Programowanie Python 1

(CP1S02005)

Politechnika Białostocka - Wydział Elektryczny

Cyfryzacja przemysłu, sem. II, studia stacjonarne I stopnia

Rok akademicki 2023/2024

Wykład nr 9 (08.05.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 9

- Pliki w Pythonie
  - otwarcie pliku - funkcja open()
  - zamknięcie pliku - metoda close()
  - odczyt z pliku tekstowego
  - zapis do pliku tekstowego
  - format JSON
  - format CSV

Python - pliki (otwarcie pliku)

- do otwarcia pliku służy wbudowana funkcja open()

```
open(file, mode='r', buffering=-1, encoding=None)
```

- file - nazwa pliku wraz ze ścieżką dostępu do pliku
- mode - tryb otwarcia pliku:
  - 'r' - otwiera plik do odczytu (domyślnie)
  - 'w' - otwiera plik do zapisu; jeśli plik już istnieje, jego zawartość zostanie usunięta; jeśli plik nie istnieje, zostanie utworzony nowy
  - 'a' - otwiera plik do dopisywania; nowa zawartość zostanie dodana na koniec istniejącej zawartości pliku
  - 'r+' - otwiera plik do odczytu i zapisu
  - 'b' - tryb binarny, używany do otwierania pliku w trybie binarnym (np. 'rb', 'wb', 'ab', 'r+b')

Python - pliki (otwarcie pliku)

- do otwarcia pliku służy wbudowana funkcja open()

```
open(file, mode='r', buffering=-1, encoding=None)
```

- buffering - określa, czy buforować dane; domyślnie jest to -1, co oznacza używanie domyślnych ustawień buforowania; można podać 0, aby wyłączyć buforowanie lub wartość większą od 0, aby określić wielkość bufora
- encoding - kodowanie pliku tekstowego; domyślnie jest to None, co oznacza używanie domyślnego kodowania dla stosowanego środowiska, np. 'utf-8', 'utf-16', 'utf-32', 'ascii'

- funkcja open() zwraca obiekt pliku, który jest używany do wykonywania operacji na pliku; obiekt ten jest zazwyczaj przypisywany do zmiennej

```
plik = open("nazwa_pliku.txt", "w")
```

Python - pliki (otwarcie pliku, przykłady)

- otwarcie pliku dane.txt do odczytu (domyślny tryb tekstowy)

```
plik = open("dane.txt", "r")
```

- otwarcie pliku dane.txt znajdującego się na dysku D w katalogu wyniki do dopisywania (domyślny tryb tekstowy)

```
plik = open("d:/wyniki/dane.txt", "a")
```

- otwarcie pliku dane.txt do odczytu i zapisu (domyślny tryb tekstowy)

```
plik = open("dane.txt", "r+")
```

- otwarcie pliku dane.dat do odczytu i zapisu w trybie binarnym

```
plik = open("wyniki/dane.dat", "rb+")
```

Python - pliki (otwarcie pliku)

- próba otwarcia nieistniejącego pliku do odczytu

```
plik = open("dane.txt", "r")
# operacje na pliku
plik.close()
```

spowoduje błąd i wywołanie wyjątku FileNotFoundError

```
Traceback (most recent call last):
  File "d:\myapp.py", line 1, in <module>
    plik = open("dane.txt", "r")
           ^^^^^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or
directory: 'dane.txt'
```

Python - pliki (otwarcie pliku)

- aby zabezpieczyć się przed otwarciem nieistniejącego pliku, można skorzystać z obsługi wyjątków (sekcja try-except)

```
try:
    plik = open("dane.txt", "r")
    # operacje na pliku
    plik.close()
except FileNotFoundError:
    print("Plik nie istnieje.")
```

```
Plik nie istnieje.
```

Python - pliki (zamknięcie pliku)

- do zamknięcia pliku służy metoda close(), która wywoływana jest na obiekcie pliku po zakończeniu operacji na nim

```
plik = open("dane.txt", "w")
# operacje na pliku
plik.close()
```

- zamykanie pliku jest ważne, ponieważ zwalnia zasoby systemowe używane przez plik i zapewnia, że wszystkie buforowane dane zostaną zapisane na dysku
- po wywołaniu metody close(), dalsze operacje na pliku za pomocą tego samego obiektu będą niemożliwe i spowodują wywołanie wyjątku: ValueError: I/O operation on closed file

Python - pliki (otwarcie i zamknięcie pliku)

- zamiast tradycyjnej metody otwarcia i zamknięcia pliku:

```
plik = open("dane.txt", "w")
# operacje na pliku
plik.close()
```

- zalecane jest korzystanie z konstrukcji with, która zapewnia automatyczne zamknięcie pliku po opuszczeniu bloku with

```
with open("dane.txt", "r") as plik:
    # operacje na pliku
```

- pomaga to uniknąć problemów związanych z niezamkniętymi plikami i zapewnia bardziej czytelny kod

Python - pliki (odczyt z pliku tekstowego)

- metoda read() odczytuje całą zawartość pliku jako pojedynczy ciąg znaków

```
with open("nazwa_pliku.txt", "r") as plik:
    zawartosc = plik.read()
    print(zawartosc)
```

- metoda readline() odczytuje kolejne linie tekstu z pliku

```
with open("nazwa_pliku.txt", "r") as plik:
    linia = plik.readline()
    while linia:
        print(linia)
        linia = plik.readline()
```

Python - pliki (odczyt z pliku tekstowego)

- metoda readlines() odczytuje wszystkie linie tekstu z pliku i zwraca je jako listę łańcuchów znakowych

```
with open("nazwa_pliku.txt", "r") as plik:
    linie = plik.readlines()
    for linia in linie:
        print(linia)
```

- można zastosować pętlę for do iteracji po pliku, każda linia pliku będzie traktowana jako kolejny element iterowalny

```
with open("nazwa_pliku.txt", "r") as plik:
    for linia in plik:
        print(linia, end="")
```

- ustawienie end="" oznacza, że print() nie będzie dodawać automatycznie znaku nowej linii na końcu każdej linii odczytanej z pliku, co zapobiega podwójnym odstępom między liniami

Python - pliki (odczyt z pliku tekstowego, przykład)

- zawartość pliku dane.txt

```
Jarosław Kamiński
Grażyna Wójcik
Piotr Wiśniewski
```

- wyświetlenie zawartości pliku z numerowaniem wierszy

```
with open("dane.txt", "r", encoding = "utf-8") as plik:
    numer_wiersza = 1
    linia = plik.readline()
    while linia:
        print(f"{numer_wiersza}: {linia}", end="")
        linia = plik.readline()
        numer_wiersza += 1
```

```
1: Jarosław Kamiński
2: Grażyna Wójcik
3: Piotr Wiśniewski
```

Python - pliki (odczyt z pliku tekstowego, przykład)

- pominięcie kodowania znaków spowoduje błędne wyświetlenie tekstu

```
with open("dane.txt", "r") as plik:
    numer_wiersza = 1
    linia = plik.readline()
    while linia:
        print(f"{numer_wiersza}: {linia}", end="")
        linia = plik.readline()
        numer_wiersza += 1
```

```
1: JarosŁ,aw KamiŁ„ski
2: GraŁŁyna WĂłjcik
3: Piotr WiŁ›niewski
```

- wynika to z innych stron kodowych przy tworzeniu pliku i jego wyświetlaniu

```
import os
os.system("chcp")
```

```
Active code page: 852
```

Python - pliki (odczyt z pliku tekstowego, przykład)

- plik tekstowy zawiera liczby całkowite
- obliczenie i wyświetlenie sumy oraz średniej arytmetycznej liczb w pliku

```
with open("liczby.txt", "r") as plik:
    suma = 0
    liczba_elementow = 0
    linia = plik.readline()
    while linia:
        liczba = int(linia)
        suma += liczba
        liczba_elementow += 1
        linia = plik.readline()
    srednia = suma / liczba_elementow
    print("Suma liczb z pliku:", suma)
    print("Średnia arytmetyczna liczb z pliku:", srednia)
```

```
23
18
-11
53
6
```

```
Suma liczb z pliku: 89
Średnia arytmetyczna liczb z pliku: 17.8
```

Python - pliki (odczyt z pliku tekstowego, przykład)

- suma i średnia liczb z pliku tekstowego + kontrola błędów

```
try:
    with open("liczby.txt", "r") as plik:
        suma = 0
        liczba_elementow = 0
        linia = plik.readline()
        while linia:
            liczba = int(linia)
            suma += liczba
            liczba_elementow += 1
            linia = plik.readline()
        srednia = suma / liczba_elementow
        print("Suma liczb z pliku:", suma)
        print("Średnia arytmetyczna liczb z pliku:", srednia)
except FileNotFoundError:
    print("Plik 'liczby.txt' nie został znaleziony.")
except ValueError:
    print("Błąd konwersji danych na liczbę całkowitą.")
except IOError:
    print("Wystąpił błąd podczas odczytu pliku.")
```

Python - pliki (zapis do pliku tekstowego)

- metoda write() służy do zapisywania pojedynczych linii lub fragmentów tekstu do otwartego pliku tekstowego

```
with open("zapis.txt", "w") as plik:
    plik.write("To jest pierwsza linia.\n")
    plik.write("To jest druga linia.\n")
```

- metoda writelines() służy do zapisywania wielu linii tekstu na raz, przyjmuje jako argument listę zawierającą linie tekstu

```
lines = ["Linia pierwsza.\n", "Linia druga.\n"]
with open("zapis.txt", "w") as file:
    file.writelines(lines)
```

Python - pliki (zapis do pliku tekstowego)

- funkcja print() ma opcjonalny parametr file, który umożliwia zapisywanie tekstu do określonego pliku zamiast na standardowe wyjście (np. ekran)

```
with open("zapis.txt", "w") as plik:
    print("Linia pierwsza.", file=plik)
    print("Linia druga.", file=plik)
```

- dane zapisywane do pliku mogą być formatowane

```
with open("zapis.txt", "w") as plik:
    imie = "Jan"
    wiek = 30
    plik.write(f"Imię: {imie}, Wiek: {wiek}\n")
```

```
Imię: Jan, Wiek: 30
```

Python - pliki (przykład zapisu do pliku tekstowego)

- zapisanie do pliku tekstowego 10 pseudolosowych liczb rzeczywistych z zakresu od 0 do 99 (z dokładnością do 3 cyfr po kropce dziesiętnej)

```
import random
los = [round(random.uniform(0, 99), 3) for _ in range(10)]
with open("losowe.txt", "w") as file:
    for liczba in los:
        file.write(f"{liczba:.3f}\n")
print("Liczby zostały zapisane.")
```

```
56.766
7.212
10.048
43.205
60.943
69.050
90.860
42.654
85.622
6.440
```

- funkcja random.uniform(a, b) z modułu random generuje pseudolosowe liczby rzeczywistej z przedziału [a, b]
- funkcja round(number, ndigit) zaokrągla liczby do określonej liczby miejsc po kropce, gdzie number to zaokrąglana liczba, a ndigits to liczba miejsc po kropce

Python - format JSON

- JSON (JavaScript Object Notation) jest formatem przechowywania i wymiany danych komputerowych opracowanym pierwotnie dla języka JavaScript (pliki z rozszerzeniem .json)
- typy stosowane przez JSON:
  - liczba (zgodne z formatem zmiennoprzecinkowym o podwójnej precyzji)

```
{
    "wiek" : 25,
    "wzrost" : 1.85
}
```

  - ciąg znaków (jako klucz nie należy stosować znaków diakrytycznych)

```
{
    "imie" : "Paweł",
    "nazwisko" : "Wiśniewski"
}
```

Python - format JSON

- typy stosowane przez JSON:
  - wartości logiczne (true, false)

```
{
    "dostepny" : true,
    "gotowy" : false
}
```

  - pusty literał

```
{
    "dane" : null
}
```

  - tabela

```
{
    "miasta" : ["Opole", "Toruń", "Krosno"]
}
```

Python - format JSON

- typy stosowane przez JSON:
  - obiekt

```
{
    "samochod" :
    {
        "marka" : "Opel",
        "model" : "Kadet",
        "rocznik" : 1998,
        "bezwypadkowy" : true
    }
}
```

- zaletą formatu JSON jest duża czytelność danych
- inne popularne formaty: CSV, XML, YAML

Python - format JSON

- w Pythonie do obsługi danych w formacie JSON służy moduł json
- funkcja json.dumps() służy do kodowania danych do formatu JSON

```
import json
data = {"name": "Jan", "age": 30, "city": "Kolno"}
json_string = json.dumps(data)
print(json_string)
```

```
{"name": "Jan", "age": 30, "city": "Kolno"}
```

```
import json
liczby = [2, 3, 5, 7, 11, 13]
json_string = json.dumps(liczby)
print(json_string)
```

```
[2, 3, 5, 7, 11, 13]
```

Python - format JSON

- funkcja json.loads() służy do dekodowania danych JSON do obiektów języka Python

```
import json
json_string = '{"name": "Jan", "age": 30, "city": "Kolno"}'
data = json.loads(json_string)
print(data)
```

```
{'name': 'Jan', 'age': 30, 'city': 'Kolno'}
```

Python - format JSON

- funkcje json.dumps() oraz json.loads() umożliwiają bezpośredni zapis i odczyt danych z pliku

```
import json

data = [1, 2, 3, 4, 5]

# Zapis danych do pliku JSON
with open("data.json", "w") as json_file:
    json.dump(data, json_file)

# Odczyt danych z pliku JSON
with open("data.json", "r") as json_file:
    loaded_data = json.load(json_file)

print(loaded_data)
```

```
[1, 2, 3, 4, 5]
```

Python - format CSV

- CSV (Comma-Separated Values) - prosty format przechowywania danych tabelarycznych, w którym każdy wiersz reprezentuje pojedynczy rekord, a wartości oddzielone są separatorem (najczęściej przecinkiem)
- separator może być również średnikiem, spacją, tabulatorem, itp.
- przykład pliku CSV (dane z oscyloskopu):

```
X,CH1,
Second,Volt,
-3.00000e-04,-3.20e-01,
-2.99000e-04,-6.40e-01,
-2.98000e-04,-7.20e-01,
-2.97000e-04,-1.04e+00,
```

- każdy wiersz danych jest oddzielany od następnego za pomocą znaku nowej linii (newline)
- pierwszy wiersz pliku CSV często zawiera nagłówki kolumn, które opisują zawartość danych w każdej kolumnie

Python - format CSV (przykład)

- lista imion męskich w rejestrze PESEL stan na 19.01.2024 - imię pierwsze

  (https://dane.gov.pl/pl/dataset/1667,lista-imion-wystepujacych-w-rejestrze-pesel-osoby-zyjace)

```
IMIĘ_PIERWSZE,PŁEĆ,LICZBA_WYSTĄPIEŃ
PIOTR,MĘŻCZYZNA,689313
KRZYSZTOF,MĘŻCZYZNA,641406
TOMASZ,MĘŻCZYZNA,536774
ANDRZEJ,MĘŻCZYZNA,533141
PAWEŁ,MĘŻCZYZNA,506006
MICHAŁ,MĘŻCZYZNA,492093
JAN,MĘŻCZYZNA,477845
MARCIN,MĘŻCZYZNA,449607
JAKUB,MĘŻCZYZNA,426709
ADAM,MĘŻCZYZNA,401674
ŁUKASZ,MĘŻCZYZNA,383582
MAREK,MĘŻCZYZNA,378001
...
```

Python - format CSV (przykład)

```
import csv

# Otwarcie pliku CSV do odczytu
with open('imie.csv', newline='', encoding="utf-8") as plik:
    # Utworzenie czytnika CSV
    reader = csv.DictReader(plik)

    # Wyświetlenie nagłówków kolumn
    print("Nagłówki kolumn:", reader.fieldnames)

    # Iteracja po każdym wierszu danych
    for row in reader:
        # Wyświetlenie danych w każdym wierszu
        print(row)
```

- csv.DictReader() tworzy czytnik CSV, który automatycznie interpretuje pierwszy wiersz jako nagłówki kolumn
- parametr fieldnames zawiera nagłówki kolumn jako listę
- każdy wiersz danych reprezentowany jest jako słownik, gdzie kluczami są nagłówki kolumn, a wartościami są odpowiadające im wartości

Python - format CSV (przykład)

- wynik uruchomienia programu

```
Nagłówki kolumn: ['IMIĘ_PIERWSZE', 'PŁEĆ', 'LICZBA_WYSTĄPIEŃ']
{'IMIĘ_PIERWSZE': 'PIOTR', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '689313'}
{'IMIĘ_PIERWSZE': 'KRZYSZTOF', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '641406'}
{'IMIĘ_PIERWSZE': 'TOMASZ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '536774'}
{'IMIĘ_PIERWSZE': 'ANDRZEJ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '533141'}
{'IMIĘ_PIERWSZE': 'PAWEŁ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '506006'}
{'IMIĘ_PIERWSZE': 'MICHAŁ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '492093'}
{'IMIĘ_PIERWSZE': 'JAN', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '477845'}
{'IMIĘ_PIERWSZE': 'MARCIN', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '449607'}
{'IMIĘ_PIERWSZE': 'JAKUB', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '426709'}
{'IMIĘ_PIERWSZE': 'ADAM', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '401674'}
{'IMIĘ_PIERWSZE': 'ŁUKASZ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '383582'}
{'IMIĘ_PIERWSZE': 'MAREK', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '378001'}
{'IMIĘ_PIERWSZE': 'MATEUSZ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '377011'}
{'IMIĘ_PIERWSZE': 'GRZEGORZ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '374820'}
{'IMIĘ_PIERWSZE': 'STANISŁAW', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '366180'}
{'IMIĘ_PIERWSZE': 'WOJCIECH', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '335311'}
{'IMIĘ_PIERWSZE': 'MARIUSZ', 'PŁEĆ': 'MĘŻCZYZNA', 'LICZBA_WYSTĄPIEŃ': '286869`}
```

Koniec wykładu nr 9

Dziękuję za uwagę!