Pliki i operacje I/O

poziom maturalny

Szymon Leszkiewicz

Wczytywanie plików tekstowych

W pythonie pliki tekstowe otwieramy za pomocą wbudowanej funkcji `open()`

Interesują nas jej dwa argumenty *file* oraz *mode* `open(file, mode)`

file to ścieżka do naszego pliku - jeżeli plik jest w tym samym folderze co program, wystarczy podać nazwę pliku

mode to opcjonalny string określający tryb w jakim chcemy otworzyć plik - domyślnie (czyli możemy nie podawać tego argumentu) jest to 'r'

Wartości mode jakie nas dodatkowo interesują to

`w` - *write*, zapis. Otwieramy plik, do którego będziemy zapisywać dane

`a` - *append*, dopisujemy dane do pliku.

ew. `x` - zwróci błąd, jeżeli plik istnieje

```
with open('plik1.txt') as plik:
  # Wczytywanie całego pliku jako jednego łańcucha znaków:
  data = plik.read()
  print(data)
  print("typ:",type(data))
```

```
sdfsv
asfa
wet
hfbv
fhwe5y
wwtdfsdg
fgdfg dfgdfg dfgdfg dfgdfg
typ: <class 'str'>
```

Przechodzenie po wierszach pliku:

```
#przydatne jeżeli w wierszu jest więcej danych
with open('plik1.txt') as plik:
  for line in plik:
    print(line.split())
```

```
['sdfsv']
['asfa']
['wet']
['hfbv']
```

```
['fhwe5y']
['wwtdfsdg']
['fgdfg', 'dfgdfg', 'dfgdfg', 'dfgdfg']
```

Wczytywanie i zapisywanie plików tekstowych jest bardzo proste. Warto jednak pamiętać o pewnych szczegółach.

Przede wszystkim instrukcja `with` w przykładach wyznacza kontekst, w którym można używać pliku. Gdy sterowanie wychodzi poza blok `with`, plik jest automatycznie zamykany.

Nie musisz stosować instrukcji with, jednak jeśli z niej nie korzystasz, pamiętaj o zamknięciu pliku:

```
a = open('plik1.txt')
dane = a.read()
a.close()
```

Przydatne funkcję przy pracy z otwieranie pliku:

`read()`

`readline()`

`readlines()`

`strip()`

`split()`

`splitlines()`

Dokumentacja funkcji `open()`

Zapis danych tekstowych

Aby zapisać plik tekstowy, należy wywołać funkcję `open()` w trybie `w`. Powoduje to usunięcie poprzedniej zawartości pliku (jeśli istniała) i jej zastąpienie.

Zapisywanie porcji danych tekstowych:

```
with open("wyniki.txt", 'w') as res:
  a = 5
  res.write(f"Wyniki obliczeń {a}")
```

O wiele wygodniejsze - przekierowanie wywołania `print()`

```
with open('wyniki.txt', 'w') as res:
  a = 5
  print("Wyniki obliczeń", a, file = res)
```

Możemy również skorzystać z trybu `a`, który za każdym razem dopisuje dane do końca pliku

Dodatek

Funkcje `map` oraz `filter`

Bardzo przydatne funkcje podczas otwierania plków.

Za pomocą `map` możemy na przykład zamienić wszystkie wartości w liście na liczby całkowite

```
with open('liczby1.txt') as plik:
  plik = plik.read().split()

  print("wczytaliśmy liczby jako str: ", plik)

  plik = list(map(int, plik))
  print("Po zamianie: ", plik)
```

```
wczytaliśmy liczby jako str:  ['1223', '1526', '457468', '233462', '23522']
Po zamianie:  [1223, 1526, 457468, 233462, 23522]
```

Albo na przykład zamienić liczby z systemu bin na liczby w systemie dziesiętnym

```
def bin2dec(a):
  return int(a, 2)

with open('binarne.txt') as plik:
  plik = plik.read().split()

  print("wczytaliśmy liczby jako str: \n", plik)

  plik = list(map(bin2dec, plik))
  print("Po zamianie: ", plik)
```

```
wczytaliśmy liczby jako str:
 ['101101', '101', '110110', '1101', '1101', '11101011', '11010101']
Po zamianie:  [45, 5, 54, 13, 13, 235, 213]
```

Ważne

podajemy tylko nazwę funkcji, która przyjmuje jeden argument, lub w miejscu funkcji deklarujemy funkcję lambda.

Szymon Leszkiewicz szymon.leszkiewicz10@gmail.com