Języki programowania – Python
Współpraca programu z tekstowym plikiem dyskowym

Marzena Nowakowska

Wydział Zarządzania i Modelowania Komputerowego
Politechnika Świętokrzyska

Budynek C, p. 3.21

spimn@tu.kielce.pl

Standardowa obsługa pliku

Python zapewnia wiele różnych sposobów wykonania operacji odczytu i zapisu pliku, które są zdefiniowane za pomocą funkcji dostępnych w różnych bibliotekach.

Do obsługi plików jest potrzebna specjalna **zmienna plikowa**, która pełni rolę uchwytu do pliku. Aby stworzyć zmienną plikową należy plik **otworzyć** (zapewnić do pliku dostęp poprzez obiekt będący ww. zmienną plikową). Wszelkie operacje związane z plikiem wykonuje się poprzez wywoływanie metod na rzecz tej zmiennej plikowej.

Plik tekstowy - plik o najprostszym formacie, zawierający dane w postaci alfanumerycznej. Jest czytelny dla człowieka i łatwo dostępny poprzez dowolny edytor tekstowy. Podstawowym formatem zapisu jest ***txt***.

Otwarcie pliku

open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)

Najczęstsze postaci wywołania: (1) **open**(file, mode='r') (2) **open**(file, mode='w')

Parametry:

file – nazwa pliku, który ma być otworzony (zmienna lub stała typu *str*), w szczególności zawierająca względną lub bezwzględną ścieżkę, przykładowo:

file = "dane.txt" *# ścieżka względna*

file = "C:\Python\Programy\Pliki\\dane.txt" *# ścieżka bezwzględna*

mode – tryb otwarcia pliku. Najczęściej stosowane tryby to: ***r*** (domyślny, do odczytu), ***w*** (do zapisu), ***a*** (do dołączenia informacji na koniec pliku). Jeżeli w definicji trybu doda się znak plus (+), to plik jest otwarty w trybie do odczytu i zapisu. Domyślnie formatem otwieranego pliku jest tekst (*t*). Jeżeli otwierany jest plik binarny, to w ciągu znaków definiujących tryb (*mode*) należy umieścić literę *b*).

encoding – sposób kodowania znaków, czyli to, jak tekst jest zapisywany i odczytywany z pliku. Aby polskie znaki były poprawnie zapisywane i odczytywane stosuje się kodowanie UTF-8. Niepodanie parametru implikuje zastosowanie kodowanie, które jest ustawione jako domyślne w SO.

Funkcja otwiera plik i zwraca zmienną plikową jako obiekt plikowy ustawiony w pozycji początkowej.

Po zakończeniu pracy z plikiem, należy go zamknąć za pomocą metody ***close()***.

Zapis pliku

print(*wartosc_do_zapisu*, sep=' ', end='\n', file=sys.stdout, flush=False)

file – zmienna plikowa identyfikująca plik do zapisu; domyślnie ekran (standardowy strumień wyjściowy *sys.stdout*),

sep – łańcuch wstawiany między wyświetlanymi wartościami, domyślnie spacja,

end – łańcuch dołączany po ostatniej wartości, domyślnie znak nowej linii,

flush – czy wymusić opróżnianie bufora strumienia, domyślnie *False*.

***zmienna_plikowa*.write**(*str_data*)

Metoda wywoływana na rzecz zmiennej plikowej (utworzonej poprzez *open*). Zapisuje daną typu tekstowego *str_data* do wskazanego pliku. Zwraca liczbę zapisanych znaków.

***zmienna_plikowa*.writelines**(*list_data*)

Metoda zapisuje elementy listy *list_data* do pliku. Elementy na liście muszą być typu *str*.

Po zakończeniu zapisu należy plik zamknąć za pomocą metody *close*(). Można wyczyścić bufor zapisu (powoduje przeniesienie danych z bufora do pliku) za pomocą metody *flush*(). Potem można kontynuować zapis (ew. sprawdzić, co się do tej pory zapisało).

Zapis pliku - print

Instrukcja wiążąca with służy do bezpiecznego zarządzania zasobami, takimi jak pliki, połączenia sieciowe i blokady. Struktura instrukcji w przypadku obsługi pliku:

with open (…) as *zmienna_plikowa*:
    instrukcja obsługi pliku 1
    instrukcja obsługi pliku 2
    …

Po zakończeniu bloku instrukcji, plik jest automatycznie zamykany, nawet jeżeli wystąpił błąd.

 # Plik jest tworzony w bieżącym folderze roboczym. ***Wariant z with***

```
with open("plik.txt", "w", encoding="utf-8") as f:
    print("To jest linia zapisana do pliku.", file=f)
    print("Linie zapisane do plik.txt")
```

 # Plik jest tworzony w bieżącym folderze roboczym. ***Wariant bez with***

```
f = open("plik.txt", "w", encoding="utf-8")
print("To jest linia zapisana do pliku.", file=f)
print("Linie zapisane do plik.txt", file =f)
f.close()
```

Zapis pliku, przykład 1 - write

```
wierszyk='''\
        Zbrodnia to niesłychana
        Pani zabiła pana
        Zabiwszy grzebie w gaju
        Na łączce przy ruczaju
...
'''
# ścieżka dojścia i nazwa pliku jako stała tekstowa
#f1 = open('D:/Python_MN/Programy_MN/CZ_03/Py_Dane/test.txt', 'w')
# poniżej: podanie z klawiatury nazwy pliku do zapisu, do zmiennej ‘plik’
plik=input("Podaj nazwę pliku do zapisu: ")
f1 = open('D:/Python_MN/Programy_MN/CZ_03/Py_Dane/'+plik, 'w')
# 'w' oznacza, że będzie zapis do pliku
f1.write(wierszyk)
f1.flush() # zapisanie wartości zmiennej 'wierszyk' w pliku (czyszczenie bufora wyjścia)
ile_znakow=f1.write("\nTo już koniec wierszyka")
# 23 znaki tekstu + 1 znak końca linii =  24 znaki
f1.write("\nW poprzedniej linii zapisano: "+ str(ile_znakow) +" znaków")
f1.close()
```

Uwaga: należy pamiętać o kodowaniu znaków

Zapis pliku, przykład 2 - write

Zapis do pliku za pomocą *write*:
(1) dana po danej
(2) wierszami

```
# uwaga: ustalić w środowisku Spyder folder roboczy
# (1)
f1 = open('Py_Dane\liczby_11.txt', 'w') # separator folderów: \
for n in range(11):
    f1.write(str(n))
f1.flush()  # czyszczenie bufora - przeniesienie zawartości do pliku
# (2)
f1 = open('Py_Dane/liczby_22.txt', 'w') # separator folderów: /
for n in range(11):
# aby dane były w kolejnych wierszach pliku należy
# do wyprowadzanej wartości dodać znak nowej linii
    f1.write(str(n)+ '\n')
f1.close()
```

Zapis pliku, przykład 3 - writelines

```
N= int(input("Ile obiektów wprowadzasz: "))
obiekty_1=[]
obiekty_2=[]
for n in range(N):
    obiekty_1.append((input("Podaj wartość obiektu %i: " %n)))
    obiekty_2.append(obiekty_1[-1] + "\n")

f1 = open('Py_Dane\lista_1.txt', 'w')    # uwaga na ścieżkę dojścia
f1.write("Elementy listy bez znaku końca wiersza\n")
f1.writelines(obiekty_1)
f1.flush()
f1.write("\nElementy listy ze znakiem końca wiersza\n")
f1.writelines(obiekty_2)
f1.close()
```

Odczyt pliku; metody zmiennej plikowej

***zmienna_plikowa*.read(*size=-1*)** Zacząć stc; 2026/11/09

Odczytuje z pliku jego zawartość od bieżącego położenia pliku (miejsca, w którym zostało przerwane ostatnie czytanie), w liczbie co najwyżej *size* bajtów (uwaga na znak końca pliku EOF – *end of file*). Jeżeli funkcja nie ma argumentu, to plik jest odczytywany do znaku EOF (domyślna wartość argumentu: -1). Metoda zwraca wczytany tekst – daną typu *str*.

***zmienna_plikowa*.readline(*size=-1*)**

Odczytuje z pliku jego zawartość od bieżącego położenia pliku, w liczbie co najwyżej *size* bajtów (uwaga na znak końca linii \n oraz EOF). Gdy brak parametru odczytuje całą linię (domyślna wartość argumentu: -1) z pliku. Końcowy znak nowej linii jest zachowany. Metoda zwraca wczytany wiersz – daną typu *str.*

***zmienna_plikowa*.readlines(*hint=-1*)**

Odczytuje kolejne linie z pliku i tworzy listę zawierającą tak odczytane wiersze. Końcowy znak nowej linii jest zachowany w każdym elemencie listy. Jeśli liczba zwróconych bajtów przekroczy liczbę wskazaną w *hint*, nie zostanie zwróconych więcej wierszy. Domyślna wartość *hint* równa -1 oznacza wszystkie wiersze od bieżącego położenia pliku).

W każdym przypadku, jeżeli przed odczytem wskaźnik pliku jest przed znakiem końca EOF, metoda zwraca pusty tekst ("").

Odczyt pliku, przykład

```
f2 = open('Py_Dane\wierszyk_1.txt', encoding="utf-8")
#  w ww. domyślnym parametrem otwarcia pliku jest 'r'
odczyt1=f2.read(); print("Odczyt1 read: ", odczyt1)
f2.seek(0)  # ustawienie pliku w pozycji początkowej
odczyt2=f2.readline(); print("Odczyt2 readline - pierwszy wiersz: ", odczyt2)
odczyt2=f2.readline(); print("Odczyt2 readline - drugi wiersz: ", odczyt2)
f2.seek(0)
odczyt3=f2.readlines(); print("Odczyt3 readlines: ", odczyt3)
f2.seek(0)
print("\nOdczyt4\nCZYTANIE W PĘTLI")
# f2 jest własnym iteratorem
for wers in f2:
    print(wers)
f2.close()
print("\nKoniec czytania pliku w pętli\n")
```

Obsługa pliku *csv*

- Format **CSV** opisuje plik tekstowy, w którym kolejne dane w wierszu są oddzielone przecinkami; *Comma Separated Value*. Jest to typ pliku, który wykorzystuje określoną strukturę do uporządkowania danych tabelarycznych. Zawiera drukowane znaki ASCII lub Unicode.
- Istnieje gotowy moduł ***csv*** Pythona służący do obsługi tego typu formatu. Implementuje klasy do odczytu i zapisu danych (np. tabelarycznych) w formacie CSV. Do odczytu i zapisu pliku w ww. formacie wykorzystuje się metody modułu działające na zmiennej plikowej (wygenerowanej za pomocą *open*):

  csv_writer = **csv.writer**(*csvfile* [, dialect='excel'] [*optional keyword args*])

  csv_reader = **csv.reader**(*csvfile* [, dialect='excel'] [*optional keyword args*])

  Parametry:

  csvfile – zmienna plikowa czyli zmienna, do której przypisano wynik funkcji *open*,

  dialect (opcjonalny) – nazwa dialektu (opis sposobu formatowania danych),

  optional keyword args – parametry formatujące nadpisujące te, które wyspecyfikowane w dialekcie, np. *delimiter*.

Zazwyczaj wywołuje się metody z wybranymi parametrami (por. następne slajdy).

Zapis pliku *csv*

- Zaimportować moduł *csv*.
- Zdefiniować obiekt - zmienną plikową organizującą dostęp do pliku; otworzyć plik do **zapisu** za pomocą funkcji *open*().
- Utworzyć obiekt klasy *writer* (np. poniżej *obiekt*) umożliwiający zapis formatu CSV, wywołując dla zmiennej plikowej funkcję modułu *csv*:

  obiekt = csv.writer(*csvfile*, delimiter = *separator*)

- Zapisać dane do pliku CSV, wywołując na rzecz obiektu klasy *writer* metody *writerow* lub *wrtierows*:

  obiekt.writerow(row) lub **obiekt.writerows**(*rows*)

  row – lista wartości składająca się z elementów zapisywanych do pliku separowanych przecinkiem

  rows – lista list określonych j.w.

- Zamknąć plik po zakończeniu zapisu – metoda *close*() zmiennej plikowej.

Zapis pliku *csv*, przykład

```
import csv
f = open('Py_Dane/lista_kwoty_a.csv', 'w')
# writer = csv.writer(f) # standardowym separatorem jest przecinek: ,
obk_writer =csv.writer(f, delimiter='\t')
naglowek = ['Imię', 'Nazwisko', 'Telefon', 'e-mail', 'Kwota']
obk_writer.writerow(naglowek)
f.flush()
dana=["Jan", "Kowalski", "101 102 103", "jan1.@tu.kielce.pl", 100]
obk_writer.writerow(dana)
dane=[["Adam", "Nawrocki", "201 202 203", "adam1.@tu.kielce.pl", 200],
      ["Maria", "Zawadzka", "301 302 303", "maria1.@tu.kielce.pl", 300],
      ["Edyta", "Górniak", "401 402 403", "edyta1.@tu.kielce.pl", 400],
      ["Jan", "Peszek", "501 502 503", "jan2.@tu.kielce.pl", 500],
      ["Maria", "Wolak", "601 602 603", "maria2.@tu.kielce.pl", 600],
      ["Janusz", "Podsiadło", "701 702 703", "janusz1.@tu.kielce.pl", 700]]
for el in dane:                          # zapis kolejnych elementów listy w pętli
    obk_writer.writerow(el)
obk_writer.writerows(dane)               # zapis całej listy (wszystkie elementy 'en bloc'
#  f.flush()
f.close()
```

Parametr *newline*

Jeżeli w funkcji *open* nie poda się wartości dla parametru *newline*, znaki \r\n (powrót karetki i nowy wiersz) są dodawane automatycznie przy operacji zapisu do pliku.

Moduł *csv* sam zarządza swoimi znakami końca linii (wiersza) i przy operacji zapisu sam dodaje znak nowego wiersza (\n). W ten sposób w zapisywanym pliku powstaje pusty wiersz.

Aby temu zapobiec, w funkcji *open* należy zdefiniować parametr jak następuje:

newline=""

Przykład

```
f = open('Py_Dane/lista_kwoty_2222.csv', 'w', newline="")
```

Odczyt pliku *csv*

- Zaimportować moduł *csv*.
- Zdefiniować obiekt - zmienną plikową organizującą dostęp do pliku; otworzyć plik do **odczytu** za pomocą funkcji *open*().
- Utworzyć obiekt klasy *reader* (np. poniżej *obiekt*) umożliwiający odczyt formatu CSV, wywołując dla zmiennej plikowej funkcję modułu *csv:*

  obiekt = csv.reader(*csvfile*, delimiter = *separator*)

- Obiekt czytający (klasy *reader*) jest iteratorem (więc obiektem iterowalnym). Każda iteracja po tym obiekcie zwraca wiersz pliku CSV (który może obejmować wiele wierszy wejściowych).
- Zamknąć plik po zakończeniu odczytu – metoda *close*() zmiennej plikowej.

Odczyt pliku *csv*, przykład

```
# Ustawienie katalogu roboczego dla programu
from os import chdir
chdir('C:\\0E1_Dydaktyka\\Politechnika\\Python\\Programy\\CZ_03')
import csv
kol1=[]; kol2=[]; kol3=[];
csvfile=open('Py_Dane\lista_odczyt.csv')  # uwaga: sprawdzić zawartość pliku przed czytaniem
readCSV = csv.reader(csvfile, delimiter= ';')
nrWiersza=0
for row in readCSV:
    nrWiersza+=1;     print(nrWiersza, row)
    if len(row) > 2:
        kol1.append(row[0]);  kol2.append(row[1]); kol3.append(row[2])
    elif len(row) == 2:
        kol1.append(row[0]); kol2.append(row[1])
    elif len(row) == 1:
        kol1.append(row[0])
print("Pierwsza kolumna danych:\n", kol1)
print("Druga kolumna danych:\n", kol2)
print("Trzecia kolumna danych:\n", kol3)
csvfile.close()
```

```
Natalia 10;Nowak 10;9
Natalia 11;Nowak 11;9
Natalia 12;Nowak 12
Natalia 13
```

Zad_05c-plikCSV_odczyt.py