Time Required: 45 minutes

TASK 1:

You are working for a company as a Junior Software Developer and you have been tasked to build add new functionality to a new dating site the company is building to check for years of a user before allowing them to create an account.

NOTE: The minimum age of a user should be 18 and the maximum age should be 150 years

1. Create a function named ***get_user_age*** that takes "*date_of_birth" a* string as an argument. The format of the date of birth should be "DDDD-MMMM-YYYY", day, month and year. The function returns an integer which represents the age of the user
2. You should convert the date of birth string we get from the function, to a datetime object.
3. Declare variable named *"num_of_days"* that holds the value current date minus the date_of_birth.
4. Divide the *num_of_days* by (*365.2425* - number of days in a year) and this should be the return value of the function.
5. Create another function that takes "age_of_user" an integer as an argument to function ***age_checker.***
6. ***age_checker*** function does the checks below:
    - if the age is less than 18, print "\nYou need to be 18 years and above to create an account.\n" and return 0
    - if the age is greater than 150, print "\nYou need to be 150 years or younger to create an account.\n" and return 0
    - else, print "\nYou are {age_of_user} years old, you can create an account\n and return 1
7. create a function called "**get_user_info()**" takes zero parameters and prompts a user for the "first name, last name, username and email" and prints this values using a f string.
8. Great, now we have 3 functions, one gives us the age of the user, the other validates the age and one gets the user info. Now let's add a way to prompt a user for their date of birth.
9. Declare a variable named attempts that holds the value 5, this will hold the number of attempts the user should try to enter their date of birth.
10. Write a while loop the checks if the attempts are greater than 0
11. Inside the while loop declare a variable that uses the input function to get the birthdate in the terminal then declare another variable named **age** that holds the age by invoking "**get_user_age("date of birth string we get from prompting user")** ".
12. Now use the value of the age variable which is the age of the user in an if statetement to check, If the value "***age_checker(age of user)"*** equals to 1 invoke "**get_user_info()**" and break else decrement attempts and continue prompting until attempts equals to zero.

Examples:

Input: 01-05-2023

Output: You need to be 18 years and above to create an account.

Input: 02-02-2004

Output: You are 19 years old, you can create an account.

Output: Prompts from get_user_info()

Input: 02-10-1850

Output: You need to be 150 years or younger to create an account.

Task 2:

The senior developer is impressed with your work and would like you to implement a simple newsletter service. The newsletters are sent every 5 days to customer emails.

Prerequisites: **pip install pytz** package in order to work well with timezones.

1. Declare a variable customer_emails equal ['test1@test.com', 'test2@test.com', 'test3@test.com', 'test4@test.com', 'test5@test.com', 'test6@test.com', 'test7@test.com', 'test8@test.com', 'test9@test.com', 'test10@test.com', 'test11@test.com', 'test12@test.com', 'test13@test.com', 'test14@test.com', 'test15@test.com', 'test16@test.com', 'test17@test.com', 'test18@test.com', 'test19@test.com', 'test20@test.com']
2. Declare another variable send_date to hold the value of the datetime object for the date and time to send the newsletters. The date needs to be timezone aware and use **CET timezone**
3. Create a function named send_newsletters that takes the customers_emails as an argument.
4. Using a for loop inside the function loop through the customers_emails list and print "email sent to user with username {user_name} with email {*email*}", user_name is the first value in the list returned after splitting the the email string by @.

5. Create another function called **run_task** that takes the current_date and send_date as arguments. Both are datetime objects and are timezone aware.

6. Inside the function check if the current_date equals to the send_date and then invoke the **send_newsletters** function, update the send_date to a date 5 days in the future and print send_date else just print "Newsletters will be sent later" and send_date without being updated.

Examples:

Input:

run_task(current_date=2023-05-07 21:24:30.319133+02:00, send_date=2023-05-07 21:24:30.319133+02:00)

Output:

email sent to user with username test1 with email test1@test.com

email sent to user with username test2 with email test2@test.com

email sent to user with username test3 with email test3@test.com

……...

2023-05-07 21:24:30.319133+02:00

Input:

run_task(current_date=2023-05-07 21:24:30.319133+02:00, send_date=2023-05-08 21:24:30.319133+02:00)

Output:

Newsletters will be sent later

2023-05-08 21:24:30.319133+02:00