1. Gra Snake

Snake to zręcznościowa gra typu labirynt, stworzona przez firmę Gremlin Industries. Celem gracza jest zdobycie jak największej liczby punktów poprzez zbieranie jedzenia lub owoców. Gracz przegrywa, gdy wąż uderzy w ścianę lub we własne ciało.

Instrukcja konfiguracji

Aby uruchomić ten skrypt, potrzebujesz jedynie następujących trzech modułów:

- **Pygame:** To zestaw modułów języka Python przeznaczonych do tworzenia gier wideo.

- **Time:** Funkcja ta służy do zliczania liczby sekund, które upłynęły od momentu odniesienia (tzw. epoki).

- **Random:** Funkcja ta służy do generowania liczb losowych w języku Python przy użyciu modułu `random`. **Pygame, Time i Random**

Kod źródłowy:

```
import pygame

import time

import random

pygame.init()

white = (255, 255, 255)

yellow = (255, 255, 102)

black = (0, 0, 0)

red = (213, 50, 80)

green = (0, 255, 0)

blue = (50, 153, 213)

dis_width = 600

dis_height = 400

dis = pygame.display.set_mode((dis_width, dis_height))

pygame.display.set_caption('Snake Game In Python')

clock = pygame.time.Clock()

snake_block = 10

snake_speed = 15

font_style = pygame.font.SysFont("bahnschrift", 25)

score_font = pygame.font.SysFont("comicsansms", 35)

def Your_score(score):
```

```
value = score_font.render("Your  Score: " + str(score), True, yellow)
dis.blit(value, [0, 0])
def our_snake(snake_block,  snake_list):
for x in snake_list:
pygame.draw.rect(dis,  black, [x[0], x[1], snake_block,
snake_block])
def message(msg,  color):
mesg = font_style.render(msg,  True, color)
dis.blit(mesg,  [dis_width / 6, dis_height / 3])
def gameLoop():
game_over = False
game_close = False
x1 = dis_width / 2
y1 = dis_height / 2
x1_change = 0
y1_change = 0
snake_List = []
Length_of_snake = 1
foodx = round(random.randrange(0, dis_width - snake_block) / 10.0)
* 10.0
foody = round(random.randrange(0, dis_height - snake_block) /
10.0) * 10.0
while not game_over:
while game_close == True:
dis.fill(blue)
message(
"You Lost! Press 'C' to Play Again or 'Q' To Quit The Game",
red)
Your_score(Length_of_snake  - 1)
pygame.display.update()
for event in pygame.event.get():
```

```
if event.type == pygame.KEYDOWN:
if event.key == pygame.K_q:
game_over = True
game_close = False
if event.key == pygame.K_c:
gameLoop()
for event in pygame.event.get():
if event.type == pygame.QUIT:
game_over = True
if event.type == pygame.KEYDOWN:
if event.key == pygame.K_LEFT:
x1_change = -snake_block
y1_change = 0
elif event.key == pygame.K_RIGHT:
x1_change = snake_block
y1_change = 0
elif event.key == pygame.K_UP:
y1_change = -snake_block
x1_change = 0
elif event.key == pygame.K_DOWN:
y1_change = snake_block
x1_change = 0
if x1 >= dis_width or x1 < 0 or y1 >= dis_height or y1 < 0:
game_close = True
x1 += x1_change
y1 += y1_change
dis.fill(blue)
pygame.draw.rect(dis,  green, [foodx, foody, snake_block,
snake_block])
snake_Head = []
snake_Head.append(x1)
```

```
snake_Head.append(y1)
snake_List.append(snake_Head)
if len(snake_List) > Length_of_snake:
del snake_List[0]
for x in snake_List[:-1]:
if x == snake_Head:
game_close = True
our_snake(snake_block, snake_List)
Your_score(Length_of_snake - 1)
pygame.display.update()
if x1 == foodx and y1 == foody:
foodx = round(
random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
foody = round(
random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
Length_of_snake += 1
clock.tick(snake_speed)
pygame.quit()
quit()
gameLoop()
```

2. Zrzut ekranu wskazanej strony internetowej

Konfiguracja

`pip install selenium`

`pip install chromedriver-binary==XX.X.XXXX.XX.X`

- „XX.X.XXXX.XX.X” to wersja sterownika chromedriver.

- Wersja sterownika chromedriver musi być zgodna z wersją przeglądarki Google Chrome.

Jak sprawdzić wersję Google Chrome

1. Kliknij ikonę menu w prawym górnym rogu ekranu.

2. Kliknij „Pomoc”, a następnie „Informacje o przeglądarce Google Chrome”.

3. W tym miejscu znajdziesz numer wersji przeglądarki Chrome.

Uruchomienie

`python snapshot_of_given_website.py <url>`

Po uruchomieniu skryptu zrzut ekranu zostanie zapisany w bieżącym katalogu.

Wymagania:

selenium==3.141.0

chromedriver-binary==85.0.4183.38.0

Kod źródłowy:

```
# -*- cofing: utf-8 -*-
import sys
from selenium import webdriver
from selenium.webdriver.chrome.options  import Options
import chromedriver_binary
script_name = sys.argv[0]
options = Options()
options.add_argument('--headless')
driver = webdriver.Chrome(options=options)
try:
url = sys.argv[1]
driver.get(url)
page_width = driver.execute_script('return  document.body.scrollWidth')
page_height = driver.execute_script('return  document.body.scrollHeight')
driver.set_window_size(page_width,  page_height)
driver.save_screenshot('screenshot.png')
driver.quit()
 print("SUCCESS")
except IndexError:
print('Usage: %s URL' % script_name)
```

3. Konwerter mowy na tekst

Ten skrypt w języku Python przekształca mowę wejściową na tekst, wykorzystując techniki NLP (przetwarzania języka naturalnego).

Wymagania

Wymagana instalacja:

* Moduł Python Speech Recognition:

`pip install speechrecognition`

* PyAudio:

* Użytkownicy systemu Linux powinni użyć polecenia:

`sudo apt-get install python3-pyaudio`

* Użytkownicy systemu Windows mogą zainstalować PyAudio, wykonując w terminalu polecenie:

`pip install pyaudio`

* Moduł Python pyttsx3:

`pip install pyttsx3`

Jak uruchomić skrypt

- Wprowadź dźwięk, mówiąc do mikrofonu.

- Uruchom skrypt `converter_terminal.py`.

- Wyświetlony zostanie wynikowy tekst.

Wymagania (użyte moduły Pythona)

PyAudio==0.2.11

SpeechRecognition==3.8.1

Kod źródłowy:

```
import speech_recognition

def record_voice():

microphone = speech_recognition.Recognizer()

with speech_recognition.Microphone() as live_phone:

microphone.adjust_for_ambient_noise(live_phone)

print("I'm trying to hear you: ")

audio = microphone.listen(live_phone)

try:

phrase = microphone.recognize_google(audio, language='en')
```

return phrase

except speech_recognition.UnkownValueError:

return "I didn't understand what you said"

if __name__ == '__main__':

phrase = record_voice()

with open('you_said_this.txt','w') as file:

file.write(phrase)

print('the last sentence you spoke was saved in you_said_this.txt')

4. Zamiana mowy na tekst

Program do konwersji mowy na tekst przy użyciu języka Python

Zależności:

pyttsx3

```python

pip install pyttsx3

```

pyaudio

```python

pip install pyaudio

```

SpeechRecognition

```

pip install SpeechRecognition

```

Uruchamianie:

Tekst zostanie zapisany w pliku output.txt

```

python speech-to-text.py

```

Kod źródłowy:

import pyttsx3

```
import speech_recognition  as sr
import os
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
#print(voices[1].id)
engine.setProperty('voice',voices[0].id)
def speak(audio):
engine.say(audio)
engine.runAndWait()
def get():
r = sr.Recognizer()
with sr.Microphone() as source:
print('say something!')
audio = r.listen(source)
print("done")
try:
text = r.recognize_google(audio)
print('google think you said:\n' +text)
except Exception as e:
print(e)
remember = open('output.txt','w')
remember.write(text)
remember.close()
get()
```

5. Test prędkości

Test prędkości z użyciem Pythona

Zależności:

youtube_dl

```
pip3 install speedtest-cli
```

```

Kod źródłowy:

import subprocess

returned_text = subprocess.check_output("speedtest-cli", shell=True,

universal_newlines=True)

print("The Result of Speed Test")

print(returned_text)

**6. Sprawdzanie pisowni**

Tutaj możesz wpisać dowolne słowo i sprawdzić, czy jest ono zapisane poprawnie.

### Wymagania wstępne

Najpierw należy zainstalować bibliotekę `textblob`.

<!--Instalacja biblioteki-->

>pip install textblob

<!--Dla Jupyter Notebook-->

Polecenie to należy uruchomić w terminalu systemowym lub terminalu wbudowanym w środowisko IDE.

<!--dla Jupyter Notebook-->

Jeśli korzystasz z Jupyter Notebook, użyj poniższego polecenia:

<!--dla Jupyter Notebook-->

>import sys

<!--polecenie-->

>!{sys.executable} -m pip install textblob

### Jak uruchomić skrypt

Najpierw zainstaluj bibliotekę `textblob`, a następnie uruchom skrypt w języku Python.

Kod źródłowy:

from textblob import TextBlob # importing textblob library

t = 1

while t:

a = input("Enter the word to be checked:- ") # incorrect spelling

print("original text: "+str(a)) #printing original text
```

```
b = TextBlob(a) #correcting the text
# prints the corrected spelling
print("corrected text: "+str(b.correct()))
t = int(input("Try Again? 1 : 0 "))
```

7. Podział pliku wideo według określonego przedziału czasowego

Skrypt ten dzieli wideo na dwa pliki po podaniu odpowiednich przedziałów czasowych.

```
pip install ffmpeg-python
```

użycie

```python
python videosplitter.py test.mp4 0 50 out1.mp4 out2.mp4
```

LUB

```python
python videosplitter.py -h
```

Wymagania – ffmpeg==1.4

Kod źródłowy:

```
import ffmpeg
import argparse
parser = argparse.ArgumentParser(description='''Split A media file
into two chunks''')
parser.add_argument('inputfile', help="Input filename")
parser.add_argument('starttime', type=float, help="Start time in
seconds")
parser.add_argument('endtime', type=float, help="End time in
seconds")
parser.add_argument('outputfile1', help="Output filename")
parser.add_argument('outputfile2', help="Output filename")
```

```
args = parser.parse_args()

in1 = ffmpeg.input(args.inputfile)

v1 = in1.filter('trim', start=float(args.starttime), end=(args.endtime))

v2 = in1.filter('trim', start=float(args.endtime))

out1 = ffmpeg.output(v1, args.outputfile1)

out2 = ffmpeg.output(v2, args.outputfile2)

out1.run()

out2.run()
```

8. Dzielenie plików

Skrypt przyjmuje indeks podziału oraz nazwę pliku, a następnie dzieli plik zgodnie z podanym indeksem.

Wymagania wstępne

Aby uruchomić ten skrypt, w systemie hosta musi być zainstalowany język Python.

Sposób uruchomienia skryptu (należy wpisać w terminalu):

`python split_files.py <plik_csv/tekstowy> <indeks_podziału/numer_linii>`

Wymagania:

pandas==1.1.0

Kod źródłowy:

```
import sys

import os

import shutil

import pandas as pd

class Split_Files:

'''

Class file for split file program

'''

def __init__(self, filename, split_number):

'''

Getting the file name and the split index

Initializing the output directory, if present then truncate it.
```

```
Getting the file extension
'''
self.file_name = filename
self.directory = "file_split"
self.split = int(split_number)
if os.path.exists(self.directory):
shutil.rmtree(self.directory)
os.mkdir(self.directory)
if self.file_name.endswith('.txt'):
self.file_extension = '.txt'
else:
self.file_extension = '.csv'
 self.file_number = 1
def split_data(self):
'''
spliting the input csv/txt file according to the index provided
'''
data = pd.read_csv(self.file_name, header=None)
data.index += 1
split_frame = pd.DataFrame()
output_file = f"{self.directory}/split_file{self.file_number}
{self.file_extension}"
for i in range(1, len(data)+1):
split_frame = split_frame.append(data.iloc[i-1])
if i % self.split == 0:
output_file = f"{self.directory}/split_file{self.file_number}
{self.file_extension}"
if self.file_extension == '.txt':
split_frame.to_csv(output_file, header=False, index=False,
sep=' ')
else:
```

```
split_frame.to_csv(output_file, header=False, index=False)
split_frame.drop(split_frame.index, inplace=True)
self.file_number += 1
if not split_frame.empty:
 output_file = f"{self.directory}/split_file{self.file_number}
{self.file_extension}"
split_frame.to_csv(output_file, header=False, index=False)
if __name__ == '__main__':
file, split_number = sys.argv[1], sys.argv[2]
sp = Split_Files(file, split_number)
sp.split_data()
```

9. Podziel folder na podfoldery

Uruchomienie

```
python <nazwa_folderu_wejściowego> <liczba_plików>
```

Kod źródłowy:

```
import glob
import os
from shutil import copy2
import sys
def get_files(path):
'''
return a list of files avialable in given folder
'''
files = glob.glob(f'{path}/*')
return files
def getfullpath(path):
'''
Return absolute path of given file
 '''
return os.path.abspath(path)
```

```
def copyfiles(src, dst):
'''
This function copy file from src to dst
if dst dir is not there it will create new
'''
if not os.path.isdir(dst):
os.makedirs(dst)
copy2(src, dst)
def split(data, count):
'''
Split Given list of files and return generator
'''
for i in range(1, len(data), count):
if i + count-1 > len(data):
start, end = (i-1, len(data))
else:
start, end = (i-1, i+count-1)
yield data[start:end]
def start_process(path, count):
files = get_files(path)
splited_data = split(files, count)
for idx, folder in enumerate(splited_data):
name = f'data_{idx}'
for file in folder:
copyfiles(getfullpath(file), getfullpath(name))
if __name__ == "__main__":
'''
driver code
To run this script
python split_and_copy.py <input folder path> <20>
'''
```

```
if len(sys.argv) != 3:
print("Please provide correct parameters \
\npython split_and_copy.py <input folder path> <count>")
sys.exit(0)
if len(sys.argv) == 3:
path = sys.argv[1]
 if os.path.isdir(path):
count = sys.argv[2]
start_process(path, int(count))
else:
print('Given directory name is not an valid directory')
else:
print('Wrong paramter are provided')
```

10. Automatyzacja arkuszy kalkulacyjnych

Funkcje automatyzacji arkuszy:

- Najpierw wczytaj dwa zestawy danych
- Skrypt porówna oba zestawy danych
- Wynikiem będzie wykres kołowy

Instrukcja automatyzacji arkuszy:

Krok 1:

Otwórz terminal

Krok 2:

Przejdź do katalogu, w którym znajduje się plik Pythona

Krok 3:

Uruchom polecenie: python script.py lub python3 script.py

Krok 4:

Usiądź wygodnie i zrelaksuj się. Pozwól skryptowi wykonać zadanie.

Wymagania

- pandas
- plotly

Kod źródłowy:

```
# importing libraries

import pandas as pd

import plotly.express as px

# storing the dataset

data1 = input("Enter first dataset")

data2 = input("Enter second dataset")

# reading the data

data_read_1 = pd.read_excel(data1)

data_read_2 = pd.read_excel(data2)

# print(df_prices,  df_home_1)

reference  = input("What is the basis of merging? ")

data_total = data_read_2.merge(data_read_1,  on=reference)

# print(df_total)

criteria_1  = input("Enter criteria 1")

criteria_2  = input("Enter criteria 2")

fig = px.pie(data_total[[criteria_1,  criteria_2]],

values=criteria_2, names=criteria_1)

fig.show()
```

11. Zapisywanie wiadomości e-mail w formacie CSV

Projekt ten zawiera prosty skrypt służący do pobierania wiadomości e-mail z serwera IMAP. Wiadomości są zapisywane w prostym, czterokolumnowym pliku CSV.

Zależności

Skrypt wymaga biblioteki BeautifulSoup oraz `lxml` do wyodrębniania tekstu z wiadomości w formacie HTML.

Uruchamianie skryptu

Konieczne jest utworzenie pliku `credentials.txt` zawierającego nazwę użytkownika i hasło do konta na serwerze IMAP (każde w osobnej linii).

Gmail – podobnie jak wielu innych dostawców usług IMAP – wymaga utworzenia oddzielnego „hasła aplikacji”, aby umożliwić działanie tego kodu, więc warto zacząć od tej czynności.

Następnie należy umieścić to hasło w pliku `credentials.txt`.

Na koniec wystarczy uruchomić skrypt poleceniem:

```
python store_emails.py
```

Spowoduje to wygenerowanie pliku `mails.csv` w bieżącym katalogu.

Wygenerowany plik CSV zawiera następujące pola dla każdej wiadomości:

* Data
* Od (Nadawca)
* Temat
* Treść wiadomości

Wymagania:

beautifulsoup4

lxml

Kod źródłowy:

```
#!/usr/bin/env python
import csv
import email
from email import policy
import imaplib
import logging
import os
import ssl
from bs4 import BeautifulSoup
credential_path = "credentials.txt"
csv_path = "mails.csv"
logger = logging.getLogger('imap_poller')
host = "imap.gmail.com"
port = 993
ssl_context = ssl.create_default_context()
def connect_to_mailbox():
```

```
# get mail connection
mail = imaplib.IMAP4_SSL(host, port, ssl_context=ssl_context)
with open(credential_path, "rt") as fr:
 user = fr.readline().strip()
pw = fr.readline().strip()
mail.login(user, pw)
# get mail box response and select a mail box
status, messages = mail.select("INBOX")
return mail, messages
# get plain text out of html mails
def get_text(email_body):
soup = BeautifulSoup(email_body, "lxml")
return soup.get_text(separator="\n", strip=True)
def write_to_csv(mail, writer, N, total_no_of_mails):
for i in range(total_no_of_mails, total_no_of_mails - N, -1):
res, data = mail.fetch(str(i), "(RFC822)")
response = data[0]
if isinstance(response, tuple):
msg = email.message_from_bytes(response[1], policy=policy.default)
# get header data
 email_subject = msg["subject"]
email_from = msg["from"]
email_date = msg["date"]
email_text = ""
# if the email message is multipart
if msg.is_multipart():
# iterate over email parts
for part in msg.walk():
# extract content type of email
content_type = part.get_content_type()
content_disposition = str(part.get("Content-Disposition"))
```

```
try:
# get the email email_body
email_body = part.get_payload(decode=True)
if email_body:
email_text = get_text(email_body.decode('utf-8'))
except Exception as exc:
logger.warning('Caught exception: %r', exc)
if (
content_type == "text/plain"
and "attachment" not in content_disposition
):
# print text/plain emails and skip attachments
# print(email_text)
 pass
elif "attachment" in content_disposition:
pass
else:
# extract content type of email
content_type = msg.get_content_type()
# get the email email_body
email_body = msg.get_payload(decode=True)
if email_body:
email_text = get_text(email_body.decode('utf-8'))
if email_text is not None:
# Write data in the csv file
row = [email_date, email_from, email_subject, email_text]
writer.writerow(row)
else:
logger.warning('%s:%i: No message extracted', "INBOX", i)
def main():
mail, messages = connect_to_mailbox()
```

```
logging.basicConfig(level=logging.WARNING)

total_no_of_mails = int(messages[0])

 # no. of latest mails to fetch

# set it equal to total_no_of_emails to fetch all mail in the inbox

N = 2

with open(csv_path, "wt", encoding="utf-8", newline="") as fw:

writer = csv.writer(fw)

writer.writerow(["Date", "From", "Subject", "Text mail"])

try:

write_to_csv(mail, writer, N, total_no_of_mails)

except Exception as exc:

logger.warning('Caught exception: %r', exc)

if __name__ == "__main__":

main()
```

12. Wyszukiwanie ciągu znaków w wielu plikach

Znajduje plik zawierający podany ciąg znaków w wybranym folderze.

Wymagania wstępne

Jedynym wymaganiem jest Python3! Do uruchomienia skryptu nie są potrzebne żadne zewnętrzne moduły.

Jak uruchomić skrypt

Aby uruchomić ten skrypt, musisz mieć zainstalowanego Pythona3 (nie Pythona2). Polecenie uruchamiające to po prostu `python3 findstring.py`; program poprosi następnie o podanie dwóch informacji: szukanego ciągu znaków oraz lokalizacji, w której ma odbyć się wyszukiwanie.

Kod źródłowy:

```
import os

text = input("input text : ")

path = input("path : ")

# os.chdir(path)

def getfiles(path):

f = 0

os.chdir(path)
```

```
files = os.listdir()
# print(files)
for file_name in files:
abs_path = os.path.abspath(file_name)
if os.path.isdir(abs_path):
getfiles(abs_path)
if os.path.isfile(abs_path):
f = open(file_name, "r")
if text in f.read():
f = 1
print(text + " found in ")
final_path = os.path.abspath(file_name)
print(final_path)
return True
if f == 1:
print(text + " not found! ")
return False
getfiles(path)
```

13. Gra w wisielca w terminalu

Projekt zawiera prosty skrypt w języku Python, umożliwiający grę w wisielca w terminalu.

Wymagania wstępne

Brak

Jak uruchomić skrypt

- Uruchom skrypt hangman.py.
- Zacznij odgadywać słowo.

Kod źródłowy:

```
import random
from json import load
# function to randomly get one word from words.py and convert the word to
```

```
uppercase
def get_word():
with open('words.json') as json_file:
data = load(json_file)
wordArray = data["word_list"]
word = random.choice(wordArray)
word = word.upper()
return word
# function to play the game
def play(word):
# intialise variable
word_completion = "_" * len(word) # generate a line to show the number
of word
guessed = False # indicate the status of guess
guessed_letters = [] # store guessed letters
guessed_words = [] # store guessed words
tries = 6 # user have 6 times of wrong
# display message and the format of the hangman
print("Let's play Hangman!")
 print(display_hangman(tries))
print(word_completion)
print("\n")
print("Length of the word: ", len(word))
print("\n")
# user can keep guessing when the tries is more than 0 and the answer is
not found yet.
while not guessed and tries > 0:
# Display message and ask for user input and convert it into uppercase
guess = input("Please guess a letter or the word: ").upper()
# check the length of the user input and is it alpha or not
if len(guess) == 1 and guess.isalpha():
```

```
# display message when user guess the same letter twice
if guess in guessed_letters:
print("You already guessed the letter", guess)
# display message and deduct the tries when user guess the wrong
letter
elif guess not in word:
print(guess, "is not in the word.")
tries -= 1
guessed_letters.append(guess)
# dispay message and store the letter when the user guess the correct
letter
else:
print("Good job,", guess, "is in the word!")
guessed_letters.append(guess)
word_as_list = list(word_completion)
indices = [i for i, letter in enumerate(word) if letter == guess]
for index in indices:
word_as_list[index] = guess
# join the guess word in the word_completion
word_completion = "".join(word_as_list)
# if there is not blank space in word_completion change the status
of guess to true
if "_" not in word_completion:
guessed = True
# check the length of the user input and is it alpha or not
elif len(guess) == len(word) and guess.isalpha():
# display message when user guess the same letter twice
if guess in guessed_words:
print("You already guessed the word", guess)
 # display message and deduct the tries when user guess the wrong
letter
```

```
elif guess != word:
print(guess, "is not the word.")
tries -= 1
guessed_words.append(guess)
# change the status of guess
else:
guessed = True
word_completion = word
# display error message for user
else:
print("Not a valid guess.")
# display the format of hangman each time of guess
print(display_hangman(tries))
print(word_completion)
print("\n")
print("Length of the word: ", len(word))
print("\n")
# if the variable of guess is true means user win the game
if guessed:
print("Congrats, you guessed the word! You win!")
 # else means user lose the game.
else:
print("Sorry, you ran out of tries. The word was " + word + ". Maybe
next time!")
# function to display the format of hangman
def display_hangman(tries):
stages = ["""
--------
| |
| 0
| \\|/
```

```
| |
| / \\
-
""",
"""
--------
| |
| 0
| \\|/
| |
| /
-
""",
 """
--------
| |
| 0
| \\|/
| |
|
-
""",
"""
--------
| |
| 0
| \\|
| |
|
-
""",
```

```
"""
--------
| |
| 0
| |
| |
|
 -
""",
"""
--------
| |
| 0
|
|
|
-
""",
"""
--------
| |
|
|
|
|
-
"""
]
return stages[tries]
# main function to start the game
def main():
```

word = get_word()

play(word)

while input("Play Again? (Y/N): ").upper() == "Y":

word = get_word()

play(word)

if __name__ == "__main__":

main()

14. Pasek postępu w terminalu przy zmianie rozmiaru obrazów

Pasek postępu w terminalu przy zmianie rozmiaru obrazów

W tym przykładzie wykorzystuję zmianę rozmiaru obrazów do zaprezentowania działania paska postępu.

Podczas jednoczesnego przetwarzania wielu obrazów pasek postępu pozwala

monitorować, ile z nich zostało już przetworzonych.

W tym celu wykorzystuję bibliotekę tqdm

` pip install tqdm `

Biblioteka ta służy do wyświetlania paska postępu.

Do zmiany rozmiaru obrazów

` pip install Pillow `

Wymagania:

tqdm==4.48.2

PIL==1.1.6

Kod źródłowy:

z tqdm import tqdm

z PIL import Image

import os

z time import sleep

def Resize_image(size, image):

if os.path.isfile(image):

try:

```
im = Image.open(image)

im.thumbnail(size, Image.ANTIALIAS)

im.save("resize/" + str(image) + ".jpg")

except Wyjątek jak np.:

print(f"Błąd: {str(ex)} do {image}")

path = input("Podaj ścieżkę do obrazów: ")

size = input("Rozmiar Wysokość, Szerokość: ")

size = tuple(map(int, size.split(",")))

os.chdir(path)

list_images = os.listdir(path)

jeśli "resize" nie ma w list_images:

os.mkdir("resize")

for image in tqdm(list_images, desc="Zmiana rozmiaru obrazów"):

Resize_image(size, image)

sleep(0.1)

print("Zmiana rozmiaru zakończona!")
```

15. Zamiana tekstu na mowę

Po uruchomieniu skryptu tekst z pliku abc.txt zostanie przekonwertowany na format MP3, zapisany, a następnie odtworzony na Twoim urządzeniu.

Wymagania wstępne

- plik abc.txt z tekstem

- moduł gTTS==2.1.1 (pobierany za pomocą polecenia `pip install gTTS`)

- moduł os (pobierany za pomocą polecenia `pip install os`)

Jak uruchomić skrypt

Wpisz wybrany tekst do pliku abc.txt, a następnie uruchom plik txtToSpeech.py. Można to zrobić, wpisując `python txtToSpeech.py` w terminalu.

Wymagania – gTTS==2.1.1

Kod źródłowy:

```
from gtts import gTTS

import os

file = open("abc.txt", "r").read()
```

```
speech = gTTS(text=file, lang='en', slow=False)
speech.save("voice.mp3")
os.system("voice.mp3")
#print(file)
```

16. Edytor tekstu

Kod źródłowy:

```
from tkinter import *
import tkinter.filedialog
class TextEditor:
@staticmethod
def quit_app(event=None):
root.quit()
def open_file(self, event=None):
txt_file = tkinter.filedialog.askopenfilename(parent=root,
initialdir="./examples")
if txt_file:
self.text_area.delete(1.0, END)
with open(txt_file) as _file:
self.text_area.insert(1.0, _file.read())
root.update_idletasks()
def save_file(self, event=None):
file = tkinter.filedialog.asksaveasfile(mode='w')
if file != None:
data = self.text_area.get('1.0', END + '-1c')
file.write(data)
file.close()
def __init__(self, root):
self.text_to_write = ""
root.title("TextEditor")
root.geometry("600x550")
```

```
frame = Frame(root, width=600, height=550)
scrollbar = Scrollbar(frame)
self.text_area = Text(frame , width=600, height=550,
yscrollcommand=scrollbar.set, padx = 10, pady=10)
scrollbar.config(command=self.text_area.yview)
scrollbar.pack(side="right", fill="y")
self.text_area.pack(side="left", fill="both", expand=True)
frame.pack()
the_menu = Menu(root)
file_menu = Menu(the_menu, tearoff=0)
file_menu.add_command(label="Open", command=self.open_file)
file_menu.add_command(label="Save", command=self.save_file)
file_menu.add_separator()
file_menu.add_command(label="Quit", command=self.quit_app)
the_menu.add_cascade(label="File", menu=file_menu)
root.config(menu=the_menu)
root = Tk()
text_editor = TextEditor(root)
root.mainloop()
```

17. Analiza pliku tekstowego

```
# -*- cofing: utf-8 -*-
import os
import sys
import collections
import string
script_name = sys.argv[0]
res = {
"total_lines":"",
"total_characters":"",
"total_words":"",
```

```
"unique_words":"",
"special_characters":""
}
try:
textfile = sys.argv[1]
with open(textfile, "r", encoding = "utf_8") as f:
data = f.read()
res["total_lines"] = data.count(os.linesep)
 res["total_characters"] = len(data.replace(" ","")) - res["total_lines"]
counter = collections.Counter(data.split())
d = counter.most_common()
res["total_words"] = sum([i[1] for i in d])
res["unique_words"]  = len([i[0] for i in d])
special_chars = string.punctuation
res["special_characters"]  = sum(v for k, v in
collections.Counter(data).items()  if k in special_chars)
except IndexError:
print('Usage: %s TEXTFILE' % script_name)
except IOError:
print('"%s" cannot be opened.' % textfile)
print(res)
```

18.Kółko i krzyżyk

Opis

Gra w kółko i krzyżyk dla dwóch graczy, napisana w języku Python.

Program pobiera od graczy współrzędne (x i y) ich ruchów. Gracze posługują się symbolami X oraz O i na zmianę wprowadzają wybrane współrzędne, dążąc do wygranej.

Wymagania wstępne

Możesz skorzystać z dowolnego kompilatora języka Python online lub pobrać środowisko programistyczne (IDE) ze strony:

https://www.python.org/

Jak uruchomić

Wystarczy uruchomić polecenie:

```sh
python tic_tac_toe.py
```

Kod źródłowy:

```
def start():
global board
board = [
['',',''],
['',',''],
['',',']
]
def print_board():
print(' ---------')
for row in board:
print(' ',row[0],'|',row[1],'|',row[2])
print(' ---------')
def have_empty_room():
for row in board:
for room in row:
if not room:
return True
return False
def set_room_state(roomxy,state):
x = int(roomxy[0])-1
 y = int(roomxy[1])-1
row = board[x]
room = row[y]
if not room:
board[x][y] = state
return True
```

```
return False
def check_xy(xy):
xy = str(xy)
if len(xy) != 2:
return False
if int(xy[0]) > 3 or int(xy[0]) < 1 or int(xy[1]) > 3 or int(xy[1]) < 1:
return False
return True
def check_for_win():
if board[0][0] == board[0][1] == board[0][2] != '':
winner = board[0][0]
print(f'{winner} won!')
elif board[1][0] == board[1][1] == board[1][2] != '':
winner = board[1][0]
print(f'{winner} won!')
 elif board[2][0] == board[2][1] == board[2][2] != '':
winner = board[2][0]
print(f'{winner} won!')
elif board[0][0] == board[1][0] == board[2][0] != '':
winner = board[0][0]
print(f'{winner} won!')
elif board[0][1] == board[1][1] == board[2][1] != '':
winner = board[0][1]
print(f'{winner} won!')
elif board[0][2] == board[1][2] == board[2][2] != '':
winner = board[0][0]
print(f'{winner} won!')
elif board[0][0] == board[1][1] == board[2][2] != '':
winner = board[0][0]
print(f'{winner} won!')
elif board[0][2] == board[1][1] == board[2][0] != '':
```

```
winner = board[0][2]
print(f'{winner} won!')
else:
 return False
return True
turn = 'o'
start()
while have_empty_room():
print_board()
print('\n')
if turn == 'o':
turn = 'x'
else:
turn = 'o'
print(f'{turn}\'s Turn!')
while True:
xy = int(input('enter x and y: '))
if check_xy(xy):
if set_room_state(str(xy),turn):
break
print('This room is full!')
continue
print('Error!')
continue
 if check_for_win():
break
print_board()
print('Game Over')
input()
```

19. Tic-Tac-Toe-AI

Dodawanie prostego systemu AI do gry w kółko i krzyżyk:

3 tryby:

- Gracz kontra gracz (tryb dla 2 osób)

- Gracz kontra AI (tryb dla 1 osoby)

- AI kontra AI (*dla zabawy*)

Źródła

Logika

- Optymalne ruchy w grze w kółko i krzyżyk

DEMO:

Plansza będzie wyświetlana po każdym ruchu gracza.

Plansza będzie wyglądać tak!

Układ pól na tej planszy 3x3 odpowiada układowi klawiszy na **klawiaturze numerycznej (po prawej stronie klawiatury)**.

Kod źródłowy:

```
####

TIC TAC TOE ####

#START;

#FUNCTIONS;

def default():

#To be printed as Default;

print("\nWelcome!  Let's play TIC TAC TOE!\n")

def rules():

print("The board will look like this!")

print("The positions of this 3 x 3 board is same as the right side of your

key board.\n")

print(" 7 | 8 | 9 ")

print("-----------")

print(" 4 | 5 | 6 ")

print("-----------")

print(" 1 | 2 | 3 ")
```

```
print("\nYou just have to input the position(1-9).")
def play():
#Asking if the player is ready;
return input("\nAre you ready to play the game? Enter [Y]es or
[N]o.\t").upper().startswith('Y')
def names():
#Player names input;
p1_name=input("\nEnter NAME of PLAYER 1:\t").capitalize()
p2_name=input("Enter NAME of PLAYER 2:\t").capitalize()
return (p1_name, p2_name)
def choice():
#Player choice input;
p1_choice = ' '
p2_choice = ' '
while p1_choice != 'X' or p1_choice != 'O': #while loop; if the
entered value isn't X or O;
#WHILE LOOP STARTS
p1_choice = input(f"\n{p1_name}, Do you want to be X or O?\t")
[0].upper()
#The input above has [0].upper() in the end;
#So the user can enter x, X, xxxx or XXX; the input will always be
taken as X;
#Thereby, increasing the user input window;
if p1_choice == 'X' or p1_choice == 'O':
#if entered value is X or O; get out of the loop;
break
print("INVALID INPUT! Please Try Again!")
#if the entered value isn't X or O, re-run the while loop;
#WHILE LOOP ENDS
#Assigning the value to p2 and then diplaying the values;
if p1_choice == 'X':
```

```
p2_choice = 'O'
elif p1_choice == 'O':
p2_choice = 'X'
return (p1_choice, p2_choice)
def first_player():
#This function will randomly decide who will go first;
import random
return random.choice((0, 1))
def display_board(board, avail):
print(" " + " {} | {} | {} ".format(board[7],board[8],board[9]) + " "
+ " {} | {} | {} ".format(avail[7],avail[8],avail[9]))
print(" " + "-----------" + " " + "-----------")
print(" " + " {} | {} | {} ".format(board[4],board[5],board[6]) + " "
+ " {} | {} | {} ".format(avail[4],avail[5],avail[6]))
print(" " + "-----------" + " " + "-----------")
print(" " + " {} | {} | {} ".format(board[1],board[2],board[3]) + " "
+ " {} | {} | {} ".format(avail[1],avail[2],avail[3]))
def player_choice(board, name, choice):
position = 0
#Initialising position as 0^; so it passes through the while loop;
while position not in [1,2,3,4,5,6,7,8,9] or not space_check(board,
position):
position = int(input(f'\n{name} ({choice}), Choose your next position:
(1-9) \t'))
if position not in [1,2,3,4,5,6,7,8,9] or not space_check(board, position)
or position == "":
#To check whether the given position is in the set [1-9] or whether it
is empty or occupied;
print(f"INVALID INPUT. Please Try Again!\n")
print("\n")
return position
```

```
# THIS IS THE FUNCTION WHERE AI IS ADDED:
def CompAI(board, name, choice):
position = 0
 possibilities = [x for x, letter in enumerate(board)  if letter == ' ' and x != 0]
# including both X and O, since if computer will win, he will place a
choice there, but if the component will win --> we have to block that move
for let in ['O', 'X']:
for i in possibilities:
# Creating a copy of the board everytime,  placing the move and
checking if it wins;
# Creating a copy like this and not this boardCopy = board, since
changes to boardCopy changes the original board;
boardCopy = board[:]
boardCopy[i] = let
if(win_check(boardCopy,  let)):
position = i
return position
openCorners = [x for x in possibilities  if x in [1, 3, 7, 9]]
if len(openCorners) > 0:
position = selectRandom(openCorners)
return position
if 5 in possibilities:
position = 5
return position
openEdges = [x for x in possibilities if x in [2, 4, 6, 8]]
if len(openEdges) > 0:
position = selectRandom(openEdges)
return position
def selectRandom(board):
import random
ln = len(board)
```

```
r = random.randrange(0,ln)
return board[r]
def place_marker(board, avail, choice, position):
#To mark/replace the position on the board list;
board[position] = choice
avail[position] = ' '
def space_check(board, position):
#To check whether the given position is empty or occupied;
return board[position] == ' '
def full_board_check(board):
#To check if the board is full, then the game is a draw;
for i in range(1,10):
if space_check(board, i):
return False
return True
def win_check(board, choice):
#To check if one of the following patterns are true; then the respective
player has won!;
#HORIZONTAL CHECK;
return (
( board[1] == choice and board[2] == choice and board[3] == choice )
or ( board[4] == choice and board[5] == choice and board[6] == choice )
or ( board[7] == choice and board[8] == choice and board[9] == choice )
#VERTICAL CHECK;
or ( board[1] == choice and board[4] == choice and board[7] == choice )
or ( board[2] == choice and board[5] == choice and board[8] == choice )
or ( board[3] == choice and board[6] == choice and board[9] == choice )
#DIAGONAL CHECK;
or ( board[1] == choice and board[5] == choice and board[9] == choice )
or ( board[3] == choice and board[5] == choice and board[7] == choice ) )
def delay(mode):
```

```
if mode == 2:
import time
time.sleep(2)
def replay():
#If the users want to play the game again?
return input('\nDo you want to play again? Enter [Y]es or [N]o:
').lower().startswith('y')
#MAIN PROGRAM STARTS;
print("\n\t\t NAMASTE! \n")
input("Press ENTER to start!")
default()
rules()
while True:
################################################################################
########
#Creating the board as a list; to be kept replacing it with user input;
theBoard = [' ']*10
#Creating the available options on the board:
available = [str(num) for num in range(0,10)] # a List Comprehension
#available = '0123456789'
print("\n[0]. Player vs. Computer")
print("[1]. Player vs. Player")
print("[2]. Computer vs. Computer")
mode = int(input("\nSelect an option [0]-[2]: "))
if mode == 1:
#Asking Names;
p1_name, p2_name = names()
# Asking Choices; Printing choices; X or O;
p1_choice, p2_choice = choice()
print(f"\n{p1_name}:", p1_choice)
print(f"{p2_name}:", p2_choice)
```

```
elif mode == 0:
p1_name = input("\nEnter NAME of PLAYER who will go against the
Computer:\t").capitalize()
p2_name = "Computer"
# Asking Choices; Printing choices; X or O;
p1_choice, p2_choice = choice()
print(f"\n{p1_name}:", p1_choice)
 print(f"{p2_name}:", p2_choice)
else:
p1_name = "Computer1"
p2_name = "Computer2"
p1_choice, p2_choice = "X", "O"
print(f"\n{p1_name}:", p1_choice)
print(f"\n{p2_name}:", p2_choice)
#Printing randomly who will go first;
if first_player():
turn = p2_name
else:
turn = p1_name
print(f"\n{turn} will go first!")
#Asking the user, if ready to play the game; Output will be True or False;
if(mode == 2):
ent = input("\nThis is going to be fast! Press Enter for the battle to
begin!\n")
play_game = 1
else:
play_game = play()
while play_game:
############################
#PLAYER1
if turn == p1_name:
```

```
#Displaying the board;
display_board(theBoard, available)
#Position of the input;
if mode != 2:
position = player_choice(theBoard, p1_name, p1_choice)
else:
position = CompAI(theBoard, p1_name, p1_choice)
print(f'\n{p1_name} ({p1_choice}) has placed on {position}\n')
#Replacing the ' ' at *position* to *p1_choice* in *theBoard* list;
place_marker(theBoard, available, p1_choice, position)
#To check if Player 1 has won after the current input;
if win_check(theBoard, p1_choice):
display_board(theBoard, available)
print("~~~
```

```
turn = p2_name
############################
#PLAYER2
elif turn == p2_name:
#Displaying the board;
 display_board(theBoard, available)
#Position of the input;
if(mode == 1):
position = player_choice(theBoard, p2_name, p2_choice)
else:
position = CompAI(theBoard, p2_name, p2_choice)
print(f'\n{p2_name} ({p2_choice}) has placed on {position}\n')
#Replacing the ' ' at *position* to *p2_choice* in *theBoard* list;
place_marker(theBoard, available, p2_choice, position)
#To check if Player 2 has won after the current input;
if win_check(theBoard, p2_choice):
display_board(theBoard, available)
print("~~~
```

```
print("~~~~~~~~~~~~~~~~~~")

break

#If none of the above is possible, next turn of Player 2;

else:

turn = p1_name

#If the users want to play the game again?

if replay():

#if Yes;

continue

else:

#if No;

break

##########################################################################
########

print("\n\n\t\t\tTHE END!")
```

20. Czas ładowania strony internetowej

Ten skrypt pobiera od użytkownika adres URL i zwraca czas potrzebny na załadowanie danej strony.

Jak tego używać?

1. Wystarczy wpisać poniższe polecenie w wierszu poleceń:

```
python time_to_load_website.py
```

2. Skrypt poprosi o podanie adresu URL. Wpisz adres i naciśnij klawisz Enter, aby zobaczyć działanie skryptu.

Przykładowe użycie:

```
<p align = "center">

<img src="sample.PNG" alt="przykład">

</p>
```

Kod źródłowy:

```
from urllib.request import urlopen

import time

def get_load_time(url):
```

"""This function takes a user defined url as input

and returns the time taken to load that url in seconds.

Args:

url (string): The user defined url.

Returns:

time_to_load (float): The time taken to load the website in seconds.

"""

if ("https" or "http") in url: # Checking for presence of protocols

open_this_url = urlopen(url) # Open the url as entered by the user

else:

open_this_url = urlopen("https://" + url) # Adding https to the url

start_time = time.time() # Time stamp before the reading of url starts

open_this_url.read() # Reading the user defined url

end_time = time.time() # Time stamp after the reading of the url

open_this_url.close() # Closing the instance of the urlopen object

time_to_load = end_time - start_time

return time_to_load

if __name__ == '__main__':

url = input("Enter the url whose loading time you want to check: ")

print(f"\nThe time taken to load {url} is {get_load_time(url):.2} seconds.")

21. Aplikacja Todo wykorzystująca Flask

Wykonaj operację taką jak

1. Dodaj zadanie

2. Usuń zadanie

3. Zaktualizuj zadanie

Aby uruchomić aplikację

- Utwórz środowisko wirtualne

- Zainstaluj wymagania

`pip install requirements.txt`

- Uruchom aplikację

`py app.py`

Wymagania

Flask==1.1.2

Flask-SQLAlchemy==2.4.4

Kod źródłowy:

```
from flask import Flask, render_template, url_for, request, redirect
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///test.db"
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
db = SQLAlchemy(app)
class Todo(db.Model):
id = db.Column(db.Integer, primary_key=True)
content = db.Column(db.String(200), nullable=False)
completed = db.Column(db.Integer, default=0)
pub_date = db.Column(db.DateTime, nullable=False,
default=datetime.utcnow)
def __repr__(self):
return "<Task %r>" % self.id
@app.route("/", methods=["POST", "GET"])
def index():
if request.method == "POST":
task_content = request.form["task"]
new_task = Todo(content=task_content)
try:
db.session.add(new_task)
db.session.commit()
return redirect("/")
 except:
return "There is an issue"
```

```
else:
tasks = Todo.query.order_by(Todo.pub_date).all()
return render_template("index.html", tasks=tasks)
@app.route("/delete/<int:id>")
def delete(id):
task = Todo.query.get_or_404(id)
try:
db.session.delete(task)
db.session.commit()
return redirect("/")
except:
return "This is an Problem while deleting"
@app.route("/update/<int:id>", methods=["POST", "GET"])
def update(id):
task = Todo.query.get_or_404(id)
if request.method == "POST":
task.content = request.form["task"]
try:
 db.session.commit()
return redirect("/")
except:
return "There is an issue"
else:
tasks = Todo.query.order_by(Todo.pub_date).all()
return render_template("index.html", update_task=task, tasks=tasks)
if __name__ == "__main__":
app.run(debug=True)
```

22. Test szybkości pisania

```
import time
string = "Python is an interpreted, high-level programming language"
```

```
word_count = len(string.split())
border = '-+-'*10
def createbox():
print(border)
print()
print('Enter the phrase as fast as possible and with accuracy')
print()
while 1:
t0 = time.time()
createbox()
print(string,'\n')
inputText = str(input())
t1 = time.time()
lengthOfInput = len(inputText.split())
accuracy = len(set(inputText.split()) & set(string.split()))
accuracy = (accuracy/word_count)
timeTaken = (t1 - t0)
 wordsperminute = (lengthOfInput/timeTaken)*60
#Showing results now
print('Total words \t :' ,lengthOfInput)
print('Time used \t :',round(timeTaken,2),'seconds')
print('Your accuracy \t :',round(accuracy,3)*100,'%')
print('Speed is \t :' , round(wordsperminute,2),'words per minute')
print("Do you want to retry",end='')
if input():
continue
else:
print('Thank you , bye bye .')
time.sleep(1.5)
break
```

23. Bot do automatycznego przestawania obserwowania na Instagramie

bb8 – Twój osobisty bot

`bb8` to sympatyczna nazwa świetnego bota, który sprawdza, które z obserwowanych przez Ciebie osób na Instagramie nie obserwują Cię w zamian.

Jak uruchomić

* Pobierz najnowszą wersję sterownika ChromeDriver i umieść ją w lokalizacji `C:\Program Files (x86)\chromedriver.exe`. Możesz go pobrać [stąd](https://chromedriver.chromium.org/).

* Uruchom skrypt i wprowadź nazwę użytkownika oraz hasło do swojego konta na Instagramie.

* To wszystko. W terminalu pojawi się lista wszystkich obserwowanych przez Ciebie kont, które nie obserwują Cię wzajemnie.

Uwaga

Pamiętaj o pobraniu zależności wymienionych w pliku requirements.txt!

Wykorzystane moduły

* selenium

Status prac

Bot jest obecnie w pełni funkcjonalny. Jednak zmiany w interfejsie (frontendzie) Instagrama mogą wymagać modyfikacji skryptu.

``

Kod źródłowy:

```
from selenium import webdriver

from getpass import getpass

import time

# Class for the bot

class InstaBot:

# Initializes bot

def __init__(self):

self.username = input('Enter your username:')

self.pw = getpass('Enter your password(will NOT appear as you type):')

self.PATH = r"C:\Program Files (x86)\chromedriver.exe"

self.driver = webdriver.Chrome(self.PATH)

# Starts Instagram

def start(self):
```

```
self.driver.get('https://www.instagram.com/')
time.sleep(2)
 return
# Logs into your account, also closes various dialogue boxes that open on
# the way
def login(self):
user_field = self.driver.find_element_by_xpath(
'//*[@id="loginForm"]/div/div[1]/div/label/input')
pw_field = self.driver.find_element_by_xpath(
'//*[@id="loginForm"]/div/div[2]/div/label/input')
login_button = self.driver.find_element_by_xpath(
'//*[@id="loginForm"]/div/div[3]/button/div')
user_field.send_keys(self.username)
pw_field.send_keys(self.pw)
login_button.click()
time.sleep(2.5)
not_now1 = self.driver.find_element_by_xpath(
'//*[@id="react-root"]/section/main/div/div/div/div/button')
not_now1.click()
time.sleep(2)
not_now2 = self.driver.find_element_by_xpath(
'/html/body/div[4]/div/div/div/div[3]/button[2]')
not_now2.click()
time.sleep(1)
return
# Opens your profile
def open_profile(self):
profile_link = self.driver.find_element_by_xpath(
'//*[@id="react-root"]/section/main/section/div[3]'
'/div[1]/div/div[2]/div[1]/a')
profile_link.click()
```

```
time.sleep(2)
return
# Opens the list of the people you follow
def open_following(self):
following_link = self.driver.find_element_by_xpath(
'/html/body/div[1]/section/main/div/header/section/ul/li[3]/a')
following_link.click()
return
# Gets the list of the people you follow
def get_following(self):
xpath = '/html/body/div[4]/div/div/div[2]'
self.following = self.scroll_list(xpath)
return
# Opens the link to 'Followers'
def open_followers(self):
 followers_link = self.driver.find_element_by_xpath(
'//*[@id="react-root"]/section/main/div/header/section/ul/li[2]/a')
followers_link.click()
return
# Gets the list of followers
def get_followers(self):
xpath = '/html/body/div[4]/div/div/div[2]'
self.followers = self.scroll_list(xpath)
return
# Scrolls a scroll box and retrieves their names
def scroll_list(self, xpath):
time.sleep(2)
scroll_box = self.driver.find_element_by_xpath(xpath)
last_ht, ht = 0, 1
# Keep scrolling till you can't go down any further
while last_ht != ht:
```

```
last_ht = ht
time.sleep(1)
ht = self.driver.execute_script(
"""
arguments[0].scrollTo(0, arguments[0].scrollHeight);
 return arguments[0].scrollHeight;
""", scroll_box)
# Gets the list of accounts
links = scroll_box.find_elements_by_tag_name('a')
names = [name.text for name in links if name.text != '']
# Closes the box
close_btn = self.driver.find_element_by_xpath(
'/html/body/div[4]/div/div/div[1]/div/div[2]/button/div')
close_btn.click()
return names
# Prints the list of people you follow who don't follow you back in
# terminal
def get_unfollowers(self):
self.unfollowers = [
x for x in self.following if x not in self.followers
]
for name in self.unfollowers:
print(name)
return
 # Closes the driver
def close(self):
self.driver.quit()
return
def main():
# Bot method calls
bb8 = InstaBot()
```

```
bb8.start()
bb8.login()
bb8.open_profile()
bb8.open_following()
bb8.get_following()
bb8.open_followers()
bb8.get_followers()
bb8.get_unfollowers()
bb8.close()
if __name__ == '__main__':
main()
```

24. Unikalne słowa w pliku tekstowym

Skrypt wyświetlający unikalne słowa w danym pliku tekstowym.

Kod źródłowy:

```
import re
# script to fetch unique sorted words from a text file.
list_of_words = []
# Alternate Method to insert file
# filename = input("Enter file name: ")
filename = "text_file.txt"
with open(filename, "r") as f:
for line in f:
# if case is ignored then Great and great are same words
list_of_words.extend(re.findall(r"[\w]+", line.lower()))
# else use this alternate method:
# list_of_words.extend(re.findall(r"[\w]+", line))
# Creating a dictionary to store the number of occurence of a word
unique = {}
for each in list_of_words:
if each not in unique:
```

```
unique[each] = 0

unique[each] += 1

# Creating a list to sort the final unique words

s = []

# If occurence of a word(val) is 1 then it is unique

for key, val in unique.items():

if val == 1:

s.append(key)

print(sorted(s))
```

25. Unstructured Supplementary Service Data

Unstructured Supplementary Service Data (USSD) – technologia znana również jako „kody szybkiego wyboru" (Quick Codes) lub „kody funkcyjne" (Feature codes) – to protokół komunikacyjny wykorzystywany przez telefony komórkowe GSM do wymiany danych z systemami komputerowymi operatora sieci komórkowej. USSD znajduje zastosowanie m.in. w przeglądaniu stron WAP, usługach typu „callback" dla telefonów na kartę (prepaid), usługach płatności mobilnych, usługach opartych na lokalizacji, serwisach informacyjnych obsługiwanych za pomocą menu oraz w procesie konfiguracji telefonu w sieci.

WYMAGANE MODUŁY

1. random

2. time

3. sys

PROCES URUCHOMIENIA

1. wykonanie forka (fork) kodu

2. klonowanie repozytorium przez SSH (git clone SSH)

3. otwarcie projektu na urządzeniu w środowisku IDE dla języka Python

4. uruchomienie skryptu

Kod źródłowy:

```
import time

import sys

print('Welcome To fastrack USSD Banking Project...')

time.sleep(8)

bank_list="""
```

```
1. Access Bank
2. Fidelity Bank
3. Guarantee Trust Bank
4. Heritage Bank
5. Polaris Bank
6. Stanbic IBTC
7. Unity Bank
8. Wema Bank
"""
gen_bvn = " "
def BVN_checker( ):
global gen_bvn
bvn = [str(i) for i in range (5)]
gen_bvn= "".join(bvn)
def open_acct( ):
global gen_bvn
print("Welcome to our online Account opening services.")
print("loading...")
# creating an empty list to serve as a temporary place holder.
temp_storage= [ ]
f_name= input("Enter your first name:")
s_name= input ("Enter your second name:")
sex = input("Enter sex [M/F]:")
BVN_checker( )
temp_storage.append(f_name)
temp_storage.append(s_name)
temp_storage.append(sex)
temp_storage.append(gen_bvn)
details= " ".join(temp_storage)
split_details = details.split(" ")
#print(split_details)
```

```
print(split_details[0]+" "+split_details[1])
print(split_details[2])
print("Your bvn is :"+split_details[3])
print("1. Press # to go back to options menu\n2. Press * to exit")
bck=input(":")
if bck=='#':
options_menu( )
else:
 sys.exit( )
exit( )
def upgrade_migrate( ):
print("Welcome to our online Upgrade/Migration services.\n 1.
Ugrade\n 2. Migrate")
print("press # is go back to the Main Menu.")
prompt = input("Enter preferred Choice:")
if prompt=="1":
time.sleep(5)
print("Upgrading...")
exit( )
elif prompt == "2":
time.sleep(5)
print("Migrating...")
exit( )
elif prompt == "#":
options_menu( )
else:
sys.exit( )
def balance ( ):
print("ACCOUNT\tBALANCE\n CHECKER")
print("press # is go back to the Main Menu.")
pin=input("Enter your 4 digit pin:")
```

```
# isdigit( ) is used to check for digits within a str while the nested if is used
to make sure the user inputs 4 digits.
###```i am to put the pin trial in a while loop```###REMINDER!!!
if len(pin)!=4:
print("Make sure its a 4digit pin.")
time.sleep(5)
balance( )
else:
if pin.isdigit( ):
time.sleep(5)
print("Loading...")
exit( )
elif pin== "#":
options_menu( )
else:
time.sleep(15)
print("wrong pin")
sys.exit( )
def transf( ):
print("1. Transfer self\n2. Transfer others")
print("press # is go back to the Main Menu.")
trnsf=input(":")
if trnsf == "#" :
options_menu( )
elif trnsf == "1":
 time.sleep(5)
print("Sending...")
exit( )
elif trnsf=="2":
time.sleep(5)
num=int(input("Enter receivers mobile number:"))
```

```
print("Transferring to",num)
exit( )
else:
if trnsf.isdigit( )!= True:
time.sleep(5)
print("Not an option")
sys.exit( )
elif trnsf.isdigit( ) and len(trnsf)>2:
time.sleep( 5)
print("wrong password.")
sys.exit( )
else:
time.sleep(10)
print("An error has occurred")
sys.exit( )
def funds( ):
time.sleep(3)
print(bank_list)
bnk = input("Select receipients Bank:")
acc_num= input("Entet account number:")
print("Sending to",acc_num)
hash= input("1.Press # to go back to options menu\n2. Press * to go
exit.")
if hash == "#":
options_menu( )
elif hash == "*":
exit( )
else:
sys.exit( )
#----------------------------------------
--------
```

```
###i'm yet to catch an error for non -digit and more than one
digit###REMINDER!!!
#-#-----------------------------------------------------
# This is the function for options.
def options_menu( ) :
print("1. Open Account\n2. Upgrade/Migrate\n3. Balance\n4.
Transfer\n5. Funds")
select_options ={
'1':open_acct,
'2':upgrade_migrate,
'3': balance,
'4':transf,
'5':funds}
choice=input("Enter an option:")
if select_options.get(choice):
select_options[choice]()
else:
sys.exit()
# This is the function which prompts the user as to whether the user wishes to
continue or stop transaction.
def exit( ):
exit= input("Do you wish to make another transaction [Y/N] :")
if exit== "N":
sys.exit( )
elif exit == "#":
options_menu( )
else:
log_in( )
# This is the function for logging using the fast code
*919#
def log_in( ):
```

```
try:
a=0
while a<3:
a+=1
USSD=input("ENTER USSD:")
if(USSD !="*919#"):
print("please re-enter USSD ...")
else:
print("Welcome to our online services how may we help you")
options_menu( )
exit( )
else:
time.sleep(10)
print("checking discrepancies...")
time.sleep(5)
print("An error has occured.")
except:
sys.exit( )
log_in( )
```

26. Rozpakowywanie pliku

Funkcje rozpakowywania plików:

- Wczytanie pliku ZIP przeznaczonego do rozpakowania
- Skrypt wypakuje wszystkie pliki do folderu docelowego

Instrukcja rozpakowywania plików:

Krok 1:

Otwórz terminal

Krok 2:

Przejdź do katalogu, w którym znajduje się plik skryptu Python

Krok 3:

Uruchom polecenie: `python script.py` lub `python3 script.py`

Krok 4:

Usiądź wygodnie i zrelaksuj się. Pozwól skryptowi wykonać zadanie.

Wymagania

- zipfile

Kod źródłowy:

```
import zipfile
target = input(r"Enter file to be unzipped:  ")
handle = zipfile.ZipFile(target)
handle.extractall("./Unzip  file/Unzip files")
handle.close()
```

27. Skracacz adresów URL

```
from __future__  import with_statement
import contextlib
from urllib.parse import urlencode
from urllib import urlencode
from urllib.request import urlopen
from urllib2 import urlopen
import sys
def short_url(url):
request_url = ('http://tinyurl.com/api-create.php?'  +
urlencode({'url':url}))
with contextlib.closing(urlopen(request_url))  as response:
return response.read().decode('utf-8  ')
def main():
for url in map(short_url, sys.argv[1:]):
print(url)
if __name__ == '__main__':
main()
```

28. Konwerter wideo na audio w języku Python

```
from pytube import YouTube
import pytube
import os
def main():
video_url = input('Enter YouTube video URL: ')
if os.name == 'nt':
path = os.getcwd() + '\\'
else:
path = os.getcwd() + '/'
name = pytube.extract.video_id(video_url)
YouTube(video_url).streams.filter(only_audio=True).first().download(filename=name)
location = path + name + '.mp4'
renametomp3 = path + name + '.mp3'
if os.name == 'nt':
os.system('ren {0}{1}'. format(location, renametomp3))
else:
 os.system('mv {0}{1}'. format(location, renametomp3))
if __name__ == '__main__':
main()
```

29. Tłumacze głosowi

Zależności:

Google Translate

```python
pip install googletrans
```

pyttsx3

```python
pip install pyttsx3
```

pyaudio

```python
pip install pyaudio
```

speech recognition

```python
pip install SpeechRecognition
```

Kod źródłowy:

```
from googletrans import Translator
import pyttsx3
import speech_recognition  as sr
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice',voices[1].id)
def speak(audio):
engine.say(audio)
engine.runAndWait()
def takeCommand():
r = sr.Recognizer()
with sr.Microphone() as source:
print("Listening...")
r.pause_threshold = 1
audio = r.listen(source)
try:
print("Recognizing...")
query = r.recognize_google(audio, language='en-in')
print(f"AK47 Said:{query}\n")
except Exception as e:
print(e)
print("Say that again Please...")
```

speak("Say that again Please...")

return "None"

return query

def Translate():

speak("what I should Translate??")

sentence = takeCommand()

trans = Translator()

trans_sen = trans.translate(sentence,src='en',dest='ca')

print(trans_sen.text)

speak(trans_sen.text)

Translate()

30. Haszowanie haseł

Zmieniacz tapet w języku Python

Zależności:

Uzyskaj klucz API tutaj: [Unsplash](https://unsplash.com/developers)

Wget

```python

pip install wget

```

Dodaj klucz API w pliku Wallpapers.py:

```

access_key = '' # tutaj wpisz swój klucz API Unsplash

```

Uruchomienie:

```

python wallpapers.py

```

Kod źródłowy:

Get the wallpaper from the internet

Save it to a temp directory

```
# Set the wallpaper
# Automate the calls to this script
import os
import requests
import wget
import subprocess
import time
import ctypes
SPI_SETDESKWALLPAPER  = 20
def get_wallpaper():
access_key  = '' # add your unspash api key here
url = 'https://api.unsplash.com/photos/random?client_id='  + access_key
params = {
'query': 'HD wallpapers',
'orientation': 'landscape'
}
response = requests.get(url, params=params).json()
image_source = response['urls']['full']
image = wget.download(image_source,
'C:/Users/projects/wallpaper.jpg')  # add the path here
return image
def change_wallpaper():
wallpaper = get_wallpaper()
ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER,
0, "C:\\Users\\projects\\wallpaper.jpg"  , 0) # add the path here as well
def main():
try:
while True:
change_wallpaper()
time.sleep(10)
except KeyboardInterrupt:
```

```
print("\nHope you like this one! Quitting.")
except Exception as e:
pass
if __name__ == "__main__":
main()
```

31. Aplikacja pogodowa

```
# import all functions from the tkinter
from tkinter import *
from tkinter import messagebox
def tell_weather() :
import requests, json
api_key = "api_key"
base_url = "http://api.openweathermap.org/data/2.5/weather?"
city_name = city_field.get()
complete_url  = base_url + "appid =" + api_key + "&q =" + city_name
response = requests.get(complete_url)
x = response.json()
if x["cod"] != "404" :
y = x["main"]
current_temperature  = y["temp"]
current_pressure  = y["pressure"]
current_humidiy  = y["humidity"]
z = x["weather"]
weather_description  = z[0]["description"]
temp_field.insert(15,  str(current_temperature)  + " Kelvin")
atm_field.insert(10,  str(current_pressure)  + " hPa")
humid_field.insert(15,  str(current_humidiy)  + " %")
desc_field.insert(10,  str(weather_description)  )
else :
messagebox.showerror("Error",  "City Not Found \n"
```

```
"Please enter valid city name")
city_field.delete(0, END)
def clear_all() :
city_field.delete(0, END)
temp_field.delete(0, END)
atm_field.delete(0, END)
humid_field.delete(0, END)
desc_field.delete(0, END)
city_field.focus_set()
if __name__ == "__main__" :
root = Tk()
root.title("Weather Application")
# Set the background colour of GUI window
root.configure(background = "light blue")
# Set the configuration of GUI window
root.geometry("425x175")
# Create a Weather Gui Application label
headlabel = Label(root, text = "Weather Gui Application", fg = 'white',
bg = 'Black')
# Create a City name : label
label1 = Label(root, text = "City name : ", fg = 'white', bg = 'dark gray')
# Create a City name : label
label2 = Label(root, text = "Temperature :", fg = 'white', bg = 'dark
gray')
# Create a atm pressure : label
label3 = Label(root, text = "atm pressure :", fg = 'white', bg = 'dark
gray')
# Create a humidity : label
label4 = Label(root, text = "humidity :", fg = 'white', bg = 'dark gray')
# Create a description :label
label5 = Label(root, text = "description :", fg = 'white', bg = 'dark gray')
```

```
headlabel.grid(row = 0, column = 1)

label1.grid(row = 1, column = 0, sticky ="E")

label2.grid(row = 3, column = 0, sticky ="E")

label3.grid(row = 4, column = 0, sticky ="E")

label4.grid(row = 5, column = 0, sticky ="E")

label5.grid(row = 6, column = 0, sticky ="E")

city_field = Entry(root)

temp_field = Entry(root)

atm_field = Entry(root)

humid_field = Entry(root)

desc_field = Entry(root)

city_field.grid(row = 1, column = 1, ipadx ="100")

temp_field.grid(row = 3, column = 1, ipadx ="100")

atm_field.grid(row = 4, column = 1, ipadx ="100")

humid_field.grid(row = 5, column = 1, ipadx ="100")

desc_field.grid(row = 6, column = 1, ipadx ="100")

button1 = Button(root, text = "Submit", bg = "pink", fg = "black",

command = tell_weather)

button2 = Button(root, text = "Clear", bg = "pink" , fg = "black",

command = clear_all)

button1.grid(row = 2, column = 1)

button2.grid(row = 7, column = 1)

# Start the GUI

root.mainloop()
```

32. API do tworzenia podsumowań stron internetowych

Projekt ten ma na celu stworzenie modelu uczenia maszynowego generującego podsumowania stron internetowych na podstawie podanych adresów URL.

Pierwsze kroki

Poniższe instrukcje pozwolą Ci uruchomić kopię projektu na lokalnym komputerze w celach programistycznych i testowych.

Wymagania wstępne

Dystrybucja języka Python

```

Anaconda

```

Instalacja

Zainstaluj w systemie dystrybucję Pythona Anaconda i utwórz środowisko wirtualne o nazwie „app”.

```

python -m venv app

```

Aktywuj środowisko wirtualne

```

Linux/Mac: source app/bin/activate

Windows: app\Scripts\activate

```

Zaktualizuj narzędzie pip do najnowszej wersji

```

pip install --upgrade pip

```

Zainstaluj zależności, korzystając z pliku requirements

```

pip install -r requirements.txt

```

Uwaga: Przed uruchomieniem jakiegokolwiek polecenia należy zawsze aktywować środowisko wirtualne.

Uruchamianie

Uruchom aplikację (upewnij się, że podajesz poprawny adres istniejącej strony internetowej)

Przykłady poprawnych poleceń

```

python app.py simple --url https://facebook.com --sentence 1 --language

english

python app.py simple --url https://facebook.com
```

python app.py simple --url https://korapay.com

python app.py bulk --path ./csv/valid_websites.csv

```

### API

Oto pełna lista opcji poleceń:

```

Narzędzie wiersza poleceń do tworzenia podsumowań stron internetowych.

Oto typowe polecenia dla tej aplikacji.

argumenty pozycyjne:

action Musi przyjąć wartość „summarize”

argumenty opcjonalne:

-h, --help wyświetl ten komunikat pomocy i zakończ działanie

--website PATH adres URL strony internetowej do podsumowania

Wymagania:

utils==1.0.1

sumeval==0.2.2

tensorflow==2.3.0

wget==3.2

sumy==0.8.1

model==0.6.0

numpy==1.19.1

newspaper==0.1.0.7

nltk==3.5

gensim==3.8.3

```

Kod źródłowy:

#!/usr/bin/python

from utils.summarize import summarize

import csv

import shutil
```

```
import os
import textwrap
import logging
import argparse
import sys
def parse_args(argv):
parser = argparse.ArgumentParser(
formatter_class=argparse.RawDescriptionHelpFormatter,
description=textwrap.dedent('''\
A command line utility for website summarization.
-----------------------------------------------
These are common commands for this app.'''))
parser.add_argument(
'action',
help='This action should be summarize')
parser.add_argument(
'--url',
help='A link to the website url'
)
parser.add_argument(
'--sentence',
help='Argument to define number of sentence for the summary',
type=int,
default=2)
 parser.add_argument(
'--language',
help='Argument to define language of the summary',
default='English')
parser.add_argument(
'--path',
help='path to csv file')
```

```
return parser.parse_args(argv[1:])
def readCsv(path):
print('\n\n Processing Csv file \n\n')
sys.stdout.flush()
data = []
try:
with open(path, 'r') as userFile:
userFileReader = csv.reader(userFile)
for row in userFileReader:
data.append(row)
except:
with open(path, 'r', encoding="mbcs") as userFile:
userFileReader = csv.reader(userFile)
for row in userFileReader:
data.append(row)
 return data
def writeCsv(data, LANGUAGE, SENTENCES_COUNT):
print('\n\n Updating Csv file \n\n')
sys.stdout.flush()
with open('beneficiary.csv', 'w') as newFile:
newFileWriter = csv.writer(newFile)
length = len(data)
position = data[0].index('website')
for i in range(1, length):
if i == 1:
_data = data[0]
_data.append("summary")
newFileWriter.writerow(_data)
try:
__data = data[i]
summary = summarize(
```

```
(data[i][position]), LANGUAGE, SENTENCES_COUNT)
__data.append(summary)
newFileWriter.writerow(__data)
except:
print('\n\n Error Skipping line \n\n')
sys.stdout.flush()
def processCsv(path, LANGUAGE, SENTENCES_COUNT):
try:
print('\n\n Proessing Started \n\n')
sys.stdout.flush()
data = readCsv(path)
writeCsv(data, LANGUAGE, SENTENCES_COUNT)
except:
print('\n\n Invalid file in file path \n\n')
sys.stdout.flush()
def main(argv=sys.argv):
# Configure logging
logging.basicConfig(filename='applog.log',
filemode='w',
level=logging.INFO,
format='%(levelname)s:%(message)s')
args = parse_args(argv)
action = args.action
url = args.url
path = args.path
LANGUAGE = "english" if args.language is None else args.language
SENTENCES_COUNT = 2 if args.sentence is None else args.sentence
if action == 'bulk':
 if path is None:
print(
'\n\n Invalid Entry!, please Ensure you enter a valid file path \n\n')
```

```
sys.stdout.flush()
return
# guide against errors
try:
processCsv(path,  LANGUAGE, SENTENCES_COUNT)
except:
print(
'\n\n Invalid Entry!, please Ensure you enter a valid file path \n\n')
sys.stdout.flush()
print('Completed')
sys.stdout.flush()
if os.path.isfile('beneficiary.csv'):
return shutil.move('beneficiary.csv',  path)
return
if action == 'simple':
# guide against errors
try:
summarize(url, LANGUAGE, SENTENCES_COUNT)
except:
print(
'\n\n Invalid Entry!, please Ensure you enter a valid web link \n\n')
sys.stdout.flush()
 print('Completed')
sys.stdout.flush()
else:
print(
'\nAction command is not supported\n for help: run python3 app.py -
h'
)
sys.stdout.flush()
return
```

if __name__ == '__main__':

main()

33. Komentarze do web scrappingu

- Ten skrypt pobierze adres URL filmu z YouTube i wygeneruje plik CSV dla użytkowników i komentarzy.

Wymagania wstępne

- Wystarczy mieć zainstalowane selenium, które służy do automatyzacji.

- Uruchom poniższy skrypt, aby zainstalować selenium

- $ pip install selenium

Jak uruchomić skrypt

- Wystarczy podmienić adres URL własnego filmu z YouTube w pliku webscrapindcomment.py

- I uruchomić polecenie w tym samym katalogu

- python webscrapindcomment.py

Wymagania - selenium==3.141.0

Kod źródłowy:

```
# -*- coding: utf-8 -*-
from selenium import webdriver
import csv
import time
items=[]
driver=webdriver.Chrome(r"C:/Users/hp/Anaconda3/chromedriver.exe")
driver.get('https://www.youtube.com/watch?v=iFPMz36std4')
driver.execute_script('window.scrollTo(1, 500);')
#now wait let load the comments
time.sleep(5)
driver.execute_script('window.scrollTo(1, 3000);')
username_elems = driver.find_elements_by_xpath('//*[@id="author-
text"]')
comment_elems = driver.find_elements_by_xpath('//*[@id="content-
text"]')
```

```
for username, comment in zip(username_elems, comment_elems):
item = {}
item['Author'] = username.text
 item['Comment'] = comment.text
items.append(item)
filename = 'C:/Users/hp/Desktop/commentlist.csv'
with open(filename, 'w', newline='', encoding='utf-8') as f:
w = csv.DictWriter(f,['Author','Comment'])
w.writeheader()
for item in items:
w.writerow(item)
```

34. Blokada stron internetowych

Ten skrypt umożliwia blokowanie stron internetowych na komputerze poprzez edycję pliku hosts.

Sposób użycia

Najpierw dodaj adresy stron do zablokowania do tablicy w obu skryptach.

W systemie Linux: `sudo python website_blocker.py`

W systemie Windows uruchom skrypt jako administrator.

Aby odblokować strony, uruchom skrypt `website_unblocker.py`.

Kod źródłowy blokady stron:

```
import platform
if platform.system() == "Windows":
pathToHosts=r"C:\Windows\System32\drivers\etc\hosts"
elif platform.system() == "Linux":
pathToHosts=r"/etc/hosts"
redirect="127.0.0.1"
websites=["https://www.websitename.com"]
with open(pathToHosts,'r+') as file:
content=file.read()
for site in websites:
```

```
if site in content:
pass
else:
file.write(redirect+" "+site+"\n")
```

Kod źródłowy odblokowujący witrynę:

```
import platform
if platform.system() == "Windows":
 pathToHosts=r"C:\Windows\System32\drivers\etc\hosts"
elif platform.system() == "Linux":
pathToHosts=r"/etc/hosts"
websites=["https://www.websitename.com"]
with open(pathToHosts,'r+') as file:
content=file.readlines()
file.seek(0)
for line in content:
if not any(site in line for site in websites):
file.write(line)
file.truncate()
```

35. Bot na WhatsAppa

```
## Instrukcja obsługi
1. Wprowadź swoje dane
2. Połącz się z Internetem
3. Wyślij wiadomość
# Uruchamianie aplikacji
- Utwórz środowisko wirtualne
- Zainstaluj wymagane zależności
`pip install requirements.txt`
- Uruchom aplikację
```

`python main.py`

Kod źródłowy:

```
import pywhatkit
from datetime import datetime
now = datetime.now()
chour = now.strftime("%H")
mobile = input('Enter Mobile No of Receiver : ')
message = input('Enter Message you wanna send : ')
hour = int(chour) + int(input('Enter hour : '))
minute = int(input('Enter minute : '))
pywhatkit.sendwhatmsg(mobile,message,hour,minute)
```

36. Automatyzacja WhatsApp

Jak uruchomić ten skrypt w języku Python?

1. Zainstaluj [chromedriver]

(https://chromedriver.storage.googleapis.com/index.html?path=2.25/) (

wybierz odpowiednią wersję)

2. `pip install selenium`

3. Upewnij się, że podano **poprawną ścieżkę** do sterownika

Chrome (chromedriver)

4. Wpisz nazwę osoby, do której chcesz wysłać wiadomość,

dokładnie tak, jak jest ona zapisana.

5. Wpisz treść wiadomości, którą chcesz wysłać.

6. Będziesz mieć **15 sekund** na zeskanowanie kodu dla WhatsApp Web.

7. Wiadomość została wysłana.

Kod źródłowy:

```
# Selenium is required for automation
# sleep is required to have some time for scanning
from selenium import webdriver
from selenium.common.exceptions  import NoSuchElementException
from selenium.webdriver.support.ui  import WebDriverWait
```

```
from selenium.webdriver.support  import expected_conditions  as EC
from selenium.webdriver.common.keys  import Keys
from selenium.webdriver.common.by  import By
from time import sleep
def whatsapp(to, message):
person = [to]
string = message
chrome_driver_binary  = "C:\\Program
Files\\Google\\Chrome\\Application\\chromedriver.exe"
# Selenium chromedriver  path
driver = webdriver.Chrome(chrome_driver_binary)
driver.get("https://web.whatsapp.com/")
sleep(15)
# This will find the person we want to send the message to in the list
for name in person:
user =
driver.find_element_by_xpath("//span[@title='{}']".format(name))
user.click()
text_box = driver.find_element_by_xpath(
 '//*[@id="main"]/footer/div[1]/div[2]/div/div[2]')
try:
text_box.send_keys(string)
sendbutton = driver.find_elements_by_xpath(
'//*[@id="main"]/footer/div[1]/div[3]/button')[0]
sendbutton.click()
sleep(10)
print('Message Sent!!')
except:
print('Error occured....')
if __name__ == "__main__":
to = input('Who do you want to send a message to? Enter the name: ')
```

content = input("What message to you want to send? Enter the message: ")

whatsapp(to, content)

37. Instagram: Obserwujący – Nieobserwujący

Wysyłaj i planuj wiadomości na WhatsAppie za pomocą zaledwie siedmiu linii kodu w języku Python.

Wykorzystane moduły

- pywhatkit

pip install [requirements.txt]

Jak to działa

- Najpierw zaloguj się do wersji WhatsApp Web, skanując kod QR.

- Wystarczy podać odpowiednie dane: numer telefonu odbiorcy (wraz z kodem kraju), treść wiadomości oraz godzinę wysyłki (w formacie 24-godzinnym).

- O wyznaczonej porze skrypt otwiera WhatsApp Web w domyślnej przeglądarce i wysyła wiadomość na podany numer telefonu.

Wymagania – pywhatkit

Kod źródłowy:

"""

Automatyczny komunikator WhatsApp

- Wyślij wiadomość do znajomego lub grupy, używając zaledwie 7 linii kodu w języku Python

"""

import pywhatkit

phoneno = input("Podaj numer telefonu odbiorcy: ")

message = input("Wpisz treść wiadomości do wysłania: ")

print("Ustal czas wysłania wiadomości WhatsApp do odbiorcy: ")

Time_hrs = int(input("- Godzina: "))

Time_min = int(input("- Minuta: "))

pywhatkit.sendwhatmsg(phoneno, message, Time_hrs, Time_min)

Wskazówka: Jeśli chcesz wysłać lub zaplanować wiadomość do grupy na WhatsAppie,

użyj poniższego kodu i podaj odpowiednie wartości parametrów.

pywhatkit.sendwhatmsg_to_group(GroupID, message, time_hour,

time_min, wait_time)

Uwaga: Identyfikator grupy (Group ID) znajduje się w linku z zaproszeniem do tej grupy,

38. Skrypt do pobierania danych z infoboksów Wikipedii

- Niniejszy skrypt w języku Python wykorzystuje bibliotekę BeautifulSoup do pobierania danych ze stron Wikipedii na podstawie zapytania użytkownika i wyodrębniania informacji z tamtejszych infoboksów.

Wymagania:

```
$ pip install -r requirements.txt
```

Wymagania:

beautifulsoup4==4.9.3

certifi==2020.12.5

chardet==4.0.0

idna==2.10

requests==2.25.1

soupsieve==2.2.1

urllib3==1.26.4

Kod źródłowy:

```
from bs4 import BeautifulSoup
import requests
from tkinter import *
info_dict = {}
def error_box():
"""
A function to create a pop-up, in case the code errors out
"""
global mini_pop
mini_pop = Toplevel()
mini_pop.title('Error screen')
mini_l = Label(mini_pop, text=" !!!\nERROR FETCHING DATA",
fg='red', font=('Arial',10,'bold'))
```

```
mini_l.grid(row=1, column=1, sticky='nsew')
entry_str.set("")
def wikiScraper():
"""
Function scrapes the infobox lying under the right tags and displays
the data obtained from it in a new window
"""
global info_dict
# Modifying the user input to make it suitable for the URL
entry = entry_str.get()
entry = entry.split()
 query = '_'.join([i.capitalize() for i in entry])
req = requests.get('https://en.wikipedia.org/wiki/'+query)
# to check for valid URL
if req.status_code == 200:
# for parsing through the html text
soup = BeautifulSoup(req.text, 'html.parser')
# Finding text within infobox and storing it in a dictionary
info_table = soup.find('table', {'class': 'infobox'})
try:
for tr in info_table.find_all('tr'):
try:
if tr.find('th'):
info_dict[tr.find('th').text] = tr.find('td').text
except:
pass
except:
error_box()
# Creating a pop up window to show the results
global popup
popup = Toplevel()
```

```
 popup.title(query)
r = 1
for k, v in info_dict.items():
e1 = Label(popup, text=k+" : ", bg='cyan4', font=('Arial',10,'bold'))
e1.grid(row=r, column=1, sticky='nsew')
e2 = Label(popup, text=info_dict[k],  bg="cyan2", font=('Arial',10,
'bold'))
e2.grid(row=r, column=2, sticky='nsew')
r += 1
e3 = Label(popup, text='', font=('Arial',10,'bold'))
e3.grid(row=r, sticky='s')
r += 1
entry_str.set("")
info_dict = {}
else:
print('Invalid URL')
error_box()
# Creating a window to take user search queries
root = Tk()
root.title('Wikipedia  Infobox')
global entry_str
entry_str = StringVar()
search_label = LabelFrame(root,  text="Search: ", font = ('Century
Schoolbook L',17))
search_label.pack(pady=10, padx=10)
user_entry = Entry(search_label,  textvariable = entry_str, font = ('Century
Schoolbook L',17))
user_entry.pack(pady=10,  padx=10)
button_frame = Frame(root)
button_frame.pack(pady=10)
submit_bt = Button(button_frame, text = 'Submit', command = wikiScraper,
```

```
font = ('Century Schoolbook L',17))
submit_bt.grid(row=0, column=0)
root.mainloop()
```

39. Skraper Wikipedii w języku Python

```
import wikipedia as wiki
print(wiki.search("Python"))
print(wiki.suggest("Pyth"))
print(wiki.summary("Python"))
wiki.set_lang("fr")
print(wiki.summary("Python"))
wiki.set_lang("en")
p = wiki.page("Python")
#To get the Title
print(p.title)
#To get the url of the article
print(p.url)
#To scrape the full article
print(p.content)
#To get all the images in the article
print(p.images)
#And to get all the referals used by Wikipedia in the article
print(p.links)
```

40. Pobieranie obrazu z Instagrama

```
# Obrazy chmury słów dla artykułu w Wikipedii
Skrypt Pythona, który prosi użytkownika o podanie danych, wyszukuje
odpowiedni artykuł w Wikipedii i generuje chmurę słów na podstawie wyszukanego artykułu.
### Wymagania wstępne
Zainstaluj modele `pip install` w pliku `requirements.txt` z poziomu wiersza poleceń.
### Jak uruchomić skrypt
```

Uruchom jak każdy inny plik Pythona. Po uruchomieniu obraz chmury słów zostanie zapisany w bieżącym katalogu. Skrypt wyświetli również prośbę o potwierdzenie (t/n), jeśli użytkownik chce zobaczyć wygenerowany obraz podczas wykonywania.

Wymaganie:

```
beautifulsoup4==4.9.1
certifi==2020.6.20
chardet==3.0.4
cycler==0.10.0
idna==2.10
kiwisolver==1.2.0
matplotlib==3.3.1
numpy==1.19.1
Pillow==7.2.0
pyparsing==2.4.7
python-dateutil==2.8.1
requests==2.24.0
six==1.15.0
soupsieve==2.0.1
urllib3==1.25.10
wikipedia==1.4.0
wordcloud==1.8.0
```

Kod źródłowy:

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
import wikipedia
import sys
import warnings
# supressing unnecessary warnings
warnings.filterwarnings("ignore")
# function to search the wikipedia article and generate the wordcloud
def gen_cloud(topic):
```

```
try:
content = str(wikipedia.page(topic).content)
except:
print("Error, try searching something else...")
sys.exit()
STOPWORDS.add('==')
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, max_words=200,
background_color="black", width=600, height=350).generate(content)
return wordcloud
# function to save the wordcloud to current directory
def save_cloud(wordcloud):
wordcloud.to_file("./wordcloud.png")
# function to display the wordcloud with matplotlib
def show_cloud(wordcloud):
 plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
# driver code
if __name__ == '__main__':
topic = input("What do you want to search: ").strip()
wordcloud = gen_cloud(topic)
save_cloud(wordcloud)
print("Wordcloud saved to current directory as wordcloud.png")
desc = input("Do you wish to see the output(y/n): ")
if desc == 'y':
show_cloud(wordcloud)
sys.exit()
```

41. Skrypt generujący podsumowania z Wikipedii (z interfejsem graficznym)

Uruchomienie tego skryptu otwiera interfejs graficzny (GUI) służący do generowania podsumowań z Wikipedii na dowolny, wybrany przez użytkownika temat.

Instrukcja konfiguracji

Aby uruchomić ten skrypt, należy mieć zainstalowane w systemie środowisko Python oraz menedżer pakietów pip. Po ich zainstalowaniu otwórz terminal w folderze projektu i wykonaj poniższe polecenie, aby zainstalować wymagane zależności:

```
pip install -r requirements.txt
```

Po spełnieniu wszystkich wymagań projektu otwórz terminal w folderze projektu i uruchom skrypt poleceniem:

```
python summary.py
```

lub

```
python3 summary.py
```

(w zależności od wersji Pythona). Upewnij się, że uruchamiasz polecenie w tym samym środowisku wirtualnym, w którym zainstalowano wymagane moduły.

Wymagania:

Pymediawiki

Kod źródłowy:

```
from tkinter import Tk, Frame, Toplevel, Entry, Button, Text, Scrollbar,
END, INSERT
from tkinter.messagebox import showerror
from mediawiki import MediaWiki
wikipedia = MediaWiki()
# Function to get summary using wikipedia module and display it
def get_summary():
try:
# clear text area
answer.delete(1.0, END)
```

```
# show summary in text area
topic = keyword_entry.get()
p = wikipedia.page(topic)
answer.insert(INSERT, p.summary)
except Exception as error:
showerror("Error", error)
# create a GUI window and configure it
root = Tk()
root.title("Wikipedia Summary")
root.geometry("770x650")
root.resizable(False, False)
root.configure(bg="dark grey")
# create a frame for entry and button
top_frame = Frame(root, bg="dark grey")
top_frame.pack(side="top", fill="x", padx=50, pady=10)
# create a frame for text area where summary will be displayed
bottom_frame = Frame(root, bg="dark grey")
bottom_frame.pack(side="top", fill="x", padx=10, pady=10)
# create a entry box where user can enter a keyword
keyword_entry = Entry(top_frame, font=("Arial", 20, "bold"), width=25,
bd=4)
keyword_entry.pack(side="left", ipady=6)
# create a search button
search_button = Button(top_frame, text="Get Summary", font=(
"Arial", 16, "bold"), width=15, bd=4, command=get_summary)
search_button.pack(side="right")
# create a scroll bar for text area
scroll = Scrollbar(bottom_frame)
# create a text area where summary will be displayed
answer = Text(bottom_frame, font=("Arial", 18), fg="black",
width=55, height=20, bd=5, yscrollcommand=scroll.set)
```

```
answer.pack(side="left", fill="y")
scroll.pack(side="left", fill="y")
# start the GUI
root.mainloop()
```

42. Gry słowne

```
import sys
list1=['a','b','c','d','e','f','g','h','i','j','k','l','m']
list2=['n','o','p','q','r','s','t','u','v','w','x','y','z']
w=input("Enter a word ")
prepartner=prepartner1=[]
postpartner1=postpartner=[]
for i in w:
if(i in list1):
prepartner.append(i)
if(i in list2):
postpartner.append(i)
for j in prepartner:
list1index=list1.index(j)
if(list2[list1index] in postpartner):#testing if all prepartners has
postpartners
pass
else:
print("YOU LOST")
sys.exit()
prepartner1=prepartner
postpartner1=postpartner
for k in prepartner:
 x=prepartner.index(k)
y=postpartner.index(list2[list1.index(k)])
```

if(w.index(prepartner[x])<w.index(postpartner[y])):

if(w.index(postpartner[y])-w.index(prepartner[x])==1):#testing3a

prepartner1.pop(x)

postpartner1.pop(y)

else:

print("YOU LOST")

sys.exit()

postpartner1.reverse()

count=0

for l in prepartner1:

if(prepartner1.index(l)==postpartner1.index(list2[list1.index(l)])):#testing3b

count+=1

if(count==len(prepartner1)):

print("GAME WON")

else:

print("GAME LOST")

43. Automatyzacja konfiguracji stanowiska pracy

Zależności:

pyinstaller

```

pip install pyinstaller

```

TUTAJ PODAJ ŚCIEŻKĘ DO EDYTORA TEKSTU LUB ŚRODOWISKA IDE:

```

codePath = "C:\\Program Files\\Sublime Text 3\\sublime_text.exe" # TUTAJ PODAJ

ŚCIEŻKĘ DO EDYTORA TEKSTU LUB ŚRODOWISKA IDE

```

Uruchomienie:

Konwersja pliku Python na format .exe

```
```

pyinstaller -F workstation.py

```

# Konfiguracja autostartu

1. Naciśnij klawisze WINDOWS + R, aby otworzyć okno „Uruchamianie”

2. Wpisz

```

shell:startup

```

3. Skopiuj plik .exe do folderu autostartu

4. Uruchom ponownie system – plik .exe powinien uruchomić się zaraz po starcie systemu

Kod źródłowy:

```
import os
import webbrowser as wb
def workstation():
codePath = "C:\\Program Files\\Sublime Text
3\\sublime_text.exe"#ADD THE PATH OF TXET EDITOR OR IDE HERE
os.startfile(codePath)
chrome_path = 'C:/Program Files
(x86)/Google/Chrome/Application/chrome.exe %s'#ADD THE PATH OF
CHROME HERE
URLS = (
"stackoverflow.com",
"github.com/ ",
"gmail.com",
"google.com",
"youtube.com"
)#ADD THE WEBSITES YOU USE WHIE WORKING
for url in URLS:
wb.get(chrome_path).open(url)
workstation()
```
```

44. Ustaw losowe tło pulpitu

Ten skrypt pobierze losowy obraz z serwisu [unsplash]

(https://source.unsplash.com/random) i ustawi go jako tło pulpitu.

Obraz zostanie zapisany pod nazwą „random.jpg”; upewnij się, że w bieżącym katalogu nie ma już pliku o takiej nazwie.

Wymagania

Linux

Zainstaluj [Nitrogen](https://wiki.archlinux.org/index.php/Nitrogen)

```
pip install requests
```

Użycie

```python
python background_linux.py
```

LUB

```python
python background_windows.py
```

Kod źródłowy:

```
from requests import get
import os
import ctypes
import sys
url = "https://source.unsplash.com/random"
file_name = "random.jpg"
def is_64bit():
return sys.maxsize > 2 ** 32
def download(url, file_name):
'''
downloading the file and saving it
'''
```

```
with open(file_name, "wb") as file:
response = get(url)
file.write(response.content)
def setup(pathtofile,version):
name_of_file = pathtofile
path_to_file = os.path.join(os.getcwd(), name_of_file)
SPI_SETDESKWALLPAPER = 20
if is_64bit():
ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER,
0, path_to_file, 0)
else:
ctypes.windll.user32.SystemParametersInfoA(SPI_SETDESKWALLPAPER,
0, path_to_file, 0)
if __name__ == "__main__":
download(url, file_name)
setup(file_name)
except Exception as e:
print(f"Error {e}")
raise NotImplementedError
```

45. Kompresja folderu i plików

```
### użycie
python zipfiles.py nazwa_pliku (lub nazwa_folderu)
przykład:
python zipfiles.py test.txt
python zipfiles.py ./test (folder)
```

Po uruchomieniu programu zostanie utworzony skompresowany plik („nazwa_pliku.zip”)

Kod źródłowy:

```
import zipfile
import sys
import os
```

```
# compress file function
def zip_file(file_path):
compress_file = zipfile.ZipFile(file_path + '.zip', 'w')
compress_file.write(path, compress_type=zipfile.ZIP_DEFLATED)
compress_file.close()
# Declare the function to return all file paths of the particular directory
def retrieve_file_paths(dir_name):
# setup file paths variable
file_paths = []
# Read all directory, subdirectories and file lists
for root, directories, files in os.walk(dir_name):
for filename in files:
# Create the full file path by using os module.
file_path = os.path.join(root, filename)
file_paths.append(file_path)
# return all paths
return file_paths
def zip_dir(dir_path, file_paths):
# write files and folders to a zipfile
compress_dir = zipfile.ZipFile(dir_path + '.zip', 'w')
with compress_dir:
# write each file separately
for file in file_paths:
compress_dir.write(file)
if __name__ == "__main__":
path = sys.argv[1]
if os.path.isdir(path):
files_path = retrieve_file_paths(path)
# print the list of files to be zipped
print('The following list of files will be zipped:')
for file_name in files_path:
```

print(file_name)

zip_dir(path, files_path)

elif os.path.isfile(path):

print('The %s will be zipped:' % path)

zip_file(path)

else:

print('a special file(socket,FIFO,device file), please input file or dir')

46. Porządkowanie plików w katalogu

Skrypt porządkujący pliki w katalogu (w porządku alfabetycznym)

Sposób użycia

```python

python main.py

```

Zostanie utworzony folder, a pliki zostaną odpowiednio przeniesione.

Kod źródłowy:

'''

This script will sort and move the files in the directory

(the alphabetical order).

'apple.txt' --> 'A'

'ryan.txt' --> 'R'

'01010.txt' --> 'Misc'

'''

import os

import shutil

filenames = []

def getfoldername(filename):

'''

'Test.txt' --> 't'

'010.txt' --> 'misc'

'zebra.txt' --> 'z'

```
'Alpha@@.txt' --> 'a'
'!@#.txt' --> 'misc'
'''
if filename[0].isalpha():
return filename[0].lower()
else:
 return 'misc'
def readdirectory():
'''
read the filename in the current directory and append them to a list
'''
global filenames
for files in os.listdir(os.getcwd()):
if os.path.isfile(os.path.join(os.getcwd(), files)):
filenames.append(files)
filenames.remove('main.py')  # removing script from the file list
# getting the first letters of the file & creating a file in the current_dir
def createfolder():
'''
creating a folders
'''
global filenames
for f in filenames:
if os.path.isdir(getfoldername(f)):
print("folder already created")
else:
os.mkdir(getfoldername(f))
 print('creating folder...')
# moving the file into the proper folder
def movetofolder():
'''
```

movetofolder('zebra.py','z')

'zebra.py'(moved to) 'z'

'''

global filenames

for i in filenames:

filename = i

file = getfoldername(i)

source = os.path.join(os.getcwd(), filename)

destination = os.path.join(os.getcwd(), file)

print(f"moving {source} to {destination}")

shutil.move(source, destination)

if __name__ == '__main__':

readdirectory()

createfolder()

movetofolder()

47. Skrypt do pobierania danych o kontaktach z LinkedIn

To skrypt stworzony przy użyciu bibliotek Selenium i Pandas, służący do pobierania listy kontaktów z LinkedIn wraz z informacjami o umiejętnościach (opcjonalnie). Wystarczy jedno polecenie, aby skrypt przygotował plik CSV z potrzebnymi danymi.

Instalacja

Upewnij się, że masz zainstalowane następujące biblioteki Pythona:

> pip3 install selenium pandas

Pozostałe wymagane elementy to standardowe moduły Pythona.

Następnie umieść plik `ChromeDriver.exe` w tym samym katalogu, w którym znajduje się skrypt. Możesz go pobrać [stąd]

(https://sites.google.com/a/chromium.org/chromedriver/downloads)

(Uwaga: Pobierz wersję zgodną z wersją Twojej przeglądarki Chrome).

Użycie

Podstawowe użycie:

> python scrapper.py -e \<email\> -p \<hasło\>

Pobieranie danych o umiejętnościach:

> python scrapper.py -e \<email\> -p \<hasło\> -s

Dodatkowe uwagi

- Czas działania skryptu zależy od liczby kontaktów na koncie. W przypadku podstawowego użycia złożoność czasowa może wynosić O(n^2).

- W przypadku pobierania umiejętności czas ten dodatkowo wzrośnie w zależności od zawartości poszczególnych profili.

- Skrypt wyświetla komunikaty informujące o aktualnym etapie działania.

- Na wydajność wpływa również szybkość połączenia internetowego.

Wynik działania

Podstawowe użycie skutkuje utworzeniem pliku „scrap.csv" zawierającego kolumny: Imię i nazwisko (Name), Nagłówek (Headline) oraz Link. Kolumna z umiejętnościami (Skills) również powstanie, ale pozostanie pusta.

Użycie trybu pobierania umiejętności spowoduje uzupełnienie tej kolumny danymi z profili; poszczególne umiejętności będą oddzielone ciągiem znaków „ -- ".

Kod źródłowy:

```
# Linkedin My_Connections Scrapper

from selenium.webdriver.common.action_chains import ActionChains

from optparse import OptionParser

from selenium import webdriver

import pandas as pd

import time

import sys

import re

pattern_name = "\\n(.+)\\n" # Used to extract names

pattern_headline = 'occupation\\n(.+)\\n' # Used to extract headlines

# Help menu

usage = """

<Script> [Options]

[Options]

-h, --help Show this help message and exit.

-e, --email Enter login email

-p, --password Enter login password

-s, --skills Flag to scrap each profile, and look at its skill set
```

```
Operation Modes:
> Basic mode
This will scrap all LinkedIn connections list with there corresponding
Name, Headline, and Profile link.
> Skills scrapper mode (-s/--skills)
(Time Consuming mode)
This will do the same job of basic mode but along with visiting each
profile and extracting the skills of each.
"""
# Load args
parser = OptionParser()
parser.add_option("-e", "--email", dest="email", help="Enter login email")
parser.add_option("-p", "--password", dest="password",
help="Enter login password")
parser.add_option("-s", "--skills", action="store_true", dest="skills",
help="Flag to scrap each profile, and look at its skill set")
def login(email, password):
"""LinkedIn automated login function"""
# Get LinkedIn login page
driver = webdriver.Chrome("chromedriver.exe")
driver.get("https://www.linkedin.com")
# Locate Username field and fill it
session_key = driver.find_element_by_name("session_key")
session_key.send_keys(email)
# Locate Password field and fill it
session_password =
driver.find_element_by_name("session_password")
session_password.send_keys(password)
# Locate Submit button and click it
submit = driver.find_element_by_class_name("sign-in-
form__submit-button")
```

```
submit.click()
# Check credentials output
if driver.title != "LinkedIn":
print("Provided E-mail/Password is wrong!")
driver.quit()
sys.exit()
# Return session
return driver
def scrap_basic(driver):
"""Returns 3 lists of Names, Headlines, and Profile Links"""
driver.get("https://www.linkedin.com/mynetwork/invite-
connect/connections/")
# Bypassing Ajax Call through scrolling the page up and down multiple
times
# Base case is when the height of the scroll bar is constant after 2 complete
scrolls
time_to_wait = 3 # Best interval for a 512KB/Sec download speed -
Change it according to your internet speed
last_height = driver.execute_script("return document.body.scrollHeight")
while True:
# Scroll down to bottom
driver.execute_script(
"window.scrollTo(0, document.body.scrollHeight);")
# This loop is for bypassing a small bug upon scrolling that causes the
Ajax call to be cancelled
for i in range(2):
time.sleep(time_to_wait)
driver.execute_script("window.scrollTo(0, 0);") # Scroll up to top
time.sleep(time_to_wait)
# Scroll down to bottom
driver.execute_script(
```

```
"window.scrollTo(0, document.body.scrollHeight);")
new_height = driver.execute_script(
"return document.body.scrollHeight") # Update scroll bar height
if new_height == last_height:
break
last_height = new_height
# Extract card without links
extracted_scrap = driver.find_elements_by_class_name(
"mn-connection-card__details")
extracted_scrap = [_.text for _ in extracted_scrap]
# Append data to a seperate list
names = []
headlines = []
for card in extracted_scrap:
# Try statements just in case of headline/name type errors
 try:
names.append(re.search(pattern_name, card)[0])
except:
names.append(" ")
try:
headlines.append(re.search(pattern_headline, card)[0])
except:
headlines.append(" ")
# Extract links
extracted_scrap = driver.find_elements_by_tag_name('a')
links = []
for i in extracted_scrap:
link = i.get_attribute("href")
if "https://www.linkedin.com/in" in link and not link in links:
links.append(link)
# Return outputs
```

```
return driver, names, headlines, links
def scrap_skills(driver, links):
skill_set = []
length = len(links)
for i in range(length):
link = links[i] # Get profile link
driver.get(link)
# Bypassing Ajax Call through scrolling through profile multiple
sections
time_to_wait = 3
last_height = driver.execute_script(
"return document.body.scrollHeight")
while True:
# Scroll down to bottom
driver.execute_script(
"window.scrollTo(0, document.body.scrollHeight);")
# This loop is for bypassing a small bug upon scrolling that causes the
Ajax call to be cancelled
for i in range(2):
time.sleep(time_to_wait)
driver.execute_script(
"window.scrollTo(0, document.body.scrollHeight/4);")
driver.execute_script(
"window.scrollTo(0, document.body.scrollHeight/3);")
driver.execute_script(
"window.scrollTo(0, document.body.scrollHeight/2);")
driver.execute_script(
"window.scrollTo(0, document.body.scrollHeight*3/4);")
time.sleep(time_to_wait)
 # Scroll down to bottom
driver.execute_script(
```

```
"window.scrollTo(0, document.body.scrollHeight);")
new_height = driver.execute_script(
"return document.body.scrollHeight") # Update scroll bar height
if new_height == last_height:
break
last_height = new_height
# Locate button
buttons = driver.find_elements_by_tag_name('button')
length = len(buttons)
for button_num in range(length):
i = buttons[button_num].get_attribute("data-control-name")
if i == "skill_details":
button = buttons[button_num]
break
# Scroll then click the button
actions = ActionChains(driver)
actions.move_to_element(button).click().perform()
# Finally extract the skills
skills = driver.find_elements_by_xpath(
"//*[starts-with(@class,'pv-skill-category-entity__name-text')]")
skill_set_list = []
 for skill in skills:
skill_set_list.append(skill.text)
# Append each skill set to its corresponding name
# Appending all to one string
skill_set.append(" -- ".join(skill_set_list))
# Return session & skills
return driver, skill_set
def save_to_csv(names, headlines, links, skills):
# If skills argument was false
if skills is None:
```

```
skills = [None]*len(names)
# Make a dataframe and append data to it
df = pd.DataFrame()
for i in range(len(names)):
df = df.append({"Name": names[i], "Headline": headlines[i],
"Link": links[i], "Skills": skills[i]}, ignore_index=True)
# Save to CSV
df.to_csv("scrap.csv", index=False, columns=[
"Name", "Headline", "Link", "Skills"])
# Start checkpoint
if __name__ == "__main__":
 (options, args) = parser.parse_args()
# Inputs
email = options.email
password = options.password
skills = options.skills
driver = login(email, password) # Login Phase
print("Successfull Login!")
print("Commencing 'My-Connections' list scrap...")
driver, names, headlines, links = scrap_basic(driver) # Basic Scrap Phase
print("Finished basic scrap, scrapped {}".format(len(names)))
if skills:
print("Commencing 'Skills' scrap...")
driver, skill_set = scrap_skills(driver, links) # Skills Scrap Phase
print("Finished Skills scrap.")
print("Saving to CSV file...")
save_to_csv(names, headlines, links, skill_set) # Save to CSV
else:
save_to_csv(names, headlines, links, None) # Save to CSV
print("Scrapping session has ended.")
# End Session
```

driver.quit()

48. Narzędzie do pobierania filmów z YouTube

Narzędzie do pobierania filmów z YouTube

Celem tego projektu jest umożliwienie szybkiego i łatwego pobierania dowolnego filmu z serwisu YouTube na urządzenie użytkownika. Użytkownik musi skopiować adres URL filmu, który chce pobrać, wkleić go w polu „wklej tutaj link" i kliknąć przycisk pobierania, aby rozpocząć proces. Po zakończeniu pobierania w oknie, poniżej przycisku, pojawi się komunikat informujący o sukcesie operacji.

Wymagania wstępne

Do realizacji projektu wykorzystujemy podstawy języka Python oraz biblioteki tkinter i pytube.

- **Tkinter** to standardowa biblioteka GUI (graficznego interfejsu użytkownika), stanowiąca jeden z najprostszych sposobów tworzenia aplikacji z interfejsem graficznym.

- **pytube** służy do pobierania filmów z serwisu YouTube.

Aby zainstalować wymagane moduły, należy wykonać poniższe polecenia w wierszu poleceń:

```

pip install tkinter

pip install pytube

```

Oto kolejne kroki tworzenia aplikacji:

Krok 1: Importowanie bibliotek

Rozpocznij projekt od zaimportowania wymaganych modułów.

W tej implementacji skryptu importujemy moduły Tkinter oraz pytube.

Krok 2: Tworzenie okna aplikacji

- **Tk()** – służy do inicjalizacji biblioteki tkinter i utworzenia okna aplikacji.

- **geometry()** – służy do ustawienia szerokości i wysokości okna.

- **resizable(0,0)** – blokuje możliwość zmiany rozmiaru okna (ustawia stały rozmiar).

- **title()** – służy do nadania oknu tytułu.

- **Label()** – widżet służący do wyświetlania tekstu, którego użytkownik nie może edytować.

- **root** – nazwa obiektu okna.

- **text** – parametr określający treść wyświetlaną w etykiecie (label).

- **font** – parametr określający czcionkę tekstu.

- **pack** – metoda rozmieszczająca widżet w układzie blokowym.

Krok 3: Tworzenie pola do wprowadzania linku

- **link** – zmienna typu ciąg znaków (string) przechowująca adres URL filmu z YouTube wprowadzony przez użytkownika.

- **Entry()** – widżet służący do tworzenia pola tekstowego umożliwiającego wprowadzanie danych. - **width** określa szerokość widżetu pola tekstowego (entry)

- **textvariable** służy do powiązania zmiennej tekstowej z widżetem pola tekstowego

- **place()** służy do umieszczenia widżetu w określonym miejscu

Krok 4: Utworzenie funkcji rozpoczynającej pobieranie

Zmienna `url` pobiera link do filmu na YouTube ze zmiennej przechowującej adres za pomocą metody **get()**, a następnie funkcja **str()** konwertuje ten link na typ tekstowy (string).

Film jest pobierany w ramach pierwszego dostępnego strumienia za pomocą metody **stream.first()**.

Widżet **Button()** służy do wyświetlenia przycisku w oknie.

- **text** – tekst wyświetlany na przycisku

- **font** – czcionka użyta do wyświetlenia tekstu

- **bg** – kolor tła

- **command** – funkcja wywoływana po kliknięciu przycisku

root.mainloop() to metoda uruchamiana w celu rozpoczęcia działania programu.

Wynik

Po uruchomieniu skryptu zobaczysz następujący efekt:

Kod źródłowy:

```
from tkinter import *

from pytube import YouTube

root = Tk()

root.geometry('700x300')

root.resizable(0, 0)

root.title("YouTube Video Downloader")

Label(root, text='Copy the link of the video you want to download from

YouTube',

font='arial 15 bold').pack()

# enter link
```

```
link = StringVar()
Label(root, text='Paste Link Here:', font='arial 15 bold').place(x=270,
y=60)
Entry(root, width=80, textvariable=link).place(x=32, y=90)
# function to download video
def Downloader():
url = YouTube(str(link.get()))
video = url.streams.first()
video.download()
Label(root, text='DOWNLOADED', font='arial 15').place(x=270,
y=210)
Button(root, text='DOWNLOAD', font='arial 15 bold', bg='white',
padx=2, command=Downloader).place(x=280, y=150)
root.mainloop()
```