|  |  |
| --- | --- |
| New York College | NYC Educational Group | University of Bolton | Tethys |

**COM4302   
Computer Science Fundamentals**

**Logbook**

**Student Name**Angelos Charitos

**Student ID**2436006

# Week 2

**This code:**

Gets three numbers, checks whether to count up or down, and prints the sequence. The end=", " in the print statement prevents each number from being on a new line. The final print() adds a newline after the sequence.

The code:

a = int(input("Enter the starting point (a): "))

b = int(input("Enter the end point (b): "))

c = int(input("Enter the step (c): "))

if a > b:

while a >= b:

print(a, end=", ")

a -= c

else:

while a <= b:

print(a, end=", ")

a += c

print()

Errors faced during this code:

I didn't encounter any significant errors during development, thanks to the simplicity of the task and the clear logic used. The most important aspect of the solution was ensuring the correct handling of ascending and descending cases, which was accomplished simply and reliably with the if/else structure.

Week3

This code:

first creates a list named numbers containing 20 random integers between 1 and 100. It then iterates through the list, checking each number for evenness using the modulo operator (%). If the remainder when dividing by 2 is 0, the number is even, and the even\_count is incremented. Finally, it prints both the list of random numbers and the count of even numbers.

The code:

import random

numbers = [random.randint(1, 100) for \_ in range(20)]

print("List of random numbers:", numbers)

even\_count = 0

for number in numbers:

if number % 2 == 0:

even\_count += 1

print("Number of even numbers:", even\_count)

My solution to counting even numbers in a list of random numbers was very straightforward and didn't require complex design or extensive planning.

This code was simple, and I didn't encounter any significant errors during development.

Week6

This code:

uses separate lists for each character type. The make\_password function builds the password character by character. It randomly chooses a character type, then randomly selects a character from that type's list. It's less concise but easier to understand for someone new to programming. The use of elif and else simplifies the decision-making process.

The code:

import random

lowercase = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

uppercase = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

symbols = ['!', '@', '#', '$', '%', '^', '&', '\*', '(', ')']

def make\_password():

password = ""

for i in range(10):

which\_list = random.randint(1,4)

if which\_list == 1:

password = password + random.choice(lowercase)

elif which\_list == 2:

password = password + random.choice(uppercase)

elif which\_list == 3:

password = password + random.choice(numbers)

else:

password = password + random.choice(symbols)

return password

print("Your password is:", make\_password())

My solution for generating random passwords, presented in a simpler, more explicit style, is straightforward. It didn't require complex design diagrams or formal pseudocode.

The program makes a password by picking letters and numbers randomly. It has lists for lowercase letters, uppercase letters, numbers, and symbols. It goes through a loop ten times. Each time, it randomly chooses one of the lists and then picks a random thing from that list to add to the password. Finally, it shows you the password it has created for you.

Week7:

This code(1):

first defines a function count\_word that takes a filename and the word to search for as input. It reads the file contents into a string, converts both the file contents and the search word to lowercase (to handle case-insensitive searching), and then uses the count() method to count the occurrences of the word in the string. It includes error handling for the case where the file doesn't exist. The main part of the script gets the word from the user and calls the count\_word function to get the count, then prints the result. Remember to create a story.txt file in the same directory as your Python script.

The code:

def count\_word(filename="story.txt", word\_to\_find=""):

"""Counts occurrences of a word in a text file."""

try:

with open(filename, "r") as f:

contents = f.read().lower()

word\_count = contents.count(word\_to\_find.lower())

return word\_count

except FileNotFoundError:

return "Error: File not found."

word = input("Enter the word to search for: ")

count = count\_word(word\_to\_find=word)

print(f"The word '{word}' appears {count} times in the file.")

This code counts how many times a word shows up in a story file. You tell it the word, and it looks in the file. It makes everything lowercase so it doesn't matter if you type the word with capital letters or not. It then tells you the number of times the word appears. If the file isn't there, it says there's an error.

This code(2):

reads the users.csv file using the csv module. It then checks if the provided username and pin match any entry. If a match is found, it prints a welcome message including the user's name; otherwise, it prints a "user not found" message. Error handling is included in case the users.csv file is missing. Remember that this code assumes your users.csv file has a header row ("username,pin,name").

The code:

import csv

def check\_credentials(username, pin):

with open('users.csv', 'r') as csvfile:

reader = csv.DictReader(csvfile

for row in reader:

if row['username'] == username and row['pin'] == pin:

return row['name']

return None

username = input("Enter username: ")

pin = input("Enter PIN: ")

name = check\_credentials(username, pin)

if name:

print(f"Welcome {name}")

else:

print("Sorry, user not found")

This program checks if a username and PIN are correct. It reads a file that has usernames, PINs, and names. You put in your username and PIN, and it looks in the file. If it finds them and they match, it says "Welcome," and your name. If not, it says that the user wasn't found. If the file isn't there, it gives an error message.

Week8:

This code:

imports the students dictionary from student\_dict.py. It then iterates through the dictionary, calculating the average grade for each student using the calculate\_average function (which handles the case of empty grade lists). Finally, it prints the names of students whose average grade is below 40, along with their average grade formatted to two decimal places.

The code:

import student\_dict

def calculate\_average(grades):

return sum(grades) / len(grades) if grades else 0

for student\_id, student\_data in student\_dict.students.items():

average\_grade = calculate\_average(student\_data['grades'])

if average\_grade < 40:

print(f"{student\_data['name']} (ID: {student\_id}): Average grade is {average\_grade:.2f}")

This code looks at a list of student grades. For each student, it figures out their average grade. Then, it only shows the names of students whose average grade is less than 40, along with their average.

Assessment 1

Code:

import random

import datetime

def get\_user\_choice():

while True:

choice = input("Choose rock, paper, or scissors: ").lower()

if choice in ["rock", "paper", "scissors"]:

return choice

else:

print("Invalid choice. Please try again.")

def get\_computer\_choice():

return random.choice(["rock", "paper", "scissors"])

def determine\_winner(user\_choice, computer\_choice):

if user\_choice == computer\_choice:

return "tie"

elif (user\_choice == "rock" and computer\_choice == "scissors") or \

(user\_choice == "paper" and computer\_choice == "rock") or \

(user\_choice == "scissors" and computer\_choice == "paper"):

return "user"

else:

return "computer"

def play\_game():

username = input("Enter your username: ")

user\_score = 0

computer\_score = 0

while user\_score < 10 and computer\_score < 10:

user\_choice = get\_user\_choice()

computer\_choice = get\_computer\_choice()

print(f"You chose: {user\_choice}, Computer chose: {computer\_choice}")

winner = determine\_winner(user\_choice, computer\_choice)

if winner == "user":

user\_score += 1

print("You win!")

elif winner == "computer":

computer\_score += 1

print("Computer wins!")

else:

print("It's a tie!")

print(f"Score: You - {user\_score}, Computer - {computer\_score}")

save\_score(username, user\_score, computer\_score)

print("Game Over!")

if user\_score >= 10:

print(f"{username} wins the match!")

else:

print("Computer wins the match!")

def save\_score(username, user\_score, computer\_score):

now = datetime.datetime.now()

timestamp = now.strftime("%Y-%m-%d %H:%M:%S")

with open("scores.txt", "a") as f:

f.write(f"{username},{timestamp},{user\_score},{computer\_score}\n")

play\_game()

Assessment 2

Code:

import os

import random

# File paths

users\_file = "users.txt"

questions\_file = "questions.txt"

score\_file = "scores.txt"

# Function to check if user exists

def check\_user\_exists(username):

if os.path.exists(users\_file):

with open(users\_file, 'r') as file:

users = file.readlines()

for line in users:

stored\_user, \_ = line.strip().split(',')

if stored\_user == username:

return True

return False

# Function to register a new user

def register\_user():

username = input("Enter a username: ")

if check\_user\_exists(username):

print("User already exists!")

return None

password = input("Enter a password: ")

with open(users\_file, 'a') as file:

file.write(f"{username},{password}\n")

print("Registration successful!")

return username

# Function to login a user

def login\_user():

username = input("Enter your username: ")

password = input("Enter your password: ")

if os.path.exists(users\_file):

with open(users\_file, 'r') as file:

users = file.readlines()

for line in users:

stored\_user, stored\_password = line.strip().split(',')

if stored\_user == username and stored\_password == password:

print("Login successful!")

return username

print("Invalid username or password.")

return None

# Function to read questions from a file

def read\_questions():

if not os.path.exists(questions\_file):

print("Questions file not found.")

return []

with open(questions\_file, 'r') as file:

lines = file.readlines()

questions = []

for line in lines:

parts = line.strip().split(',')

if len(parts) == 5:

question = {

'question': parts[0],

'correct\_answer': parts[1],

'options': parts[2:]

}

questions.append(question)

return questions

# Function to select 5 random questions

def select\_random\_questions():

questions = read\_questions()

if len(questions) < 5:

print("Not enough questions available.")

return []

return random.sample(questions, 5)

# Function to ask a question

def ask\_question(question):

print(f"\n{question['question']}")

options = question['options'] + [question['correct\_answer']]

random.shuffle(options)

for idx, option in enumerate(options, 1):

print(f"{idx}. {option}")

answer = input("Select the correct answer (1-4): ")

try:

answer\_idx = int(answer) - 1

if options[answer\_idx] == question['correct\_answer']:

return True

else:

return False

except ValueError:

print("Invalid input. Please enter a number between 1 and 4.")

return False

# Function to update score

def update\_score(username, score):

if os.path.exists(score\_file):

with open(score\_file, 'r') as file:

scores = file.readlines()

with open(score\_file, 'w') as file:

updated = False

for line in scores:

user, \_ = line.strip().split(',')

if user == username:

file.write(f"{username},{score}\n")

updated = True

break

if not updated:

file.write(f"{username},{score}\n")

else:

with open(score\_file, 'w') as file:

file.write(f"{username},{score}\n")

print(f"Your score is {score}")

# Function to get current score

def get\_current\_score(username):

if os.path.exists(score\_file):

with open(score\_file, 'r') as file:

scores = file.readlines()

for line in scores:

user, score = line.strip().split(',')

if user == username:

return int(score)

return 0

# Main game function

def play\_quiz():

print("Welcome to the quiz!")

username = None

while not username:

choice = input("1. Login\n2. Register\nEnter choice: ")

if choice == '1':

username = login\_user()

elif choice == '2':

username = register\_user()

else:

print("Invalid choice, please try again.")

score = get\_current\_score(username)

print(f"Your current score: {score}")

while True:

questions = select\_random\_questions()

if not questions:

print("Sorry, there was a problem retrieving the questions.")

return

score = 0

for question in questions:

if ask\_question(question):

score += 1

update\_score(username, score)

play\_again = input("Do you want to play again? (y/n): ")

if play\_again.lower() != 'y':

print("Thanks for playing!")

break

if \_\_name\_\_ == "\_\_main\_\_":

play\_quiz()

Github link

https://github.com/AngelosCharitos/school-python