

Politechnika Wrocławska
Wydział Elektroniki

ZALICZENIE TI 2021/2022, GRUPA SR/P
11-13

Przygotował:
ADAM ORCZYKOWSKI

28 stycznia 2022

Spis treści

Wstęp	2
1 LAB 1	4
2 LAB 2	5
2.1 Zapoznanie ze środowiskiem GIT	5
2.2 Tworzenie środowiska GIT	5
2.2.1 Tworzenie nowego repozytorium	5
2.2.2 Dodawanie zmian	7
2.2.3 Pobieranie zmian	7
2.2.4 Sprawdzanie zmian pomiędzy dwoma wersjami	8
3 LAB 3	9
3.1 Wykonywanie i wyświetlanie prostych obliczeń	9
3.2 Instrukcje warunkowe	9
3.3 Pętle	10
4 LAB 4	11
4.1 Biblioteka "Numpy"	11
4.1.1 Tworzenie macierzy	11
4.1.2 Mnożenie macierzy	11
4.1.3 Obliczanie macierzy odwrotnej	12
4.1.4 Obliczanie wyznacznika macierzy	12
4.2 Biblioteka "Matplotlib"	12
5 LAB 5	14
5.1 Konfigurowanie wyglądu wykresu	14
5.2 Inne typy wykresów	14
5.2.1 Wykres słupkowy	15
5.2.2 Wykres kołowy	15
6 Zadania	17
6.1 Stwórz wykres napięcia ładowania i rozładowania kondensatora	17
6.1.1 Ładowanie kondensatora	17
6.1.2 Rozładowanie kondensatora	18
6.2 Zadanie ze studentami	18
6.2.1 Pobieranie i zapisywanie danych do pliku	19
6.2.2 Pobieranie i tworzenie wykresu danych	20

7	LaTeX	23
7.1	Tworzenie ogólnego zarysu dokumentu	23
7.1.1	Definiowanie rodzaju	23
7.1.2	Tworzenie spisu treści	25
7.1.3	Tworzenie rozdziałów, sekcji oraz podsekcji	25
7.2	Wykorzystywanie funkcji LaTeX'a	25
7.2.1	Wyliczenia	25
7.2.2	Wzory	25
7.2.3	Grafiki	27
7.2.4	Kody źródłowe	27
7.2.5	Bibliografie	27

Spis rysunków

2.1	Przejsście do wybranego katalogu	6
2.2	Ustalenie informacji autora	6
2.3	Tworzenie repozytorium oraz serwera	7
2.4	Dodawanie i wysyłanie pliku	7
2.5	Pobieranie wybranej przez nas wersji pliku	8
2.6	Sprawdzanie zmian pomiędzy plikami	8
5.1	Wynik działania programu - wykres słupkowy	15
5.2	Wynik działania programu - wykres kołowy	16
6.1	Wynik działania programu - tabela ze wczytanego pliku, wyniki operacji w programie oraz tabela wczytana do pliku	20
6.2	Wykres przedstawiający ile studentów uzyskało daną ocenę	22
6.3	Wartości odpowiadające liczbie studentów, którzy uzyskali daną ocenę .	22
7.1	Przykładowe rozszerzenia oraz przykładowe definiowanie dokumentu . .	24
7.2	Wszystkie podane poniżej komendy, w celu poprawnego działania należy wpisać po tej komendzie	24
7.3	Komenda, która tworzy spis treści w naszym dokumencie	24
7.4	Komenda, która dzieli nasz dokument, na wybrane przez nas części . .	25
7.5	Wyliczanie, poprzez numerowanie kolejnych pozycji	26
7.6	Wyliczanie, poprzez wykropkowanie kolejnych pozycji	26
7.7	Przykłady zawierania wzoru w tekście	26
7.8	Zawieranie grafiki w dokumencie	27
7.9	Dodawanie kodu źródłowego	27
7.10	Przykładowy szablon w pliku .bib	27
7.11	Odwołanie się do artykułu o nazwie "greenwade93"	28
7.12	Wyświetlenie bibliografii w dokumencie	28

Rozdział 1

LAB 1

Począs pierwszych labolatoriów przedstawione zostały nam zasady, jakie panują w pracowni, nasze obowiązki względem znajdujących się w labolatorium urządzeń oraz zasady zaliczania. Ponadto przedstawione zostały nam również zasady BHP, które obowiązywały na terenie pracowni. Dowiedzieliśmy się również, jakich programów będziemy się uczyli na owych labolatoriach.

Rozdział 2

LAB 2

2.1 Zapoznanie ze środowiskiem GIT

Podczas drugich labolatoriów omawialiśmy środowisko GIT. Jest to system kontroli wersji, który pozwala nam zarządzać naszą pracą w różnych programach komputerowych. System ten śledzi wszystkie wykonywane zmiany w plikach, a także umożliwia przywrócenie ich dowolnej, wcześniejszej wersji. Program ten ma wiele zalet gdyż nie musimy się martwić tym, że plik może nam gdzieś zginąć, lub się uszkodzić, ponieważ w każdej chwili możemy pobrać ostatnią wersję z serwera. Pomaga nam również w sytuacji, gdy stwierdzimy się, że w pliku nad którym pracowaliśmy od dłuższego czasu jest błąd i potrzebujemy się cofnąć kilka wersji wstecz, by zacząć od nowa pisać dany fragment.

2.2 Tworzenie środowiska GIT

Pierwszym naszym krokiem w celu zrozumienia na jakich zasadach opiera się środowisko GIT, było ukończenie samouczka na stronie Katacoda[3]. Omówone tam zostały następujące funkcje programu:

1. Tworzenie nowego repozytorium;
2. Dodawanie zmian;
3. Pobieranie zmian;
4. Sprawdzanie, jakie zaszły zmiany pomiędzy dwoma wersjami;

Poniżej zostały omówione główne funkcje środowiska GIT w oparciu o samouczek na stronie "Katacoda" [3] oraz stronę internetową "Konkretny.pl" [1]. Zdjęcia użyte do zilustrowania krok po kroku wykonywanych prac zostały zrobione w aplikacji pobranej ze strony "git for windows" [2].

2.2.1 Tworzenie nowego repozytorium

Aby stworzyć nowe repozytorium, trzeba pokazać programowi, gdzie znajduje się nasz projekt. Aby to zrobić używamy w konsoli komend:

- `dir` - wyświetlenie się wszystkich plików znajdujących się w danej lokalizacji
- `cd nazwakatalogu` - przejście do wybranego katalogu

```

MINGW64/c/Users/adama/documents/GIT
adama@DESKTOP-KC5J588 MINGW64 ~
$ dir
3D\ Objects      Downloads
AppData          Favorites
CLionProjects    IntelGraphicsProfiles
Contacts         Links
Cookies          Menu\ Start
Dane\ aplikacji  Moje\ dokumenty
Desktop          Music
Documents        NTUSER.DAT

adama@DESKTOP-KC5J588 MINGW64 ~
$ cd documents

adama@DESKTOP-KC5J588 MINGW64 ~/documents
$ dir
W celu zachowania prywatności
niektóre pliki zostały wymazane
GTR
Georg\ Wilhelm\ Friedrich\ Hegel\ -\ Nauka\ logiki.\ T.\ 1\ -\ komentarz.docx
Klei
Latex\ backup.txt
Moja\ muzyka
Moje\ obrazy

adama@DESKTOP-KC5J588 MINGW64 ~/documents
$ cd GIT

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$ dir
Przykładowy\ plik.txt

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$

```

Rysunek 2.1: Przejście do wybranego katalogu

Po przejściu do docelowej lokalizacji wpisujemy komendy:

- `git config --global user.name Twoje imię i nazwisko`
- `git config --global user.email Twój adres email`

```

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$ git config --global user.name Adam O

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$ git config --global user.email przykładowy.adres@gmail.com

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$ |

```

Rysunek 2.2: Ustalenie informacji autora

Komendy te wpisujemy tylko podczas pierwszego uruchomienia środowiska. GIT zapisuje wtedy u siebie informacje o autorze. Następnie wpisujemy komendę:

- `git init`

Komenda ta jest odpowiedzialna za stworzenie naszego repozytorium i możliwa nam dalszą pracę. Aby nasze pliki miały się gdzie zapisać potrzebujemy musimy dodać nasz serwer. Robimy to komendą:

- `git remote add nazwa-repozytorium git://adres/nazwa-pliku.git`

Po przejściu całej tej procedury możemy przejść do wysłania pliku.

```

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$ git init
Reinitialized existing Git repository in C:/Users/adama/Documents/GIT/.git/

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$ git remote add Probne_repozytorium git://documents/GIT/Plik.git

adama@DESKTOP-KC5J588 MINGW64 ~/documents/GIT (master)
$

```

Rysunek 2.3: Tworzenie repozytorium oraz serwera

2.2.2 Dodawanie zmian

Aby dodać zmianę i wysłać ją na serwer musimy najpierw określić, które pliki mają być tam dołączone. wpisujemy komendę:

- `git add *`

Sprawi ona, że automatycznie dodadzą się wszystkie pliki z lokalizacji, w której się znajdujemy. Po tej operacji musimy zatwierdzić zmiany w projekcie oraz nadać im nazwę. W tym celu stosujemy:

- `git commit -m 'nazwa commita '`

Gdy już dodamy wszystkie pliki i zatwierdzimy zmiany, możemy je wysłać na serwer używając komendy:

- `git push nazwa-repozytorium master`

```

adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git add *

adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git commit -m 'probny commit'
[master 5c8b94d] probny commit
1 file changed, 1 insertion(+)
create mode 100644 "Przyk\305\202adowy plik1222.txt"

adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git push Probne_repozytorium master

```

Rysunek 2.4: Dodawanie i wysyłanie pliku

Po zatwierdzeniu tej komendy, wszystkie wybrane przez nas wcześniej pliki zostały wysłane na wybrany wcześniej serwer.

2.2.3 Pobieranie zmian

Aby pobrać z serwera najbardziej aktualną wersję naszego pliku używamy komendy:

- `git pull nazwa-repozytorium`

Gdy okaże się, że aktualna wersja jest zła i potrzebujemy się cofnąć do poprzedniej wersji postępujemy według podanego poniżej przykładu:

- `git log`

Po wpisaniu i zatwierdzeniu tej komendy wyświetlą nam się wszystkie dostępne wersje naszego pliku. Aby pobrać wybraną przez nas wersję wpisujemy kolejną komendę:

- `git reset --hard nazwa-specjalna-commita` - przy czym za nazwę wpisujemy ciąg znaków, który wyświetlił nam się przy požądanej przez nas wersji commita po wpisaniu poprzedniej komendy.

```
adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git pull Probne_repozytorium

adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git log
commit 5c8b94d0521a4d2522b44aef8ddf41ab78bb3419 (HEAD -> master)
Author: Adam <przykładowy.adres@gmail.com>
Date: Tue Jan 25 16:50:43 2022 +0100

    probny commit

commit 5c667d8b7ddb1d6ac98832f1d00e2f98fcd260c
Author: Adam <przykładowy.adres@gmail.com>
Date: Tue Jan 25 16:43:43 2022 +0100

    Pierwszy commit

commit d155670894ba07826d47e0a2603457292bb4b13d
Author: Adam <przykładowy.email@gmail.com>
Date: Tue Jan 25 15:32:21 2022 +0100

    Proba

adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git reset d155670894ba07
Unstaged changes after reset:
D Przykładowy plik.txt
```

Rysunek 2.5: Pobieranie wybranej przez nas wersji pliku

2.2.4 Sprawdzanie zmian pomiędzy dwoma wersjami

W przypadku, gdy chcielibyśmy sprawdzić zmiany pomiędzy poszczególnymi commitami możemy posłużyć się komendą:

- `git log -p -n` - Gdzie za 'n' podstawiamy liczbę commitów, które chcemy przeanalizować, poczynawszy od najnowszego. Gdy chcemy sprawdzić wszystkie commity, możemy to pominąć przy wpisywaniu.

Zostaną wtedy wyświetlone wszystkie różnice pomiędzy wybranymi przez nas commitami.

```
adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$ git log -p
commit d155670894ba07826d47e0a2603457292bb4b13d (HEAD -> master)
Author: Adam <przykładowy.email@gmail.com>
Date: Tue Jan 25 15:32:21 2022 +0100

    Proba

diff --git "a/Przyk\305\202adowy plik.txt" "b/Przyk\305\202adowy plik.txt"
new file mode 100644
index 0000000..e69de29

adama@DESKTOP-KC5J588 MINGW64 ~/documents/git (master)
$
```

Rysunek 2.6: Sprawdzanie zmian pomiędzy plikami

Rozdział 3

LAB 3

Podczas trzecich labolatoriów skupiliśmy się na programowaniu językiem Python. Do zalet tego języka zaliczyć można dużo prostszą i przyjemniejszą dla oka składnię. Ma on również wiele dodatkowych pakietów jak numpy oraz matplotlib. Do jego wad można jednak zaliczyć to, że w porównaniu z C++ jest on dużo wolniejszy.

3.1 Wykonywanie i wyświetlanie prostych obliczeń

Pierwszym naszym zadaniem było zadeklarowanie zmiennych oraz wyświetlenie wyniku sumy/różnicy/mnożenia/dzielenia pomiędzy nimi. Aby to zrobić, trzeba najpierw przypisać do wybranych przez nas zmiennych jakąś wartość, a następnie z użyciem polecenia "print" wyświetlić wynik pożądaną przez nas operacji.

```
a = 1
b = 2

print (a+b)
```

3.2 Instrukcje warunkowe

Następnie przeszliśmy do instrukcji warunkowych. W celu napisania instrukcji warunkowej, musimy po wpisaniu komendy *if* podać warunek, dla którego dana funkcja ma być spełniona, po czym napisać poniżej, jakie czynności mają zajść w przypadku spełnienia warunku. Gdy to zrobimy, możemy wpisać jeszcze komendę *elif*, w której również umieszczamy warunek oraz instrukcję w przypadku jego spełnienia, a na sam koniec wpisujemy komendę *else*, przy której wpisujemy już tylko instrukcje, jakie mają zostać wykonane w przypadku niespełnienia się żadnego z poprzednich warunków.

```
a = 5
if a>5:
    print ('a>5')
elif a<5:
    print ('a<5')
else:
    print ('a=5')
```

3.3 Pętle

Kolejnym krokiem było poznanie pętli. Pierwszą z nich jest pętla *while*. Na jej składnie wliczają się: warunek oraz czynności, które w odróżnieniu od funkcji *if* będą się wykonywały tak długo, jak długo jest spełniony warunek. Trzeba więc uważać, aby podczas jego tworzenia nie stworzyć warunku, który zawsze będzie spełniony, gdyż program będzie działał w nieskończoność.

- Przykład pętli, której wynikiem będzie działanie w nieskończoność

```
a = 5
while a<6:
    print ("Prawda")
```

- Przykład pętli wyświetlającej prawdę, gdy "a" jest mniejsze od 6

```
a = 5
while a<6:
    print ("Prawda")
    a = a+1
```

Kolejnym rodzajem jest pętla *for*. Działa ona nieco inaczej w porównaniu ze swoim odpowiednikiem w C++. W warunku tej pętli wpisujemy sobie zmienną, a następnie wyrażenie *inrange(x,y)*, gdzie przedział od "x" do "y" pokazuje nam, jakie wartości będzie po kolei przyjmowała zadeklarowana zmienna, przy czym należy pamiętać, że przedział jest podawany w zakresie (x;y]. Po podaniu warunku, tak samo jak w pętli *while* podajemy czynności, które mają zostać wykonane, gdy warunek jest spełniony.

```
for i in range (0,6):
    print (i)
```

W tym przypadku program wyświetli nam po kolei: 0, 1, 2, 3, 4, 5.

Rozdział 4

LAB 4

Podczas tych labolatoriów pracowaliśmy w Pythonie z wykorzystaniem pakietów "numpy" oraz "matplotlib" które znacznie zwiększają nasze możliwości w Pythonie, oraz przyspieszają pracę nad kodem.

4.1 Biblioteka "Numpy"

Dzięki bibliotece "numpy" zyskujemy możliwość wykonywania operacji na macierzach, takich jak: tworzenie macierzy o dowolnym wymiarze, odczytywanie wybranej wartości, działania na macierzach, ich transponowanie, obliczanie wyznacznika i wiele innych funkcji, których wykonanie bez owej biblioteki zajęło by nam wiele czasu.

- Aby móc używać pakietu "numpy" trzeba zaimportować do swojego programu wpisując komendę:

```
import numpy as np
```

4.1.1 Tworzenie macierzy

Aby stworzyć macierz, a potem ją wyświetlić, musimy ją sobie nazwać. Następnie odnosząc się do biblioteki "numpy" - wpisując *np* tworzymy swoją macierz, wpisując po przecinku wartości, jakie chcemy, żeby miała.

```
A = np.matrix([
    [ 1,2,3,...],
    [ 11,12,13,...],
    [ 21,22,23,...],
    [...,...,...]
])
print (A)
```

4.1.2 Mnożenie macierzy

Aby pomnożyć ze sobą macierze, wystarczy zadeklarować wybrane przez nas dwie macierze, pamiętając o zasadach ich mnożenia, a następnie wyświetlić wynik działania:

```

A = np.matrix(
    [ [ 1,2,3],
      [ 3,4,5],
      [ 6,7,8]
    ])

B = np.matrix(
    [ [7,6,3],
      [ 3,42,5],
      [ 6,7,8]
    ])
AA= A*B

print (AA)

```

4.1.3 Obliczanie macierzy odwrotnej

Żeby znaleźć macierz odwrotną, wystarczy podnieść naszą macierz do potęgi "-1" (co robimy poprzez wpisanie `**`) i wyświetlić otrzymany wynik:

```

A = np.matrix(
    [ [ 2,1],
      [ 5,3]
    ])

B = A**(-1)

print (B)

```

4.1.4 Obliczanie wyznacznika macierzy

Również liczenie wyznacznika macierzy, z wykorzystaniem pakietu "numpy" jest proste i nie trzeba tego robić ręcznie. Aby go policzyć, wystarczy użyć dedykowanej do tego funkcji `np.linalg.det()`, która sama obliczy nam wyznacznik dowolnej macierzy i zwróci jego wartość.

```

A = np.matrix(
    [
        [ 22,1],
        [ 5,3]
    ])

print (np.linalg.det(A))

```

4.2 Biblioteka "Matplotlib"

Dzięki tej bibliotece jesteśmy w stanie rysować w programie wykresy funkcji, porównywać je między sobą, wyznaczać punkty przecięć dla wielu wykresów.

- Żeby móc używać pakietu "numpy" trzeba zaimportować do swojego programu wpisując komendę:

```
import matplotlib.pyplot as plt
```

By stworzyć wykres funkcji, trzeba zadeklarować dwie zmienne. W pierwszej, podajemy przedział, jaki będzie wyświetlany, a następnie podajemy, skok wartości wyświetlonych w tym przedziale. Druga zmienna odpowiada za przekazanie funkcji matematycznej, jaką chcemy, żeby program nam narysował:

```
x = np.arange(4, 10, 0.1)
```

Wartość "4" odpowiada za początek rysowania wykresu, "10" za jego koniec, a "0.1" odpowiada skok wyświetlonych elementów.

```
y = 2*x+2
```

Więcej możliwości biblioteki "Matplotlib" poznaliśmy na następnych zajęciach.

Rozdział 5

LAB 5

Podczas tych labolatoriów poznawaliśmy kolejne możliwości poznanych przez nas pakietów. Dowiedzieliśmy się jak dodawać opisy osi, siatkę, zmieniać kolor i grubość wykresów liniowych tworzyć wykresy kilku funkcji na tym samym układzie współrzędnych, zaznaczać ich wspólne części, tworzyć wykresy kołowe i słupkowe.

5.1 Konfigurowanie wyglądu wykresu

Aby nasz wykres nie był "surowy" i można było łatwiej go odczytywać, możemy użyć następujących funkcji:

- Dodanie siatki do wykresu:

```
plt.grid()
```

- Dodanie legendy:

```
plt.plot(x,y, label = '$ wzór funkcji $')  
#...  
plt.legend()
```

- Dodanie opisu osi:

```
plt.xlabel ("Nazwa osi X")  
plt.ylabel ("Nazwa osi Y")
```

- Dodanie opisu wykresu:

```
plt.title("Nazwa wykresu funkcji")
```

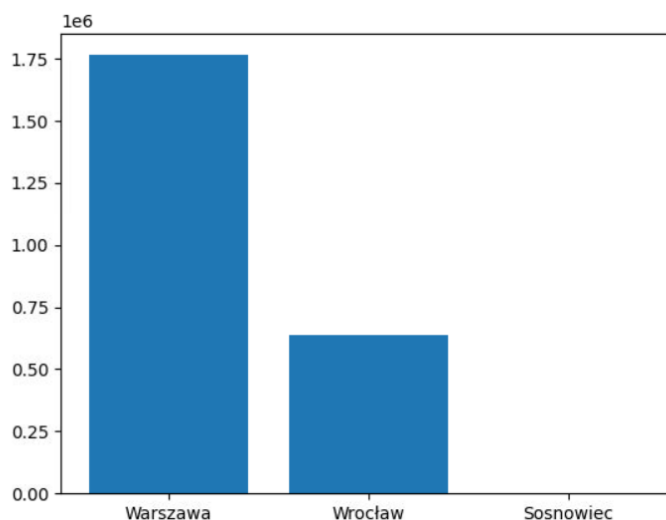
5.2 Inne typy wykresów

Jak już zostało wspomniane w rozdziale 4.2, biblioteka "matplotlib" oferuje nam wiele możliwości i oprócz tworzenia wykresów liniowych z jej pomocą możemy stworzyć inne typy wykresów, takie jak wykresy słupkowe lub kołowe.

5.2.1 Wykres słupkowy

Aby stworzyć wykres słupkowy, musimy tak jak w przypadku tworzenia wykresu funkcji liniowej, zadeklarować osie, na których będzie się wyświetlał wykres. Przykładem może być wykres liczby mieszkańców różnych miast w Polsce:

```
miasto = ['Warszawa', 'Wrocław' , 'Sosnowiec']
populacja = [1765000, 638659, 8]
#Dla każdego miasta (oś x) przypisana jest liczba mieszkańców (oś y),
#którą wywołuje się później funkcją:
plt.bar(miasto, populacja)
plt.show()
```



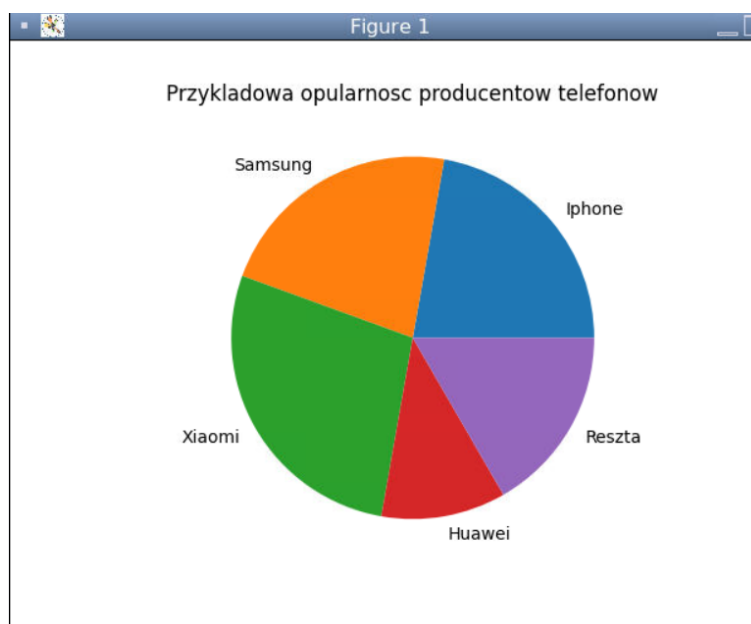
Rysunek 5.1: Wynik działania programu - wykres słupkowy

5.2.2 Wykres kołowy

Stworzenie wykresu kołowego jest bardzo podobne do tworzenia wykresu słupkowego. Różnice pojawiają się dopiero pod koniec pisania kodu, kiedy decydujemy, co tak dokładnie chcemy wyświetlić.

```
import matplotlib.pyplot as plt

ilosc = (4, 4, 5, 2 ,3)
marka = ('Iphone', 'Samsung', 'Xiaomi', 'Huawei', 'Reszta')
plt.pie(ilosc, labels = marka)
plt.title('Przykładowa opularnosc producentow telefonow')
plt.show()
```

Rysunek 5.2: Wynik działania programu - wykres kołowy

Rozdział 6

Zadania

6.1 Stwórz wykres napięcia ładowania i rozładowania kondensatora

6.1.1 Ładowanie kondensatora

$$\begin{array}{l} 1 \quad U_c(t) = U \cdot 1 - e^x \\ 2 \quad x = (-t)/R \cdot C \end{array}$$

Tabela 6.1: Wzory na ładowanie kondensatora

```
import numpy as np
import matplotlib.pyplot as plt

U = 6.0
R = 6250
C1 = 0.00047
C2 = 0.00094
e = 2,718281828459045235360

XX1 = R * C1
XX2 = R * C2
YY = 1-2,718281828459045235360
t = np.linspace(0, 10, 1000)
Uc1=U*(YY**(-t/XX1))
Uc2=U*(YY**(-t/XX2))

plt.grid()
plt.xlabel ("Czas")
plt.ylabel ("Napięcie")
plt.title("Wykres ładowania kondensatora")
plt.legend()
```

```
plt.plot(t , Uc1)
plt.plot(t , Uc2)
plt.show()
```

6.1.2 Rozładowanie kondensatora

Wzór rozładowania kondensatora:

$$U_c(t) = U * e^x$$

$$x = (-t)/R * C$$

```
import numpy as np
import matplotlib.pyplot as plt
```

```
Uc = 6.0
R = 6250
C1 = 0.00047
C2 = 0.00094
e = 2,718281828459045235360
```

```
t = t = np.linspace(0, 10, 1000)
U1 = Uc * e**(-t/(R*C1))
U2 = Uc * e**(-t/(R*C2))
```

```
plt.grid()
plt.xlabel ("Czas")
plt.ylabel ("Napięcie")
plt.title("Wykres ładowania kondensatora")
plt.legend()
plt.plot(t, U1)
plt.plot(t, U2)
plt.show()
plt.figure()
```

6.2 Zadanie ze studentami

Wczytaj plik csv, wylicz ocenę końcową z podanego poniżej wzoru. Na podstawie uzyskanych danych stwórz wykres oraz zapisz do pliku:

- średnią uzyskanych ocen,
- liczbę osób z poszczególnymi ocenami,
- liczbę osób które uzyskały zaliczenie
- liczbę osób które nie zaliczyły kursu.

Wzór na obliczenie oceny:

$$Ocena = 0.6 * WYK + 0.4 * LAB \quad (6.1)$$

6.2.1 Pobieranie i zapisywanie danych do pliku

```
import pandas as pd
import numpy as np
data = pd.read_csv('oceny.csv')
print(data) #Pobranie i wyświetlenie pliku

nie_zdalo=data[data['ocena_lab']<3.0]['ocena_lab'].count()+data[data['ocena_wyk']<3.0]
#Obliczenie, ile studentów nie zaliczyło grupy kursów
#Z uwzględnieniem, że istnieją studenci, którzy nie zaliczyli jednego kursu
#bądź nie zaliczyli obu
print()
print('Nie zdalo',nie_zdalo,'osob')

zaliczylo = (data[data['ocena_lab']>=2.0]['ocena_lab'].count()-nie_zdalo)
print('Zaliczylo',zaliczylo,'osob')
#Obliczenie, ilu studentów zdało kursy

idx=['Ilosc ocen 2.0','Ilosc ocen 2.5','Ilosc ocen 3.0','Ilosc ocen 3.5','Ilosc ocen 4.0']
#Stworzenie nazw dla tabel, które będą wyświetlały się
#w wygenerowanym później pliku

wyk=pd.Series([data[data['ocena_wyk']==2.0]['ocena_wyk'].count(),
data[data['ocena_wyk']==2.5]['ocena_wyk'].count(),
data[data['ocena_wyk']==3.0]['ocena_wyk'].count(),
data[data['ocena_wyk']==3.5]['ocena_wyk'].count(),
data[data['ocena_wyk']==4.0]['ocena_wyk'].count(),
data[data['ocena_wyk']==4.5]['ocena_wyk'].count(),
data[data['ocena_wyk']==5.0]['ocena_wyk'].count(),
data['ocena_wyk'].mean(),nie_zdalo,zaliczylo],index=idx)
#Zestawienie danych z wykładów, tj. ocena końcowa,
#ilość osób która zdała, średnia ocen
lab=pd.Series([data[data['ocena_lab']==2.0]['ocena_lab'].count(),
data[data['ocena_lab']==2.5]['ocena_lab'].count(),
data[data['ocena_lab']==3.0]['ocena_lab'].count(),
data[data['ocena_lab']==3.5]['ocena_lab'].count(),
data[data['ocena_lab']==4.0]['ocena_lab'].count(),
data[data['ocena_lab']==4.5]['ocena_lab'].count(),
data[data['ocena_lab']==5.0]['ocena_lab'].count(),
data['ocena_lab'].mean(),nie_zdalo,zaliczylo],index=idx)
#Zestawienie danych z laboratoriów, tj. ocena końcowa,
#ilość osób która zdała, średnia ocen

df=pd.DataFrame([wyk,lab],index=['wyklady',"laboratoria"])
```

```
df.to_csv("Podsumowanie_ocen.csv")
#Stworzenie pliku "podsumowanie_ocen"
#i zawarcie w nich wygenerowanych wcześniej danych

zestawienie = pd.read_csv('Podsumowanie_ocen.csv')
print(zestawienie)
#Wyświetlenie owego pliku
```

```

      Unnamed: 0  indeks_studenta  ocena_lab  ocena_wyk
0              0             962560         5.0         3.0
1              1             905220         2.0         4.5
2              2             882698         2.0         3.5
3              3             856084         3.5         2.5
4              4             909337         4.0         5.0
..           ...              ...         ...         ...
493           493             909272         5.0         3.5
494           494             860148         4.0         2.0
495           495             919541         2.0         4.0
496           496             915447         4.5         4.0
497           497             905212         4.0         5.0

[498 rows x 4 columns]

Nie zdalo 248 osob
Zaliczylo 250 osob
      Unnamed: 0  ...  Ilosc osob ktore nie zaliczyly
0      wykłady  ...                               250.0
1  laboratoria  ...                               250.0

[2 rows x 11 columns]
> 
```

Rysunek 6.1: Wynik działania programu - tabela ze wczytanego pliku, wyniki operacji w programie oraz tabela wczytana do pliku

6.2.2 Pobieranie i tworzenie wykresu danych

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import csv

mniej_2 = 0
mniej_3 = 0
mniej_35 = 0
mniej_4 = 0
mniej_45 = 0
mniej_5 = 0
mniej_55 = 0

#Poniżej: otworzenie pliku oceny.csv,
```

```

#wskazanie kolumn z ocenami z labolatoriów i wykładów,
#zamiana ocen na liczby zmiennoprzecinkowe,
#obliczenie oceny koncowej z podanego wzoru,
#stworzenie listy warunkowej w celu pogrupowania ocen.

with open('oceny.csv', 'r', encoding='utf-8') as csvfile:
    csvreader = csv.DictReader(csvfile)

    for row in csvreader:
        ocena_lab = float (row['ocena_lab'])
        ocena_wyk = float (row['ocena_wyk'])
        ocena_koncowa = ocena_lab * 0.4 + ocena_wyk * 0.6
        if ocena_koncowa <3.0:
            mniej_2 = mniej_2 +1
        elif (ocena_koncowa >=3.0) and (ocena_koncowa <3.5):
            mniej_3 = mniej_3 +1
        elif (ocena_koncowa >=3.5) and (ocena_koncowa <4.0):
            mniej_35 = mniej_35 +1
        elif (ocena_koncowa >=4.0) and (ocena_koncowa <4.5):
            mniej_4 = mniej_4 +1
        elif (ocena_koncowa >=4.5) and (ocena_koncowa <5.0):
            mniej_45 = mniej_45 +1
        elif (ocena_koncowa >=5.0) and (ocena_koncowa <5.5):
            mniej_5 = mniej_5 +1
        else:
            mniej_55 = mniej_55 +1

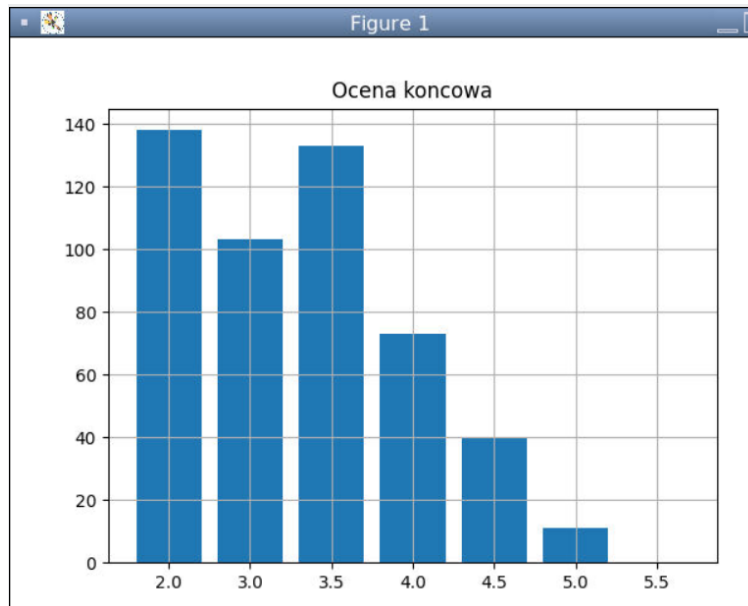
    print(mniej_2)
    print(mniej_3)
    print(mniej_35)
    print(mniej_4)
    print(mniej_45)
    print(mniej_5)
    print(mniej_55)

ocena = ['2.0', '3.0', '3.5', '4.0', '4.5', '5.0', '5.5']
ile_osob = [mniej_2, mniej_3, mniej_35, mniej_4, mniej_45, mniej_5, mniej_55]

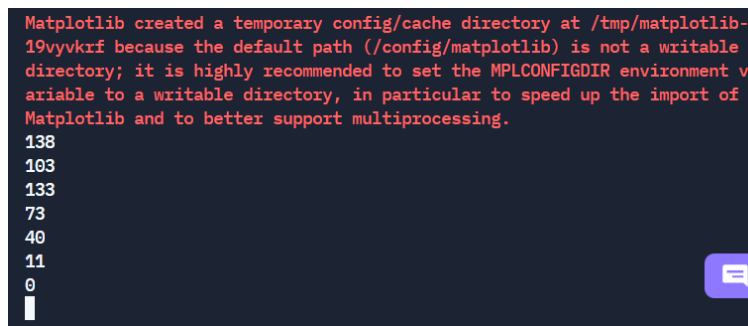
plt.title("Ocena koncowe") #Nadanie tytułu naszemu wykresowi
plt.grid() #Pokazanie siatki na wykresie
plt.bar(ocena, ile_osob) #Zadeklarowanie wykresu słupkowego i jego zmiennych
plt.show()

```

Używając danych które odczytaliśmy z pliku oraz wzoru 6.1 możemy obliczyć i wyświetlić w formie wykresu słupkowego ilość studentów, która otrzymała daną ocenę.



Rysunek 6.2: Wykres przedstawiający ile studentów uzyskało daną ocenę



Rysunek 6.3: Wartości odpowiadające liczbie studentów, którzy uzyskali daną ocenę

Rozdział 7

L^AT_EX

LaTeX pozwala nam na tworzenie dokumentów podobnie jak Word z pakietu MS Office, z tą różnicą że w LaTeX 'ie możemy w dużo prostszy i przyjemniejszy sposób stworzyć artykuł który zawiera:

- stronę tytułową,
- spis treści,
- rozdziały,
- sekcje,
- podsekcje.

Możemy również w łatwy sposób dodawać:

- wyliczenia,
- wzory,
- grafiki,
- kody źródłowe,
- bibliografie.

7.1 Tworzenie ogólnego zarysu dokumentu

7.1.1 Definiowanie rodzaju

LaTeX pozwala nam stworzyć wiele rodzajów dokumentów, wykorzystując przy tym różne paczki, jednak ich mnogość sprawia, że nie da się ich tutaj wszystkich wymienić, a w internecie dostępne jest wiele szablonów udostępnianych przez instytucje z całego świata [7.1](#). Warto jednak wspomnieć, że dodawanie różnych rozszerzeń, jak i ustalanie autora czy tytułu dokumentu robimy przez komendę [7.2](#).


```
%Ustalenie rodzaju dokumentu
\documentclass{article}

% Włączenie polskich znaków
\usepackage[polish]{babel}

% Ustalenie marginesów
\usepackage[letterpaper,top=2cm,bottom=2cm,left=3cm,right=3cm,marginparwidth=1.7
5cm]{geometry}

% Przydatne paczki
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage[colorlinks=true, allcolors=blue]{hyperref}

%Ustalenie autora i tytułu
\title{Tytuł}
\author{Autor}
```

Rysunek 7.1: Przykładowe rozszerzenia oraz przykładowe definiowanie dokumentu

```
\begin{document}
```

Rysunek 7.2: Wszystkie podane poniżej komendy, w celu poprawnego działania należy wpisać po tej komendzie

```
\maketitle
```

Rysunek 7.3: Komenda, która tworzy spis treści w naszym dokumencie

7.1.2 Tworzenie spisu treści

W celu stworzenia spisu treści wystarczy wpisać 7.3. Program sam go stworzy na podstawie rzeczy, które zostały przez nas wpisane do dokumentu.

7.1.3 Tworzenie rozdziałów, sekcji oraz podsekcji

Aby stworzyć w swoim dokumencie rozdziały - chapter, sekcje - section, czy podsekcje subsection, wpisujemy kolejno komendy:

```
\begin{document}

\chapter{Tytuł Rozdziału 1}
Tekst...
\section{Tytuł sekcji 1}
Tekst...
\subsection{Tytuł podsekcji 1}
Tekst...
\chapter{Tytuł Rozdziału 2}
Tekst...
\section{Tytuł sekcji 2}
Tekst...
\subsection{Tytuł podsekcji 2}
Tekst...

\end{document}
```

Rysunek 7.4: Komenda, która dzieli nasz dokument, na wybrane przez nas części

7.2 Wykorzystywanie funkcji LaTeX'a

7.2.1 Wyliczenia

Wyliczenia w LaTeX'ie możemy stosować na wiele sposobów. Poniżej pokazane są dwa najczęstsze przykłady 7.5, 7.6.

7.2.2 Wzory

Aby dodać wzór, musimy użyć jednej z pokazanych poniżej komend 7.7.

```

\begin{enumerate}
\item pierwsza pozycja
\item druga pozycja
\end{enumerate}

```

Rysunek 7.5: Wyliczanie, poprzez numerowanie kolejnych pozycji

```

\begin{itemize}
\item pierwsza pozycja
\item druga pozycja
\end{itemize}

```

Rysunek 7.6: Wyliczanie, poprzez wykropkowanie kolejnych pozycji

```

\begin{equation}
Tutaj wpisujemy nasz wzór
\end{equation}

```

```

$$tutaj również wpisujemy nasz wzór$$

```

A tak można wstawić wzór $wzór$ żeby był on częścią tekstu

Rysunek 7.7: Przykłady zawierania wzoru w tekście

7.2.3 Grafiki

W celu dodania grafiki, należy ją najpierw załadować do swojego edytora, a następnie wywołać ją za pomocą komendy:

```
\begin{figure}
\centering
\includegraphics[width=0.3\textwidth]{nazwa_zdjęcia.jpg}
\caption{\label{fig:frog}Tutaj wstawiamy opis obrazka.}
\end{figure}
```

Rysunek 7.8: Zawieranie grafiki w dokumencie

7.2.4 Kody źródłowe

By dodać kod źródłowy, np. z Pythona, i chcemy żeby nie był on w żaden sposób zmodyfikowany, musimy go zawrzeć w komendzie:

```
\begin{minted}{nazwa_języka}
Kod
\end{minted}
```

Rysunek 7.9: Dodawanie kodu źródłowego

7.2.5 Bibliografie

Aby stworzyć bibliografię, musimy stworzyć plik z rozszerzeniem *.bib* w którym umieszczamy wszystkie informacje na temat np.: książek lub prac używając do tego szablonów dostępnych w dokumentacji LaTeX'a, w zależności od tego, skąd zostały zapożyczone informacje 7.10. Następnie w przypadku odwołania się do danej pracy wpisujemy ko-

```
@article{greenwade93,
  author = "George D. Greenwade",
  title = "The {C}omprehensive {T}ex {A}rchive {N}etwork ({CTAN})",
  year = "1993",
  journal = "TUGBoat",
  volume = "14",
  number = "3",
  pages = "342--351"
}
```

Rysunek 7.10: Przykładowy szablon w pliku *.bib*

mendę 7.11. Aby wyświetlić na koniec spis wszystkich wykorzystanych prac, musimy wpisać 7.12.

```
\cite{greenwade93}
```

Rysunek 7.11: Odwołanie się do artykułu o nazwie "greenwade93"

```
\bibliographystyle{abbrv}  
\bibliography{nazwa_pliku}
```

Rysunek 7.12: Wyświetlenie bibliografii w dokumencie

Bibliografia

- [1] <http://konkretny.pl/jak-korzystac-z-gita-poradnik.html>.
- [2] <https://gitforwindows.org>.
- [3] <https://www.katacoda.com/courses/git>.