Politechnika Wrocławska

Podstawy programowania W11OPA-SI0072G
Wstęp do programowania W11FTE-SI0141WL
Wstęp do programowania W11IKW-SI0080WL

rok akademicki 2023/24
semestr letni

Wykład 6

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji (1)

- Odczyt i zapis plików - wprowadzenie
- Otwarcie pliku
- Zapis danych
- Zamknięcie pliku
- Odczyt danych
- Obsługa znaku nowego wiersza
- Obsługa danych liczbowych
- Zapis i odczyt z wykorzystaniem pętli

T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

Plan prezentacji (2)

- Obsługa błędów
- Konstrukcja try-except
- Obsługa wielu wyjątków
- Domyślny komunikat wyjątku
- Klauzula else
- Klauzula finally

T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

Odczyt i zapis plików

- Programy komputerowe nie muszą bazować tylko na danych podanych bezpośrednio przez użytkownika, mogą korzystać z danych zapisanych w plikach
- Przykłady programów zapisujących dane w plikach:
  - edytory tekstu,
  - edytory graficzne,
  - arkusze kalkulacyjne,
  - gry,
  - przeglądarki WWW.

Politechnika Wrocławska

Odczyt i zapis plików

Aby użyć w programie pliku, należy:

1. Otworzyć plik – po otworzeniu pliku następuje utworzenie połączenia między programem a plikiem (plik można otworzyć do zapisu lub do odczytu)
2. Przetworzyć plik – program zapisuje lub odczytuje dane
3. Zamknąć plik – gdy program zakończy przetwarzanie pliku, należy zamknąć połączenie między plikiem a programem

Politechnika Wrocławska

Odczyt i zapis plików

Typy plików:

- pliki tekstowe – dane tekstowe zapisane w kodowaniu ASCII lub Unicode
- pliki binarne – dane, które nie zostały skonwertowane na postać tekstu

Politechnika Wrocławska

Odczyt i zapis plików

Metody dostępu do plików:

- dostęp sekwencyjny – dane są przetwarzane po kolei – od początku pliku do końca
- dostęp swobodny – można przeskoczyć do konkretnego miejsca w pliku bez konieczności przetwarzania wcześniejszych danych

Politechnika Wrocławska

Otworzenie pliku

- Funkcja **`open()`** tworzy obiekt reprezentujący plik i wiąże go z konkretnym plikiem na dysku
- Ogólna postać wywołania

```
zmienna_pliku = open(nazwa_pliku, tryb)
```

- ***`zmienna_pliku`*** – to zmienna reprezentująca plik
- ***`nazwa_pliku`*** – ciąg tekstowy określający nazwę pliku
- ***`tryb`*** – ciąg tekstowy określający tryb

Politechnika Wrocławska

Otworzenie pliku

Wybrane tryby otwarcia plików

tryb	opis
`'r'`	plik otwarty tylko do odczytu – zawartość nie może być modyfikowana
`'w'`	plik otwarty do zapisu – zawartość istniejącego pliku zostanie usunięta, jeśli plik nie istnieje zostanie utworzony
`'a'`	plik otwarty do zapisu nowych danych – wszystkie nowe dane zostaną umieszczone na końcu pliku, jeśli plik nie istnieje będzie utworzony

Politechnika Wrocławska

Zapis danych w pliku

- Funkcja **`write()`** jest metodą obiektu reprezentującego plik
- Ogólna postać wywołania metody **`write()`**

```
zmienna_pliku.write(ciąg_tekstowy)
```

- Przykładowo

```
file.write('Jan Kowalski')
```

```
name = 'Jan Kowalski'
file.write(name)
```

Politechnika Wrocławska

Zamknięcie pliku

- Funkcja `close()` jest metodą obiektu reprezentującego plik
- Ogólna postać wywołania metody `close()`

```
zmienna_pliku.close()
```

Politechnika Wrocławska

Zapis danych w pliku

Przykład

01_file_write.py

```
1   # Przykładowy program wykonujący
2   # zapis do pliku.
3
4   def main():
5       #otwarcie pliku "noblisci.txt" do zapisu
6       outfile = open('noblisci.txt','w')
7
8       #zapisanie w pliku imion i nazwisk
9       #polskich laureatów Nagrody Nobla
10      #w dziedzinie literatury
11      outfile.write('Henryk Sienkiewicz\n')
12      outfile.write('Władysław Reymont\n')
13      outfile.write('Czesław Miłosz\n')
14      outfile.write('Wisława Szymborska\n')
15      outfile.write('Olga Tokarczuk\n')
16
17      outfile.close()
18
19  main()
```

Politechnika Wrocławska

Odczyt danych z pliku

- Funkcja **`read()`** jest metodą obiektu reprezentującego plik, która pozwala wczytać całą zawartość pliku do pamięci
- Ogólna postać wywołania metody **`read()`**

```
contents = zmienna_pliku.read()
```

Politechnika Wrocławska

Odczyt danych z pliku

- Funkcja **`readline()`** jest metodą obiektu reprezentującego plik, która pozwala odczytać jedną linię z pliku
- Ogólna postać wywołania metody **`readline()`**

```
contents = zmienna_pliku.readline()
```

Politechnika Wrocławska

Odczyt danych z pliku

01_file_write.py 02_file_read.py

```
1  # Przykładowy program wykonujący
2  # odczyt z pliku.
3
4  def main():
5      #otwarcie pliku "nobliści.txt" do odczytu
6      infile = open('nobliści.txt','r')
7
8      #wczytanie zawartości pliku metodą read()
9      file_contents = infile.read()
10
11     #zamknięcie pliku
12     infile.close()
13
14     #wyświetlenie danych wczytanych do pamięci
15     print(file_contents)
16
17 main()
```

Politechnika Wrocławska

Odczyt danych z pliku

01_file_write.py 02_file_read.py

```
1  # Przykładowy program wykonujący
2  # odczyt z pliku.
3
4  def main():
5      #otwarcie pliku "nobliści.txt" do odczytu
6      infile = open('nobliści.txt','r')
7
8      #wczytanie zawartości pliku metodą read()
9      file_contents = infile.read()
10
11     #zamknięcie pliku
12     infile.close()
13
14     #wyświetlenie danych wczytanych do pamięci
15     print(file_contents)
16
17 main()
```

Politechnika Wrocławska

Odczyt danych z pliku

```
Henryk Sienkiewicz
Władysław Reymont
Czesław Miłosz
Wisława Szymborska
Olga Tokarczuk
```

01_file_write.py | 02_file_read.py

```
1  # Przykładowy program wykonujący
2  # odczyt z pliku.
3
4  def main():
       #otwarcie pliku "noblisci.txt" do odczytu
       infile = open('noblisci.txt','r')

       #wczytanie zawartości pliku metodą read()
       file_contents = infile.read()

       #zamknięcie pliku
       infile.close()

       #wyświetlenie danych wczytanych do pamięci
       print(file_contents)

main()
```

Politechnika Wrocławska

Odczyt danych z pliku

01_file_write.py | 02_file_read.py | 03_file_read.py

```python
# Przykładowy program wykonujący
# odczyt z pliku.

def main():
    #otwarcie pliku "nobliści.txt" do odczytu
    infile = open('nobliści.txt','r')

    #wczytanie zawartości pliku metodą readline()
    line1 = infile.readline()
    line2 = infile.readline()
    line3 = infile.readline()
    line4 = infile.readline()
    line5 = infile.readline()

    #zamknięcie pliku
    infile.close()

    #wyświetlenie danych wczytanych do pamięci
    print(line1)
    print(line2)
    print(line3)
    print(line4)
    print(line5)


main()
```

Politechnika Wrocławska

Odczyt danych z pliku

```
Henryk Sienkiewicz

Władysław Reymont

Czesław Miłosz

Wisława Szymborska

Olga Tokarczuk
```

01_file_write.py | 02_file_read.py | 03_file_read.py

```
# Przykładowy program wykonujący
# odczyt z pliku.


ie pliku "nobliści.txt" do odczytu
= open('nobliści.txt','r')

nie zawartości pliku metodą readline()
infile.readline()
infile.readline()
infile.readline()
infile.readline()
infile.readline()

ęcie pliku
close()

tlenie danych wczytanych do pamięci
ine1)
ine2)
ine3)
ine4)
    print(line5)


main()
```

Politechnika Wrocławska

Znak nowego wiersza

- Znak nowego wiersza służy do rozdzielania linii tekstu w pliku
- Ciągi tekstowe mają metodę `rstrip()`, którą można wykorzystać do usunięcia określonych znaków tekstowych (np. znaku nowego wiersza) z końca ciągu

Politechnika Wrocławska

Znak nowego wiersza

01_file_write.py | 02_file_read.py | 03_file_read.py | 04_file_write.py

```
1  # Przykładowy program wykonuje
2  # zapis danych tekstowych w pliku
3  # dodając znak nowego wiersza.
4
5  def main():
6      print('Podaj imiona trójki przyjaciół.')
7      name1 = input('Przyjaciel #1: ')
8      name2 = input('Przyjaciel #2: ')
9      name3 = input('Przyjaciel #3: ')
10
11     #otwarcie pliku "imiona.txt" do zapisu
12     outfile = open('imiona.txt','w')
13     #zapisanie 3 linii tekstu
14     #(dołączony znak nowej linii)
15     outfile.write(name1 + '\n')
16     outfile.write(name2 + '\n')
17     outfile.write(name3 + '\n')
18     #zamknięcie pliku
19     outfile.close()
20
21 main()
```

Politechnika Wrocławska

Znak nowego wiersza

01_file_write.py | 02_file_read.py | 03_file_read.py | 04_file_write.py

```
1  # Przykładowy program wykonuje
2  # zapis danych tekstowych w pliku
3  # dodając znak nowego wiersza.
4
5  def main():
6      print('Podaj imiona trójki przyjaciół.')
7      name1 = input('Przyjaciel #1: ')
8      name2 = input('Przyjaciel #2: ')
9      name3 = input('Przyjaciel #3: ')
10
11     #otwarcie pliku "imiona.txt" do zapisu
12     outfile = open('imiona.txt','w')
13     #zapisanie 3 linii tekstu
14     #(dołączony znak nowej linii)
15     outfile.write(name1 + '\n')
16     outfile.write(name2 + '\n')
17     outfile.write(name3 + '\n')
18     #zamknięcie pliku
19     outfile.close()
20
21 main()
```

Politechnika Wrocławska

Znak nowego wiersza

01_file_write.py | 02_file_read.py | 03_file_read.py | 04_file_write.py | 05_file_rstrip.py

```
1  # Przykładowy program wykonuje
2  # odczyt danych tekstowych z pliku (readline),
3  # i usuwa znak nowej linii (rstrip).
4
5  def main():
6      print('Program wczytuje imiona trójki przyjaciół.')
7
8      #otwarcie pliku "imiona.txt" do odczytu
9      infile = open('imiona.txt','r')
10     #odczytanie 3 linii tekstu
11     line1 = infile.readline()
12     line2 = infile.readline()
13     line3 = infile.readline()
14     #usunięcie znaków nowej linii
15     line1 = line1.rstrip('\n')
16     line2 = line2.rstrip('\n')
17     line3 = line3.rstrip('\n')
18     #zamknięcie pliku
19     infile.close()
20
21     print('Przyjaciel #1:',line1)
22     print('Przyjaciel #2:',line2)
23     print('Przyjaciel #3:',line3)
24
25 main()
```

Politechnika Wrocławska

Znak nowego wiersza

01_file_write.py | 02_file_read.py | 03_file_read.py | 04_file_write.py | 05_file_rstrip.py

```
1  # Przykładowy program wykonuje
2  # odczyt danych tekstowych z pliku (readline),
3  # i usuwa znak nowej linii (rstrip).
4
5  def main():
6      print('Program wczytuje imiona trójki przyjaciół.')
7
8      #otwarcie pliku "imiona.txt" do odczytu
9      infile = open('imiona.txt','r')
10     #odczytanie 3 linii tekstu
11     line1 = infile.readline()
12     line2 = infile.readline()
13     line3 = infile.readline()
14     #usunięcie znaków nowej linii
15     line1 = line1.rstrip('\n')
16     line2 = line2.rstrip('\n')
17     line3 = line3.rstrip('\n')
18     #zamknięcie pliku
19     infile.close()
20
21     print('Przyjaciel #1:',line1)
22     print('Przyjaciel #2:',line2)
23     print('Przyjaciel #3:',line3)
24
25 main()
```

Politechnika Wrocławska

Zapis danych liczbowych

- Funkcja **`write()`** zapisuje ciągi znaków, zatem dane liczbowe przed zapisaniem do pliku funkcją **`write()`** należy skonwertować na łańcuch znakowy funkcją **`str()`**

Politechnika Wrocławska

Zapis danych liczbowych

06_file_write_numbers.py

```python
# Przykładowy program pokazuje
# zamianę danych liczbowych na tekst
# przed zapisaniem ich do pliku.

def main():
    print('Podaj trzy liczby całkowite')
    num1 = int(input('Pierwsza liczba: '))
    num2 = int(input('Druga liczba: '))
    num3 = int(input('Trzecia liczba: '))

    #otwarcie pliku "liczby.txt" do zapisu
    outfile = open('liczby.txt','w')
    #zapisanie 3 liczb po skonwertowaniu
    #na łańcuch tekstowy
    outfile.write(str(num1) + '\n')
    outfile.write(str(num2) + '\n')
    outfile.write(str(num3) + '\n')
    #zamknięcie pliku
    outfile.close()

main()
```

Politechnika Wrocławska

Zapis danych liczbowych

06_file_write_numbers.py

```python
# Przykładowy program pokazuje
# zamianę danych liczbowych na tekst
# przed zapisaniem ich do pliku.

def main():
    print('Podaj trzy liczby całkowite')
    num1 = int(input('Pierwsza liczba: '))
    num2 = int(input('Druga liczba: '))
    num3 = int(input('Trzecia liczba: '))

    #otwarcie pliku "liczby.txt" do zapisu
    outfile = open('liczby.txt','w')
    #zapisanie 3 liczb po skonwertowaniu
    #na łańcuch tekstowy
    outfile.write(str(num1) + '\n')
    outfile.write(str(num2) + '\n')
    outfile.write(str(num3) + '\n')
    #zamknięcie pliku
    outfile.close()

main()
```

Politechnika Wrocławska

Odczyt danych liczbowych

06_file_write_numbers.py | 07_file_read_numbers.py

```
1  # Przykładowy program pokazuje
2  # odczyt danych liczbowych z pliku
3  # zapisanych w postaci tekstu.
4
5  def main():
6      print('Program wczytuje trzy liczby całkowite')
7      print('z pliku "liczby.txt" i oblicza ich sumę.')
8
9      #otwarcie pliku "liczby.txt" do odczytu
10     infile = open('liczby.txt','r')
11     #odczytanie 3 linii tekstu
12     #zamiana liczb na int
13     num1 = int(infile.readline())
14     num2 = int(infile.readline())
15     num3 = int(infile.readline())
16     #zamknięcie pliku
17     infile.close()
18
19     #obliczenie sumy liczb
20     total = num1 + num2 + num3
21
22     #wyświetlenie liczb i stosowanego komunikatu
23     print('Odczytane liczby to:',num1, num2, num3)
24     print('Suma liczb wynosi:', total)
25
26 main()
```

Politechnika Wrocławska

Odczyt danych liczbowych

06_file_write_numbers.py | 07_file_read_numbers.py

```
1  # Przykładowy program pokazuje
2  # odczyt danych liczbowych z pliku
3  # zapisanych w postaci tekstu.
4
5  def main():
6      print('Program wczytuje trzy liczby całkowite')
7      print('z pliku "liczby.txt" i oblicza ich sumę.')
8
9      #otwarcie pliku "liczby.txt" do odczytu
10     infile = open('liczby.txt','r')
11     #odczytanie 3 linii tekstu
12     #zamiana liczb na int
13     num1 = int(infile.readline())
14     num2 = int(infile.readline())
15     num3 = int(infile.readline())
16     #zamknięcie pliku
17     infile.close()
18
19     #obliczenie sumy liczb
20     total = num1 + num2 + num3
21
22     #wyświetlenie liczb i stosowanego komunikatu
23     print('Odczytane liczby to:',num1, num2, num3)
24     print('Suma liczb wynosi:', total)
25
26 main()
```

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

- Zwykle pliki zawierają duże ilości informacji, które są przetwarzane z wykorzystaniem pętli

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

08_file_write_grades.py

```python
# Przykładowy program pokazuje
# zapis danych liczbowych do pliku
# z wykorzystaniem pętli.

def main():
    print('Program zapisuje do pliku "oceny.txt"')
    print('oceny podane przez użytkownika.')

    number = int(input('Ile ocen chcesz podać? '))

    #otwarcie pliku "oceny.txt" do zapisu
    outfile = open('oceny.txt','w')

    #pobranie od użytkownika ocen
    #i zapisanie ich do pliku.
    for count in range(1,number+1):
        grade = float(input(
            'Podaj ' + str(count) + ' ocenę: '))
        outfile.write(str(grade) + '\n')

    #zamknięcie pliku
    outfile.close()

main()
```

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

08_file_write_grades.py

```
1  # Przykładowy program pokazuje
2  # zapis danych liczbowych do pliku
3  # z wykorzystaniem pętli.
4
5  def main():
6      print('Program zapisuje do pliku "oceny.txt"')
7      print('oceny podane przez użytkownika.')
8
9      number = int(input('Ile ocen chcesz podać? '))
10
11     #otwarcie pliku "oceny.txt" do zapisu
12     outfile = open('oceny.txt','w')
13
14     #pobranie od użytkownika ocen
15     #i zapisanie ich do pliku.
16     for count in range(1,number+1):
17         grade = float(input(
18             'Podaj ' + str(count) + ' ocenę: '))
19         outfile.write(str(grade) + '\n')
20
21     #zamknięcie pliku
22     outfile.close()
23
24 main()
```

Przetwarzanie plików za pomocą pętli

- Funkcja **`readline()`** po osiągnięciu końca pliku zwraca pusty łańcuch

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

09_file_read_grades.py

```python
# Przykładowy program pokazuje odczyt
# danych liczbowych do pliku
# z wykorzystaniem pętli.
# Funkcja readline() zwraca pusty łańcuch
# jeśli osiągnięty zostanie koniec pliku.

def main():
    print('Program odczytuje oceny')
    print('z pliku "oceny.txt".')

    #otwarcie pliku "oceny.txt" do zapisu
    infile = open('oceny.txt','r')

    #odczytanie z pliku linii tekstu
    line = infile.readline()

    #dopóki linia nie jest pusta
    while line != '':
        #skonwertuj tekst na liczbę
        amount = float(line)
        #wyświetl ocenę
        print(format(amount,'.1f'))
        #wczytaj kolejną linię
        line = infile.readline()

    #zamknięcie pliku
    infile.close()

main()
```

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

09_file_read_grades.py

```python
# Przykładowy program pokazuje odczyt
# danych liczbowych do pliku
# z wykorzystaniem pętli.
# Funkcja readline() zwraca pusty łańcuch
# jeśli osiągnięty zostanie koniec pliku.

def main():
    print('Program odczytuje oceny')
    print('z pliku "oceny.txt".')

    #otwarcie pliku "oceny.txt" do zapisu
    infile = open('oceny.txt','r')

    #odczytanie z pliku linii tekstu
    line = infile.readline()

    #dopóki linia nie jest pusta
    while line != '':
        #skonwertuj tekst na liczbę
        amount = float(line)
        #wyświetl ocenę
        print(format(amount,'.1f'))
        #wczytaj kolejną linię
        line = infile.readline()

    #zamknięcie pliku
    infile.close()

main()
```

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

- Do odczytu danych można użyć również pętli **`for`**
- Obiekt reprezentujący plik udostępnia iterator, który może być wykorzystany w pętli **`for`**

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

09_file_read_grades.py | 10_file_read_grades.py

```
1  # Przykładowy program pokazuje odczyt
2  # danych liczbowych do pliku
3  # z wykorzystaniem pętli for.
4  # Pętla for "przechodzi" przez plik
5  # dzięki temu, że obiekt reprezentujący
6  # plik udostępnia iterator.
7
8  def main():
9      print('Program odczytuje oceny')
10     print('z pliku "oceny.txt".')
11
12     #otwarcie pliku "oceny.txt" do zapis
13     infile = open('oceny.txt','r')
14
15     #dla kolejnych linii pliku
16     for line in infile:
17         #skonwertuj tekst na liczbę
18         amount = float(line)
19         #wyświetl ocenę
20         print(format(amount,'.1f'))
21
22     #zamknięcie pliku
23     infile.close()
24
25 main()
```

Politechnika Wrocławska

Przetwarzanie plików za pomocą pętli

09_file_read_grades.py | 10_file_read_grades.py

```
1  # Przykładowy program pokazuje odczyt
2  # danych liczbowych do pliku
3  # z wykorzystaniem pętli for.
4  # Pętla for "przechodzi" przez plik
5  # dzięki temu, że obiekt reprezentujący
6  # plik udostępnia iterator.
7
8  def main():
9      print('Program odczytuje oceny')
10     print('z pliku "oceny.txt".')
11
12     #otwarcie pliku "oceny.txt" do zapis
13     infile = open('oceny.txt','r')
14
15     #dla kolejnych linii pliku
16     for line in infile:
17         #skonwertuj tekst na liczbę
18         amount = float(line)
19         #wyświetl ocenę
20         print(format(amount,'.1f'))
21
22     #zamknięcie pliku
23     infile.close()
24
25 main()
```

Obsługa błędów

- W wielu sytuacjach poprawność działania programu zależy od dostarczonych danych

Politechnika Wrocławska

Obsługa błędów

01_dzielenie.py

```python
#Program pokazuje sytuację, w której możliwe jest
#zgłoszenie wyjątku związanego z dzieleniem przez 0.

def main():
    num1 = int(input('Podaj liczbę: '))
    num2 = int(input('Podaj następną liczbę: '))

    #dzielenie nie będzie wykonane jeśli num2 == 0
    result = num1 / num2
    print(num1, 'dzielone przez', num2, 'daje', result)

main()
```

Politechnika Wrocławska

Obsługa błędów

01_dzielenie.py 02_dzielenie.py

```python
#Program pokazuje sytuację, w której uniknięto
#zgłoszenia wyjątku ZeroDivisionError.

def main():
    num1 = int(input('Podaj liczbę: '))
    num2 = int(input('Podaj następną liczbę: '))

    if num2 != 0:
        result = num1 / num2
        print(num1, 'dzielone przez', num2, 'daje', result)
    else:
        print('Nie można dzielić przez zero.')

main()
```

Politechnika Wrocławska

Konstrukcja try-except

01_dzielenie.py | 02_dzielenie.py | 03_dzielenie.py

```
1  #Program pokazuje obsługę wyjątku ValueError
2  #zgłoszonego przez funkcję int().
3
4  def main():
5      try:
6          #jeśli napisu nie można zamienić na liczbę całkowitą
7          #to jest zgłaszany wyjątek ValueError
8          num1 = int(input('Podaj liczbę: '))
9          num2 = int(input('Podaj następną liczbę: '))
10
11         if num2 != 0:
12             result = num1 / num2
13             print(num1, 'dzielone przez', num2, 'daje', result)
14         else:
15             print('Nie można dzielić przez zero.')
16
17     except ValueError:
18         print('Błąd!')
19
20 main()
21
```

Politechnika Wrocławska

Konstrukcja try-except

01_dzielenie.py | 02_dzielenie.py | 03_dzielenie.py

```
1  #Program pokazuje obsługę wyjątku ValueError
2  #zgłoszonego przez funkcję int().
3
4  def main():
5      try:
6
7
8
9
10
11                 spróbuj wykonać ten kod
12
13
14
15
16
17     except ValueError:
18         jeśli zostanie zgłoszony wyjątek ValueError - to przejdź tu
19
20 main()
21
```

Politechnika Wrocławska

Konstrukcja try-except

01_dzielenie.py | 02_dzielenie.py | 03_dzielenie.py

```
1   #Program pokazuje obsługę wyjątku ValueError
2   #zgłoszonego przez funkcję int().
3
4   def main():
5       try:
6           #jeśli napisu nie można z
7           #to jest zgłaszany wyjąte
8           num1 = int(input('Podaj l
9           num2 = int(input('Podaj r
10
11          if num2 != 0:
12              result = num1 / num2
13              print(num1, 'dzielone przez', num2, 'daje', result)
14          else:
15              print('Nie można dzielić przez zero.')
16
17      except ValueError:
18          print('Błąd!')
19
20  main()
21
```

jeśli funkcja int() nie może zamienić napisu na liczbę zgłosi wyjątek ValueError

Politechnika Wrocławska

Konstrukcja try-except

04_odczyt_pliku.py

```
1  #Program pokazuje inną sytuację, w której zgłaszany
2  #jest wyjątek.
3
4  def main():
5      filename = input('Podaj nazwę pliku: ')
6
7      #funkcja open() zgłasza wyjątek, jeśli
8      #nie istnieje plik o podanej nazwie
9      infile = open(filename,'r')
10
11     contents = infile.read()
12     print(contents)
13     infile.close()
14
15 main()
```

Politechnika Wrocławska

Konstrukcja try-except

04_odczyt_pliku.py 05_odczyt_pliku.py

```python
#Program pokazuje inną sytuację, w której zgłaszany
#jest wyjątek.

def main():
    try:
        filename = input('Podaj nazwę pliku: ')
        #funkcja open() zwraca wyjątek typu IOError
        infile = open(filename,'r')
        contents = infile.read()
        print(contents)
        infile.close()

    except IOError:
        #wyjątek jest obsłużony - wyświetlono komunikat
        print('Wystąpił błąd podczas próby odczytania')
        print('pliku o nazwie', filename)

main()
```

Politechnika Wrocławska

Obsługa wielu wyjątków

- Z kodu wykonywanego wewnątrz klauzli try mogą być zgłaszane różne wyjątki
- Powtórzenie klauzli except pozwala wykonywać różny kod zależnie od sytuacji

Politechnika Wrocławska

Obsługa wielu wyjątków

04_odczyt_pliku.py | 05_odczyt_pliku.py | 06_odczyt_pliku.py

```
#Program pokazuje obsługę wyjątków różnych typów.
#Program podsumowuje liczby zestawione w pliku
#sales_data.txt.

def main():
    total = 0.0

    try:
        #wyjątek IOError może być zgłoszony
        #przy otwarciu pliku
        infile = open('sales_data.txt','r')
        for line in infile:
            #wyjątek ValueError może być zgłoszony
            #przy konwertowaniu danych tekstowych
            amount = float(line)
            total += amount

        infile.close()
        print(format(total, '.2f'))
```

Politechnika Wrocławska

Obsługa wielu wyjątków

04_odczyt_pliku.py | 05_odczyt_pliku.py | 06_odczyt_pliku.py

```
20
21      except IOError:
22          print('Wystąpił błąd podczas odczytu pliku')
23
24      except ValueError:
25          print('W pliku znajdują się dane inne niż liczbowe')
26
27      except:
28          print('Wystąpił błąd.')
29
30  main()
```

Politechnika Wrocławska

Obsługa wielu wyjątków

- W przykładzie są obsłużone wyjątki typu:
  - IOError
  - ValueError

Obsługa wielu wyjątków

- Klauzula except może też obsługiwać wszystkie wyjątki

Politechnika Wrocławska

Obsługa wielu wyjątków

04_odczyt_pliku.py | 05_odczyt_pliku.py | 06_odczyt_pliku.py | 07_odczyt_pliku.py

```
#Program pokazuje obsługę wszystkich wyjątków
#(niezależnie od typu).


def main():
    total = 0.0

    try:
        infile = open('sales_data.txt','r')
        for line in infile:
            amount = float(line)
            total += amount

        infile.close()

        print(format(total, '.2f'))

    #klauzla except obsługuje wszystkie wyjątki
    except:
        print('Wystąpił błąd.')

main()
```

Politechnika Wrocławska

Obsługa wielu wyjątków

08_dzielenie.py

```python
#Program pokazuje obsługę dwóch typów wyjątków
#na przykładzie dzielenia dwóch liczb.

def main():
    try:
        num1 = int(input('Podaj liczbę: '))
        num2 = int(input('Podaj następną liczbę: '))

        result = num1 / num2
        print(num1, 'dzielone przez', num2, 'daje', result)

    except ValueError:
        print('Niepoprawne dane wejściowe.')

    except ZeroDivisionError:
        print('Błąd dzielenia przez 0.')

main()
```

Politechnika Wrocławska

Domyślny komunikat wyjątku

- Podczas obsługi wyjątku można opcjonalnie przypisać obiekt wyjątku
- Przykładowo:

```
except ValueError as err:
```

Politechnika Wrocławska

Domyślny komunikat wyjątku

- Podczas obsługi wyjątku można opcjonalnie przypisać obiekt wyjątku
- Przykładowo:

```
except ValueError as err:
```

- Można w ten sposób wyświetlić domyślny komunikat

Politechnika Wrocławska

Domyślny komunikat wyjątku

08_dzielenie.py 09_dzielenie.py

```python
#Program pokazuje obsługę dwóch typów wyjątków
#na przykładzie dzielenia dwóch liczb.
#Program wypisuje domyślny komunikat wyjątku.

def main():
    try:
        num1 = int(input('Podaj liczbę: '))
        num2 = int(input('Podaj następną liczbę: '))

        result = num1 / num2
        print(num1, 'dzielone przez', num2, 'daje', result)

    except ValueError as err:
        print(err)

    except ZeroDivisionError as err:
        print(err)

main()
```

Politechnika Wrocławska

Domyślny komunikat wyjątku

08_dzielenie.py | 09_dzielenie.py | 10_dzielenie.py

```
1  #Program pokazuje obsługę dwóch typów wyjątków
2  #na przykładzie dzielenia dwóch liczb.
3  #Program wypisuje domyślny komunikat wyjątku.
4
5  def main():
6      try:
7          num1 = int(input('Podaj liczbę: '))
8          num2 = int(input('Podaj następną liczbę: '))
9
10         result = num1 / num2
11         print(num1, 'dzielone przez', num2, 'daje', result)
12
13     #wszystkie typy wyjątków "pasują" do typu Exception
14     except Exception as err:
15         print(err)
16
17 main()
```

Politechnika Wrocławska

Klauzula else

- Konstrukcja try-except może być rozszerzona o klauzulę else
- Polecenia umieszczone w bloku else są wykonywane po poleceniach bloku try tylko wtedy, gdy nie zostanie zgłoszony żaden wyjątek

Politechnika Wrocławska

Klauzula else

11_odczyt_pliku.py

```
#Program pokazuje działanie konstrukcji:
#try-except-else

def main():
    total = 0.0

    try:
        infile = open('sales_data.txt','r')
        for line in infile:
            amount = float(line)
            total += amount

        infile.close()

    except Exception as err:
        print(err)

    else:
        print(format(total, '.2f'))

main()
```

Politechnika Wrocławska

Klauzula else

11_odczyt_pliku.py

```
1  #Program pokazuje działanie konstrukcji:
2  #try-except-else
3
4  def main():
5      total = 0.0
6
7      try:
8
9
10
11         spróbuj wykonać ten kod
12
13
14
15     except Exception as err:
16
17         obsłuż wyjątek
18     else:
19
20         wykonaj jeśli nie było wyjątków
21 main()
```

Politechnika Wrocławska

Klauzula else

11_odczyt_pliku.py | 12_dzielenie.py

```
1  #Program pokazuje działanie konstrukcji:
2  #try-except-else
3
4  def main():
5      try:
6          num1 = int(input('Podaj liczbę: '))
7          num2 = int(input('Podaj następną liczbę: '))
8          result = num1 / num2
9
10     except Exception as err:
11         print(err)
12
13     else:
14         print(num1, 'dzielone przez', num2, 'daje', result)
15
16
17 main()
```

Politechnika Wrocławska

Klauzula finally

- Konstrukcja try-except może być rozszerzona o klauzulę finally
- Polecenia umieszczone w bloku finally są wykonywane niezależnie od tego, czy został zgłoszony jakiś wyjątek

Politechnika Wrocławska

Klauzula finally

11_odczyt_pliku.py | 12_dzielenie.py | 13_dzielenie.py

```
1  #Program pokazuje działanie konstrukcji:
2  #try-except-finally
3
4  def main():
5      try:
6
7
8
9
10     except Exception as err:
11
12
13     else:
14                                                    esult)
15
16     finally:
17
18
19 main()
```

spróbuj wykonać ten kod

obsłuż wyjątek

wykonaj jeśli nie było wyjątków

wykonaj zawsze

Politechnika Wrocławska

Klauzula finally

11_odczyt_pliku.py | 12_dzielenie.py | 13_dzielenie.py

```
1  #Program pokazuje działanie konstrukcji:
2  #try-except-finally
3
4  def main():
5      try:
6          num1 = int(input('Podaj liczbę: '))
7          num2 = int(input('Podaj następną liczbę: '))
8          result = num1 / num2
9
10     except Exception as err:
11         print(err)
12
13     else:
14         print(num1, 'dzielone przez', num2, 'daje', result)
15
16     finally:
17         print('Koniec programu!')
18
19 main()
```

Politechnika Wrocławska

Podsumowanie (1)

- Funkcje pozwalające na otwarcie i zamknięcie plików:
  - `open()`
  - `close()`
- Funkcje pozwalające na zapis i odczyt danych tekstowych:
  - `write()`
  - `read()`
  - `readline()`
- Funkcja `rstrip()` służąca do pracy z łańcuchami znakowymi
- Wykorzystanie pętli `while` i `for` do pracy z plikami

Politechnika Wrocławska

Podsumowanie (2)

- Obsługa błędów
- Konstrukcja try-except
- Obsługa wielu wyjątków
- Domyślny komunikat wyjątku
- Klauzula else
- Klauzula finally