user to input letters,letter or number
    or underscore,email,and numbers.Then it returns the input.

    Parameters
    ----------
    requirement:str
        requirement that user asks

    Returns
    -------
    str
        required values after verification

    Examples
    --------
    >>>get_user_input_function('letter')
    'abc'
```

```
    """
    #check the requirement
    if requirement == 'letter':
        #have loop to keep asking the input until its valid.
        while True:
            #asking the user to input letter.
            letter = input('you can only type letters\n')   # I added '\n' at the end s
            #checking if the input is valid
            if letter.isalpha():
                #if it is true jump out of the loop and continue to the next step
                break
            else:
                #if it is false then remind the user and keep looping
                print("your input is invalid,you can only type letters\n")
        #return input
        return letter

    #check the requirement
    elif requirement == 'letter_or_number_or_underscore':
        #have a similar while loop in order to keep asking user to input until it is va
        while True:
            #asking user to input letter_or_number_or_underscore
            letter_or_number_or_underscore = input('it can only be letter or number or
            #create a set of validation
            validation = set(("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRST
            # convert  input into set of characters
            verification = set((letter_or_number_or_underscore))
            #check if the input is valid
            if verification.issubset(validation):
                #if it is true then break in order to jump out of the loop and continue
                break
            #if it is false output statement to tell user he typed wrong and stay in th
            else:
                print("your input is invalid\n")
        #return input
        return letter_or_number_or_underscore

    #check the requirement
    elif requirement == 'email':
        #repeating similar loop
        while True:
            #asking user to input email
            email = input('it has to involve @ and .com \n')
            #check if the input is valid
            if '@' in email:
                if'.com' in email:
                    break
                #have to print reminder here otherwise if input only contain "@" it wou
                else:
                    print('your input is invalid')
            else:
                print('your input is invalid')
        #return input
        return email

    #check the requirement
    elif requirement == 'number':
        #have a loop asking for correct input
        while True:
            #asking user to input number
```

```
            number = input('you can only type number here\n')
            #check if there is only numbers input
            if number.isdigit():
                break
            else:
                print("your input is invalid,only number is allowed")
        #return the input
        return number
```

2. Encryption function

This function has a string type positional argument. This function is used to encrypt user input passwords. When we use a web application and enter our password. Our password values will not be stored directly as plain text into the application's database. Because if an attacker get the database information, they can obtain the user password text. Commonly, users' passwords will be encrypted with some algorithms(like MD5) to avoid further loss when database leakage happens. Our function emulates a password encryption process. The final encrypted password will follow the requirements listed below.

One variable all_punctuation is provided whose value is all*punctuation* ="""*!"#$\%&'()*+,-./:;<=>?@[]^`{|}~""".

- get the character of all_punctuation at input string length module all_punctuation length as the first_character.
- The second_character position in all_punctuation is the input string length module 5.
- The third_character position in all_punctuation is the input string length module 10.
- Start character "^^^" and End character "$$$" for the final encrypted string.

Example:

- input string: "password"
- first_character: ")"
- second_character: "$"
- third_character: ")"
- Encrypted result: "^^^)p)$$a$$)))s))))s)$$w$$)))o))))r)$$d$$$$$"
   The encrypted string will be returned at the end of this function.

In [2]:

```
def encryption(input_str):
    """
    encrypt the password.

    Based on the user input,this function add punctuations to
    the input value by following a certain pattern.

    Parameters
    ----------
```

```
    input_str:str
        password gets from user input

    Returns
    -------
    str
        required values after encryption

    Examples
    --------
    >>>encryption('password')
    '^^^)p)$$a$$)))s))))s)$$w$$)))o))))r)$$d$$$$$'
    """
    all_punctuation = """!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
    #get the first character
    first_character = all_punctuation[len(input_str)%len(all_punctuation)]
    #get the second character
    second_character = all_punctuation[len(input_str)%5]
    #get the third character
    third_character = all_punctuation[len(input_str)%10]
    #split password into every 3 letters
    password_list = [input_str[i:i+3] for i in range(0,len(input_str),3)]
    #initialize password
    password = ''
    #use for loop to edit each 3 letters
    for x in password_list:
        #check the index of the list(x)
        if len(list(x)) == 3:
            #convert each element to list of every letter and then concatenate punctuat
            password += first_character+list(x)[0]+first_character+second_character*2+l
        elif len(list(x)) == 2:
            password += first_character+list(x)[0]+first_character+second_character*2+l
        elif len(list(x)) == 1:
            password += first_character+list(x)[0]+first_character
    #add the first and last 3 characters
    password = """^^^""" + password + """$$$"""
    #return the password
    return password
```

3. Generate user id function

This function contains two positional arguments that are number_of_digits(int type), number_list(list type, a list of str). Based on the number_of_digits, you are required to generate an all digit string and all the string in the number_list should be unique.

For example, the number_of_digits = 7, the generated string should only contain 7 digits. If the number_list = [“1234567”, “2345678”], the newly generated id cannot be the same as any element in the given list. The generated string id should be returned.

In [3]:

```
import random

def generate_user_id(number_of_digits, number_list):
    """
    To generate a random number .
```

```
    Based on the number of digit input ,this function generates a random number
    with required digits.And it authenticates generated number's existence within
    the provied list.

    Parameters
    ----------
    number_of_digits : int
        required number

    number_list : str
        provided list to be used for verification

    Returns
    -------
    str
        user id which is randomly generated

    Examples
    --------
    >>>generate_user_id(3, ['100','234'])
    '125'
    """
    #generate a random number with asked digits and convert it to string type.
    user_id = str(random.randint(10**(number_of_digits-1),10**number_of_digits-1))
    #check if the number is in the list
    while user_id in number_list:
        #if it is true then generate another new id
        user_id = str(random.randint(10**(number_of_digits-1),10**number_of_digits-1))
    #return id
    return user_id
```

4. Check username exist function

This function contains two positional arguments that are username(str type) and user_list(list type, a list of list). The user_list looks like [[username1, password1, email1, postcode1],[username2, password2, email2, postcode2]...]. This function should check whether the username string exists in the user_list or not and return the boolean result.

For example, given user_list=[[“aaaaa”, “bbbbb”, “aaa\@gmail.com”, “3000”], [“eeeee”, “fffff”, “eee\@gmai.com”, “4000”]], if the given username is “aaaaa”, return True.

In [4]:

```
def check_username_exist(username,user_list):
    """
    To check the matched value and return boolean.

    Based on the input username and provided user list ,this function
    checks if the user name exists in the provided user list.

    Parameters
    ----------
    username : str
        user input value

    user_list : list
        provided list to be used for verification
```

```
    Returns
    -------
    boolean
        depend on the match of username and user list

    Examples
    --------
    >>>check_username_exist(abc,['abc','123'])
    True
    """
    #set a variable i for usage
    i = 0
    #get through user_list in order to check each list within it
    for info in user_list:
        #check if username exists in info .
        if username in info:
            i += 1
    #check if i >0 to decide the return boolean
    if i > 0:
        return True
    else:
        return False
```

5. Authenticate username and password function

This function contains three positional arguments that are username(str type), password(str type) and user_dict(dict type). The user_dict looks like {user_id1: [username1, password1, email1, postcode1], user_id2: [username2, password2, email2, postcode2].....}. You are required to check whether the given username and password can match one item in the user_dict.

In [5]:

```
"""
For example, the username="aaaa", password="12333",
user_dict={"12345": ["aaaa", "^^^&1&!!2!!&&&3&&&&3&!!3!!$$$", "aa@gmail.com", "3151"],
           "34567": ["bbbbb", "^^^%1%%%2%%%%%2%%%%2%$$$", "bb@gmail.com", "3000"]},
the authentication result will be True.
If the username="bbbbb", password="12333", the authentication result will be False.
"""
# define the function
def authenticate_username_password(username,password,user_dict):
    """
    Authenticate the username and the password.

    Based on the input username and input password ,this function
    checks if these two values matchs in the provided dictionary.

    Parameters
    ----------
    username : str
        user input value

    password : str
        user input value

    user_dict : dict
        provided user dictionary containing user information
```

```
    Returns
    -------
    boolean
        depend on the authentification of username,password and user
        dictionary.

    Examples
    --------
    >>>authenticate_username_password('aaaa','12333',{“12345”: [“aaaa”, “^^^&1&!!2!!&&&
    True
    """
    #initialize a variable to use later
    i = 0
    #access to the key of user_dict
    key = [x for x in user_dict.keys()]
    #access to the element of user id
    for j in key:
        #check if username matchs
        if username == user_dict[j][0]:
            #check if password matchs
            if encryption(password) == user_dict[j][1] :
                i += 1
    #check if i >0 to decide the return boolean
    if i > 0:
        return True
    else:
        return False
```

6. Add user to list function

This function has two positional arguments that are user_id_list(list type) and user_list(list type). The user_id_list looks like [‘1234’, ‘5123’, ‘62345’,.....] and the user_list looks like [[username1, password1, email1, postcode1],[username2, password2, email2, postcode2]...]. In this function, you should call the get user input function several times to ask the user to input username(only contains letters), password(contains letter or number or underscore), email(email format) and postcode(only contains numbers). Username cannot have duplicates in the user_list(call check username exist function here). After getting the postcode, you are required to generate a unique user_id for this user.
The rules are listed below.

- 1000 <= postcode < 2000 → generate a 7 digits user id
- 2000 <= postcode < 3000 --> generate a 8 digits user id
- 3000 <= postcode < 4000 --> generate a 9 digits user id
- 4000 <= postcode < 5000 --> generate a 10 digits user id
   After generating the unique user_id, it should be added into the user_id_list.
  Once getting all the necessary information from user input, a new user(format: [username, password, email, postcode]) should be added to the user_list. The password should be encrypted when adding user info into user_list.

For example, after getting user input, a user like [“aaaaa”, “^^^%1%%%2%%%%%2%%%%2%$$$”, “aa\@gmail.com”, “3131”] can be added into the user_list and a user id “123456789” can be added into the user_id_list.

In [6]:

```
def add_user_to_list(user_id_list,user_list):
    """
    add provided informations to the list.

    Based on the input username ,input password ,input email,
    input postcode ,this function will call other functions in order
    to implement authentification and many functionalities and then
    add valid informations to the list.

    Parameters
    ----------
    user_id_list : list
        the list contains user id

    user_list : list
        a list contains user informations

    Returns
    -------
    None

    Examples
    --------
    >>>add_user_to_list(user_id_list,user_list)
     [‘1234’, ‘5123’, ‘62345’][[username1, password1, email1, postcode1],[username2, pa
    """
    #output to tell user to type the name
    print("please enter your username\n")
    #call the function to verify the input
    username = get_user_input_function('letter')
    #check if username already exist
    while check_username_exist(username,user_list):
        #output to tell user to type another name
        print("user name exists,please try another username\n")
        #call the function to verify the input
        username = get_user_input_function('letter')
        #check if the name exist
        if not check_username_exist(username,user_list):
            break
    #output to tell user to type the password
    print("please enter your password\n")
    #call the function to verify the input
    password = get_user_input_function('letter_or_number_or_underscore')
    #encrypt the password
    password = encryption(password)
    #output to tell user to type the email
    print("please enter your email\n")
    #call the function to verify the input
    email = get_user_input_function('email')
    #output to tell user to type the email
    print("please enter your postcode\n")
    #call the function to verify the input
    postcode = get_user_input_function('number')
    #check post code and initialise how much digit is the id
```

```
    if 1000 <= int(postcode) < 2000:
        number_of_digits = 7
    elif 2000 <= int(postcode) < 3000:
        number_of_digits = 8
    elif 3000 <= int(postcode) < 4000:
        number_of_digits = 9
    elif 4000 <= int(postcode) < 5000:
        number_of_digits = 10
    else:
        number_of_digits = 11
    #generate user id based on the postcode
    user_id = generate_user_id(number_of_digits, user_id_list)
    #add it to the user id list
    user_id_list.append(user_id)
    #add the user input to the user list
    user_list.append([username,password,email,postcode])
    return None
```

7. Test function

This function contains the test code using previous defined functions. The test function steps are listed below. You can also add more steps if you need.

1. Define a user id list.
2. Define a user list. Each user is also a list which contains username(str type), encrypted password(str type), email(str type) and postcode(str type). The format is like [[username1, password1, email1, postcode1],[username2, password2, email2, postcode2]...].
3. Add several users by calling add user to list function.
4. Convert the user id list and user list to a dictionary.
5. Call the authentication of username and password function.
6. When a user enters “q”, the program can quit. Otherwise, keep asking the user to input and do authentication.
7. Print out “username password correct” or “username or password incorrect” according to the authentication result.

In [7]:
```
# define user id list
user_id_list= []
# define user list
user_list = []
#add users
add_user_to_list(user_id_list,user_list)
add_user_to_list(user_id_list,user_list)
add_user_to_list(user_id_list,user_list)
```

```
please enter your username

you can only type letters
yujuntao
please enter your password

it can only be letter or number or underscore
123456
please enter your email
```

```
it has to involve @ and .com
asdf@df.com
please enter your postcode

you can only type number here
234234
please enter your username

you can only type letters
sfsdfs
please enter your password

it can only be letter or number or underscore
asfdf
please enter your email

it has to involve @ and .com
sdfdf@vcvc.com
please enter your postcode

you can only type number here
54543
please enter your username

you can only type letters
sdfsdf
please enter your password

it can only be letter or number or underscore
sdfdsf
please enter your email

it has to involve @ and .com
gfdgf@dfdf.com
please enter your postcode

you can only type number here
45356
```

In [8]:

```python
#create a dictionary
user_dict = {user_id_list[k] : user_list[k] for k in range(len(user_id_list))}
```

In [9]:

```python
#tell the user how to quit
print('type \"q\" and press enter to quit\n')
#have a loop to ask user to input
while True:
    #ask user to input the username and the password
    username = input('type your username\n')
    #add quit
    if username == 'q':
        break
    password = input('type your password\n')
    #add quit
    if username == 'q':
        break
    #call the authenticate function  and tell user the result
    if authenticate_username_password(username,password,user_dict):
```

```
        print('username password correct')
    else:
        print('username or password incorrect')
```

```
type "q" and press enter to quit

type your username
yujuntao
type your password
123456
username password correct
type your username
sdfdf
type your password
bfb4
username or password incorrect
type your username
q
```