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 [ ]:
```
"""
Name: Shixin Huang
ID:31029248
Start date: 20220315
Last modified date:20220331
"""
```

In [2]:
```
#Get user input function
def input_req(requirement,user_input):
    """
    According to the “requirement” value input, asks the user
    to input a corresponding value and return this input.
    If user input cannot match the “requirement”,
    keep asking user to input until a valid result is obtained.
    Finally, the valid value should be returned.

    Parameters
    ----------
    requirement : str
        “letter”, “number”, “letter_or_number_or_underscore”, “email”

    user_input : str

    Returns
    -------------
    str
        user_input

    Examples
    --------------------
   plz type:
   >>>email
    plz letter:
    >>>ahsung@qqq.com
    ahsung@qqq.com

    plz type:
    >>>number
    plz letter:
    >>>ahsulng111
```

```
    invaild number  plz check
    plz number only be [0-9]:
    >>>11112222
    1112222
    """
    #If “requirement” is “letter”, user input can only be letters from [a-zA-Z].
    if requirement == 'letter':
        while True:
            if user_input.isalpha() == True:   #user input can only be letters
                break
            else:
                print(" invaild",requirement," plz check")
                user_input =input("only be letters from [a-zA-Z]:")


    #If “requirement” is “number”, user input can only be [0-9].
    elif (requirement == 'number'):
        while True:
            if user_input.isdigit() == True:   #user input can only be number
                break
            else:
                print(" invaild",requirement," plz check")
                user_input =input("plz number only be [0-9]:")


    #If “requirement” is “letter_or_number_or_underscore”,
    #user input can only be [a-zA-z0-9_].
    elif (requirement == 'letter_or_number_or_underscore'):
        while True:
            if user_input.replace('_','').isalnum() == True:
                #remove "_" and check user input can only be [a-zA-z0-9_]
                break
            else:
                print(" invaild",requirement," plz check")
                user_input =input("can only be [a-zA-z0-9_ :")


    #If “requirement” is “email”,
    #the user input must contain “@” and “.com”.
    elif (requirement == 'email'):
        while True:
            if user_input.find('@')!= -1 and user_input.find('.com')!= -1:
                #user input must contain  “@” and “.com”
                break
            else:
                print(" invaild",requirement," plz check")
                user_input =input("must contain “@” and “.com” :")
    return user_input#return user_input to check match
```

In [3]:

```
#test input_req function

requirement = input("plz type:")
user_letter = input("plz letter:")
input_req(requirement,user_letter)
```

```
plz type:letter
plz letter:aaaa
'aaaa'
```

Out[3]:

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 [4]:

```
#Encryption function
def encryption(input_str):
    all_punctuation = """!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
    # your answer
    """
    This function is used to encrypt user input passwords.
    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.

    Parameter
    -----------------
    input_str : str

    Returns
    -----------------
    str
        encrypted_result

    Examples
    --------------------
    plz input password:
    >>>password
    plz letter:
    '^^^)p)$$a$$)))s))))s)$$w$$)))o))))r)$$d$$$$$'

    """

    encrypted_result = '^^^'#Start character "^^^"

    for i in range(len(input_str)):#loop to select character position
```

```
        if i%3 == 0:
            #character of all_punctuation at
            #input string length module all_punctuation length as the first_character.
            character_position = len(input_str) % len(all_punctuation)
            character = all_punctuation[character_position]
            encrypted_result = encrypted_result + character + input_str[i] + character

        elif i%3 == 1:
            #The second_character position in all_punctuation
            #is the input string length module 5.
            character_position = len(input_str) % 5
            character = all_punctuation[character_position]
            encrypted_result = encrypted_result + character*2 + input_str[i] + character*2

        else:
            #The third_character position in all_punctuation is the
            #input string length module 10.
            character_position = len(input_str) % 10
            character = all_punctuation[character_position]
            encrypted_result = encrypted_result + character*3 + input_str[i] + character*3

    encrypted_result = encrypted_result + '$$$'# End character "$$$"
    return encrypted_result
```

In [5]:

```
#test Encryption function
pw = input("plz input password: ")
encryption(pw)
```

Out[5]:

```
plz input password: password
'^^^)p)$$a$$)))s))))s)$$w$$)))o))))r)$$d$$$$$'
```

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 [6]:

```
# Generate user id function
import random
def uid_generate(number_of_digits,number_list):
    """
    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

    Parameter
    ----------------
    number_of_digits : int

    number_of_digits : list

    Returns
    ----------------
    str
        num_str

    Examples
    --------------------
   generated string digits:
    >>>7
```

```
    '4549454'

    """

    q = number_of_digits
    #user input decide digits of generate number
    number = random.choices(range(10),k=q)
    #generate random user id

    for i in range(len(number)):
        number[i] = str(number[i])
        #change generate number into string type
    num_str = ''.join(number)
    #put generate number into list type

    if num_str  in number_list:
        #check whether  number same with given list
        for i in range(len(number)):
            number[i] = str(number[i])
        num_str = ''.join(number)
        #new generated string id
    else:
        num_str

    return num_str
```

In [7]:

```
#test Generate user id function
number_list = ["1234567", "2345678"]
number = input("generated string digits: ")

n = int(number)
uid_generate(n,number_list)
```

Out[7]:

```
generated string digits: 7
'0078993'
```

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 [9]:

```
# Check username exist function
def check_username(username,user_list):
    """
    This function should check whether the username string exists
    in the user_list or not and return the boolean result

    Parameter
    -----------------
    username : str

    user_list : list

    Returns
    -----------------
    boolean
        username in username_list
```

```
    Examples
    ----------
    >>>old_list = [["a", 2, 3, 4], ["b", 5, 6, 8], ["c", 5, 6, 8], ["d", 5, 6, 8]]
    gcheck username:
    >>>d
    True
    """
    username_list = []

    for row in user_list:
        username_list.append(row[0])#select all user name from user_list

    return username in username_list
```

In [10]:

```
#test Check username exist function
old_list = [["a", 2, 3, 4], ["b", 5, 6, 8], ["c", 5, 6, 8], ["d", 5, 6, 8]]
name = input(" check username:")
check_username(name,old_list)
```

Out[10]:

```
 check username:a
True
```

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 [ ]:

```
"""
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 your function here
```

In [11]:

```
# Authenticate username and password function
def authenticate(username,password,user_dict):
    """
    check whether the given username and password
    can match one item in the user_dict.

    Parameter
    ----------
    username : str

    user_list : list

    user_dict: dict

    Returns
    ----------
    boolean

    Examples
    ----------
    >>>u_dict={"12345": ["aaaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
           "34567": ["bbbbb","^^^%1%%%2%%%%%2%%%%2%$$$"]}
    account name:
    >>>aaaa
```

```
    account password:
    >>>12333
    True

    account name:
    >>>bbb
    account password:
    >>>1225
    False
    """
    pw = encryption(password)#encrypt user input passwords
    user_account = {}

    for uid in user_dict.values():
        user_account[uid[0]] = uid[1]
        #select username and password belong from user_dict
        #into  dict type user_account

    if username in user_account:
        if user_account[username] == pw:
            #check username and password can match
            return True
        else:
            return False
    else:
        return False
```

In [12]:

```
#test Authenticate username and password function
u_dict={"12345": ["aaaa","^^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
           "34567": ["bbbbb","^^^^%1%%%2%%%%%2%%%%2%$$$"]}
uname = input("account name: ")
pword = input("account password: ")
authenticate(uname,pword,u_dict)
```

```
account name: aaaa
account password: 12333
True
```

Out[12]:

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 [36]:

```
# Add user to list function
def add_new_user(user_id_list,user_list):
    """
    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.
    postcode  to generate a unique user_id for this user.
    new user_id, it should be added into the user_id_list
    new user(format: [username, password, email, postcode])
    should be added to the user_list.

    Parameter
    -----------------
    user_id_list : list

    user_list : list

    Returns
    -----------------


    Examples
    --------------------
    >>>usera_list = [["aaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
    ["bbb", "^^^%1%%%2%%%%%2%%%%2%$$$"]]
    >>>usera_idlist = ["1234567", "2345678"]
    plz input username:
    >>>ahsu
    plz input password:
    >>>23a7dhs
    plz input email:
    >>>asu@hah.com
    plz input postcode:
    >>>2322
    [['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$'], ['bbb', '^^^%1%%%2%%%%%2%%%%2%$$$'],
    ['ahsu', '^^^(2(##3##(((a((((7(##d##(((h((((s($$$', 'asu@hah.com', 2322]]
    ['1234567', '2345678', '39859095']
    """

    # user_name input and check unique
    user_name = input("plz input username: ")
    user_name = input_req('letter',user_name)
    while check_username(user_name,user_list) == True:
        user_name = input("username existed!: ")
        user_name = input_req('letter',user_name)

    # user_password input check and encryption
    user_password = input("plz input password: ")
    user_password = input_req('letter_or_number_or_underscore',user_password)
    user_password = encryption(user_password)

    # user_email input check
    user_email = input("plz input email: ")
    user_email = input_req('email',user_email)

    # user_postcode input check
    user_postcode = input("plz input postcode: ")
    user_postcode = input_req('number',user_postcode)
    user_postcode = int(user_postcode)

    #check user_postcode vaild
```

```
    while(user_postcode<1000 or user_postcode>5000):
        user_postcode = input("plz check postcode: ")
        user_postcode = input_req('number',user_postcode)
        user_postcode = int(user_postcode)

    #generate a 7 digits unique user id
    if (1000 <= user_postcode < 2000):
        user_id = uid_generate(7,user_id_list)
        user_id_list.append(user_id)
    #generate a 8 digits unique user id
    elif(2000 <= user_postcode < 3000):
        user_id = uid_generate(8,user_id_list)
        user_id_list.append(user_id)
    #generate a 9 digits unique user id
    elif(3000 <= user_postcode < 4000):
        user_id = uid_generate(9,user_id_list)
        user_id_list.append(user_id)
    #generate a 10 digits unique user id
    elif(4000 <= user_postcode < 5000):
        user_id = uid_generate(10,user_id_list)
        user_id_list.append(user_id)

    #user_id and new user into list input
    user_new_list = [user_name,user_password,user_email,user_postcode]
    user_list.append(user_new_list)
```

In [37]:

```
#test Add user to list function
usera_list = [["aaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"], ["bbb", "^^^%1%%%2%%%%%2%%%%2%$$$"]]
usera_idlist = ["1234567", "2345678"]
add_new_user(usera_idlist,usera_list)
print(usera_list)
print(usera_idlist)
```

```
plz input username: asga
plz input password: 13fsa
plz input email: has@aha.com
plz input postcode: 3213
[['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$'], ['bbb', '^^^%1%%%2%%%%%2%%%%2%$$$'], ['asga', '^^^&1
&!!3!!&&&f&&&&s&!!a!!$$$', 'has@aha.com', 3213]]
['1234567', '2345678', '477875346']
```

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 [75]:

```
#Convert the user id list and user list to a dictionary.
def list_convert_dic(user_id_list,user_list,user_dict):
    """
```

```
    Convert the user id list and user list to a dictionary.

    Parameter
    -----------------
    user_id_list : list
    user_list : list
    user_dict : dic

    Returns
    -----------------

    Examples
    ---------------------

    """
    for i in range(len(user_list)):
        #select correlate in user_id and user_list
        a=user_id_list[i]
        b=user_list[i]
        user_dict.update({a:b})
```

In [76]:

```
#test Convert list to dictionary function
usera_idlist = ["1234567", "2345678"]
usera_list = [["aaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
              ["bbb", "^^^%1%%%2%%%%%2%%%%2%$$$"]]
user_dict = {}

list_convert_dic(usera_idlist,usera_list,user_dict)
print(user_dict)
```

```
{'1234567': ['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$'], '2345678': ['bbb', '^^^%1%%%2%%%%%2%%%%2%$$$']}
```

In [77]:

```
#do authentication until enters “q”, the program can quit.
def user_authentication(user_dict):
    """
    do authentication until enters “q”, the program can quit

    Parameter
    -----------------
    user_dict : dic

    Returns
    -----------------
    str : "program quit"

    Examples
    ---------------------
    >>>usera_dict = {'1234567': ['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$']}
    account name:
    >>>aaa
    account password:
    >>>12333
    username password correct
    account name:
    >>>q
    program quit

    """
    uname = input("account name: ")
    #if name or password enters “q”, the program can quit
    while uname != 'q':
        pword = input("account password: ")
        if pword == 'q':
            break

        #check username password correct?
```

```
        if authenticate(uname,pword,user_dict)== True:
            print("username password correct")
        else:
            print("username or password incorrect")
        uname = input("account name: ")
    print("program quit")
```

In [78]:

```
#test user_authentication function
usera_dict = {'1234567': ['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$'],
              '2345678': ['bbb', '^^^%1%%%2%%%%%2%%%%2%$$$']}
user_authentication(usera_dict)
```

```
account name: aaa
account password: 12333
username password correct
account name: q
program quit
```

In [73]:

```
#input check and add user then authentication
def account_management(user_id_list,user_list,user_dict):
    """
    input check and add user then authentication

    Parameter
    ----------------
    user_id_list : list
    user_list : list
    user_dict : dic

    Returns
    ----------------
    str : "program quit"

    Examples
    --------------------

    """
    print("Add user information and check")
    print("==============================")

    add_new_user(user_id_list,user_list)
    #check continue Add users?
    add_con = input("add more user(y to continue)? ")
    while add_con == 'y':
        add_new_user(user_id_list,user_list)
        add_con = input("add more user(y to continue)? ")

    #convert user_id_list,user_list into dic type
    list_convert_dic(user_id_list,user_list,user_dict)

    print("authentication of username and password")
    print("================================")

    #check username password correct?
    user_authentication(user_dict)
```

8. Code encapsulation

Here is the my defined function combined for the question above.

First code block is the function part.

Second code block is the test function part.

In [82]:

```
# Get user input function
```

```
def input_req(requirement, user_input):
    """
    According to the “requirement” value input, asks the user
    to input a corresponding value and return this input.
    If user input cannot match the “requirement”,
    keep asking user to input until a valid result is obtained.
    Finally, the valid value should be returned.

    Parameters
    ----------
    requirement : str
        “letter”, “number”, “letter_or_number_or_underscore”, “email”

    user_input : str

    Returns
    -------------
    str
        user_input

    Examples
    ---------------------
   plz type:
   >>>email
    plz letter:
    >>>ahsung@qqq.com
    ahsung@qqq.com

    plz type:
    >>>number
    plz letter:
    >>>ahsu1ng111
    invaild number  plz check
    plz number only be [0-9]:
    >>>11112222
    1112222
    """
    # If “requirement” is “letter”, user input can only be letters from [a-zA-Z].
    if requirement == 'letter':
        while True:
            if user_input.isalpha() == True:  # user input can only be letters
                break
            else:
                print(" invaild", requirement, " plz check")
                user_input = input("only be letters from [a-zA-Z]:")


    # If “requirement” is “number”, user input can only be [0-9].
    elif (requirement == 'number'):
        while True:
            if user_input.isdigit() == True:  # user input can only be number
                break
            else:
                print(" invaild", requirement, " plz check")
                user_input = input("plz number only be [0-9]:")


    # If “requirement” is “letter_or_number_or_underscore”,
    # user input can only be [a-zA-z0-9_].
    elif (requirement == 'letter_or_number_or_underscore'):
        while True:
            if user_input.replace('_', '').isalnum() == True:
                # remove "_" and check user input can only be [a-zA-z0-9_]
                break
            else:
                print(" invaild", requirement, " plz check")
                user_input = input("can only be [a-zA-z0-9_ :")
```

```
    # If “requirement” is “email”,
    # the user input must contain “@” and “.com”.
    elif (requirement == 'email'):
        while True:
            if user_input.find('@') != -1 and user_input.find('.com') != -1:
                # user input must contain  “@” and “.com”
                break
            else:
                print(" invaild", requirement, " plz check")
                user_input = input("must contain “@” and “.com” :")
    return user_input  # return user_input to check match


# Encryption function
def encryption(input_str):
    all_punctuation = """!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
    # your answer
    """
    This function is used to encrypt user input passwords.
    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.

    Parameter
    ----------------
    input_str : str

    Returns
    -----------------
    str
        encrypted_result

    Examples
    --------------------
    plz input password:
    >>>password
    plz letter:
    '^^^)p)$$a$$)))s))))s)$$w$$)))o))))r)$$d$$$$$'

    """

    encrypted_result = '^^^'  # Start character “^^^”

    for i in range(len(input_str)):  # loop to select character position
        if i % 3 == 0:
            # character of all_punctuation at
            # input string length module all_punctuation length as the first_character.
            character_position = len(input_str) % len(all_punctuation)
            character = all_punctuation[character_position]
            encrypted_result = encrypted_result + character + input_str[i] + character

        elif i % 3 == 1:
            # The second_character position in all_punctuation
            # is the input string length module 5.
            character_position = len(input_str) % 5
            character = all_punctuation[character_position]
            encrypted_result = encrypted_result + character * 2 + input_str[i] + character * 2

        else:
            # The third_character position in all_punctuation is the
            # input string length module 10.
            character_position = len(input_str) % 10
            character = all_punctuation[character_position]
            encrypted_result = encrypted_result + character * 3 + input_str[i] + character * 3
```

```
    encrypted_result = encrypted_result + '$$$'  # End character "$$$"
    return encrypted_result


# Generate user id function
import random


def uid_generate(number_of_digits, number_list):
    """
    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

    Parameter
    -----------------
    number_of_digits : int

    number_of_digits : list

    Returns
    -----------------
    str
        num_str

    Examples
    ---------------------
   generated string digits:
    >>>7
   '4549454'

    """

    q = number_of_digits
    # user input decide digits of generate number
    number = random.choices(range(10), k=q)
    # generate random user id

    for i in range(len(number)):
        number[i] = str(number[i])
        # change generate number into string type
    num_str = ''.join(number)
    # put generate number into list type

    if num_str in number_list:
        # check whether  number same with given list
        for i in range(len(number)):
            number[i] = str(number[i])
        num_str = ''.join(number)
        # new generated string id
    else:
        num_str

    return num_str


# Check username exist function
def check_username(username, user_list):
    """
    This function should check whether the username string exists
    in the user_list or not and return the boolean result

    Parameter
    -----------------
    username : str

    user_list : list

    Returns
```

```
    ----------------
    boolean
        username in username_list

    Examples
    --------------------
    >>>old_list = [["a", 2, 3, 4], ["b", 5, 6, 8], ["c", 5, 6, 8], ["d", 5, 6, 8]]
    gcheck username:
    >>>d
    True
    """
    username_list = []

    for row in user_list:
        username_list.append(row[0])  # select all user name from user_list

    return username in username_list


# Authenticate username and password function
def authenticate(username, password, user_dict):
    """
    check whether the given username and password
    can match one item in the user_dict.

    Parameter
    ----------------
    username : str

    user_list : list

    user_dict: dict

    Returns
    ----------------
    boolean

    Examples
    --------------------
    >>>u_dict={"12345": ["aaaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
           "34567": ["bbbbb","^^^%1%%%2%%%%%2%%%%2%$$$"]}
    account name:
    >>>aaaa
    account password:
    >>>12333
    True

    account name:
    >>>bbb
    account password:
    >>>1225
    False
    """
    pw = encryption(password)  # encrypt user input passwords
    user_account = {}

    for uid in user_dict.values():
        user_account[uid[0]] = uid[1]
        # select username and password belong from user_dict
        # into  dict type user_account

    if username in user_account:
        if user_account[username] == pw:
            # check username and password can match
            return True
        else:
            return False
    else:
        return False
```

```
# Add user to list function
def add_new_user(user_id_list, user_list):
    """
    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.
    postcode  to generate a unique user_id for this user.
    new user_id, it should be added into the user_id_list
    new user(format: [username, password, email, postcode])
    should be added to the user_list.

    Parameter
    -----------------
    user_id_list : list

    user_list : list

    Returns
    -----------------


    Examples
    ---------------------
    >>>usera_list = [["aaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
    ["bbb", "^^^%1%%%2%%%%%2%%%%2%$$$"]]
    >>>usera_idlist = ["1234567", "2345678"]
    plz input username:
    >>>ahsu
    plz input password:
    >>>23a7dhs
    plz input email:
    >>>asu@hah.com
    plz input postcode:
    >>>2322
    [['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$'], ['bbb', '^^^%1%%%2%%%%%2%%%%2%$$$'],
    ['ahsu', '^^^(2(##3##(((a((((7(##d##(((h((((s($$$', 'asu@hah.com', 2322]]
    ['1234567', '2345678', '39859095']
    """

    # user_name input and check unique
    user_name = input("plz input username: ")
    user_name = input_req('letter', user_name)
    while check_username(user_name, user_list) == True:
        user_name = input("username existed!: ")
        user_name = input_req('letter', user_name)

    # user_password input check and encryption
    user_password = input("plz input password: ")
    user_password = input_req('letter_or_number_or_underscore', user_password)
    user_password = encryption(user_password)

    # user_email input check
    user_email = input("plz input email: ")
    user_email = input_req('email', user_email)

    # user_postcode input check
    user_postcode = input("plz input postcode: ")
    user_postcode = input_req('number', user_postcode)
    user_postcode = int(user_postcode)

    # check user_postcode vaild
    while (user_postcode < 1000 or user_postcode > 5000):
        user_postcode = input("plz check postcode: ")
        user_postcode = input_req('number', user_postcode)
        user_postcode = int(user_postcode)
```

```
    # generate a 7 digits unique user id
    if (1000 <= user_postcode < 2000):
        user_id = uid_generate(7, user_id_list)
        user_id_list.append(user_id)
    # generate a 8 digits unique user id
    elif (2000 <= user_postcode < 3000):
        user_id = uid_generate(8, user_id_list)
        user_id_list.append(user_id)
    # generate a 9 digits unique user id
    elif (3000 <= user_postcode < 4000):
        user_id = uid_generate(9, user_id_list)
        user_id_list.append(user_id)
    # generate a 10 digits unique user id
    elif (4000 <= user_postcode < 5000):
        user_id = uid_generate(10, user_id_list)
        user_id_list.append(user_id)

    # user_id and new user into list input
    user_new_list = [user_name, user_password, user_email, user_postcode]
    user_list.append(user_new_list)


# Convert the user id list and user list to a dictionary.
def list_convert_dic(user_id_list, user_list, user_dict):
    """
    Convert the user id list and user list to a dictionary.

    Parameter
    -----------------
    user_id_list : list
    user_list : list
    user_dict : dic

    Returns
    -----------------

    Examples
    ----------------------

    """
    for i in range(len(user_list)):
        # select correlate in user_id and user_list
        a = user_id_list[i]
        b = user_list[i]
        user_dict.update({a: b})


# do authentication until enters “q”, the program can quit.
def user_authentication(user_dict):
    """
    do authentication until enters “q”, the program can quit

    Parameter
    -----------------
    user_dict : dic

    Returns
    -----------------
    str : "program quit"

    Examples
    --------------------
    >>>usera_dict = {'1234567': ['aaa', '^^^&1&!!2!!&&&3&&&&3&!!3!!$$$']}
    account name:
    >>>aaa
    account password:
    >>>12333
    username password correct
    account name:
```

```
    >>>q
    program quit

    """
    uname = input("account name: ")
    # if name or password enters “q”, the program can quit
    while uname != 'q':
        pword = input("account password: ")
        if pword == 'q':
            break

        # check username password correct?
        if authenticate(uname, pword, user_dict) == True:
            print("username password correct")
        else:
            print("username or password incorrect")
        uname = input("account name: ")
    print("program quit")


# input check and add user then authentication
def account_management(user_id_list, user_list, user_dict):
    """
    input check and add user then authentication

    Parameter
    ----------------
    user_id_list : list
    user_list : list
    user_dict : dic

    Returns
    ----------------
    str : "program quit"

    Examples
    --------------------

    """
    print("Add user information and check")
    print("==============================")

    add_new_user(user_id_list, user_list)
    # check continue Add users?
    add_con = input("add more user(y to continue)? ")
    while add_con == 'y':
        add_new_user(user_id_list, user_list)
        add_con = input("add more user(y to continue)? ")

    # convert user_id_list,user_list into dic type
    list_convert_dic(user_id_list, user_list, user_dict)

    print("authentication of username and password")
    print("==================================")

    # check username password correct?
    user_authentication(user_dict)
```

In [83]:

```
# run the test function here
usera_id_list = ["1234567", "2345678"]
usera_list = [["aaa","^^^&1&!!2!!&&&3&&&&3&!!3!!$$$"],
              ["bbb", "^^^%1%%%2%%%%%2%%%%2%$$$"]]
usera_dict = {}
account_management(usera_id_list,usera_list,usera_dict)
```

```
Add user information and check
==============================
```

```
plz input username: aaa
username existed!: ccc
plz input password: @@@
 invaild letter_or_number_or_underscore  plz check
can only be [a-zA-z0-9_ :12345
plz input email: aaa
 invaild email  plz check
must contain “@” and “.com”:ccc@ccc.com
plz input postcode: 9999
plz check postcode: 5432
plz check postcode: 3422
add more user(y to continue)? y
plz input username: ddd
plz input password: 234566
plz input email: ddd@ddd.com
plz input postcode: 3244
add more user(y to continue)? n
authentication of username and password
==================================
account name: ccc
account password: 12345
username password correct
account name: ddd
account password: 23456
username or password incorrect
account name: q
program quit
```

In [ ]: