Politechnika Wrocławska

Programowanie proceduralne

INP001210WL

rok akademicki 2020/21

semestr letni

Wykład 7

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji

- 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

Obsługa błędów

- W wielu sytuacjach poprawność działania programu zależy od dostarczonych danych

Politechnika Wrocławska

Obsługa błędów

01_dzielenie.py

File Edit Format Run Options Window Help

```
#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

02_dzielenie.py

File Edit Format Run Options Window Help

```
#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

03_dzielenie.py

File Edit Format Run Options Window Help

```
#Program pokazuje obsługę wyjątku ValueError
#zgłoszonego przez funkcję int().

def main():
    try:
        #jeśli napisu nie można zamienić na liczbę całkowitą
        #to jest zgłaszany wyjątek ValueError
        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.')

    except ValueError:
        print('Błąd!')

main()
```

Konstrukcja try-except

03_dzielenie.py

File Edit Format Run Options Window Help

```
#Program pokazuje obsługę wyjątku ValueError
#zgłoszonego przez funkcję int().

def main():
    try:
        
        
        
        
        
        
        
    except ValueError:
        
main()
```

spróbuj wykonać ten kod

jeśli zostanie zgłoszony wyjątek ValueError – to przejdź tu

Konstrukcja try-except

03_dzielenie.py

File Edit Format Run Options Window Help

```
#Program pokazuje obsługę wyjątku ValueError
#zgłoszonego przez funkcję int().

def main():
    try:
        #jeśli napisu nie można zami
        #to jest zgłaszany wyjątek V
        num1 = int(input('Podaj licz
        num2 = int(input('Podaj nast

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

    except ValueError:
        print('Błąd!')

main()
```

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

Politechnika Wrocławska

Konstrukcja try-except

04_odczyt_pliku.py

File Edit Format Run Options Window Help

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

def main():
    filename = input('Podaj nazwę pliku: ')

    #funkcja open() zgłasza wyjątek, jeśli
    #nie istnieje plik o podanej nazwie
    infile = open(filename,'r')

    contents = infile.read()
    print(contents)
    infile.close()

main()
```

Politechnika Wrocławska

Konstrukcja try-except

05_odczyt_pliku.py

File Edit Format Run Options Window Help

```
#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

06_odczyt_pliku.py

File Edit Format Run Options Window Help

```
#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

06_odczyt_pliku.py

File Edit Format Run Options Window Help

```
    except IOError:
        print('Wystąpił błąd podczas odczytu pliku')

    except ValueError:
        print('W pliku znajdują się dane inne niż liczbowe')

    except:
        print('Wystąpił błąd.')

main()
```

Politechnika Wrocławska

Obsługa wielu wyjątków

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

Politechnika Wrocławska

Obsługa wielu wyjątków

- Klauzula except może też obsługiwać wszystkie wyjątki

Politechnika Wrocławska

Obsługa wielu wyjątków

07_odczyt_pliku.py

File Edit Format Run Options Window Help

```
#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

File Edit Format Run Options Window Help

```
#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

09_dzielenie.py

File Edit Format Run Options Window Help

```
#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

10_dzielenie.py

File Edit Format Run Options Window Help

```
#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)

    #wszystkie typy wyjątków "pasują" do typu Exception
    except Exception as err:
        print(err)

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

File Edit Format Run Options Window Help

```
#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()
```

Klauzula else

11_odczyt_pliku.py

File Edit Format Run Options Window Help

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

def main():
    total = 0.0

    try:
        spróbuj wykonać ten kod

    except Exception as err:
        obsłuż wyjątek
    else:
        wykonaj jeśli nie było wyjątków
main()
```

Politechnika Wrocławska

Klauzula else

12_dzielenie.py

File Edit Format Run Options Window Help

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

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

    except Exception as err:
        print(err)

    else:
        print(num1, 'dzielone przez', num2, 'daje', result)


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

13_dzielenie.py

File Edit Format Run Options Window Help

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

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

    except Exception as err:
        print(err)

    else:
        print(num1, 'dzielone przez', num2, 'daje', result)

    finally:
        print('Koniec programu!')

main()
```

Politechnika Wrocławska

Podsumowanie

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