Модель об'єкта файла в Python 3.X

В Python 3.X рядки типу str завжди зображають текст Юнікоду (символи ASCII чи багатобайтові символи), а рядки типів bytes і bytearray зображають прості двійкові дані. Python 3.X відрізняє файли, які містять текст і двійкові дані.

● Текстові файли зберігають текст, що складається з символів Юнікоду. Зміст текстових файлів в сценаріях завжди зображений рядками типу str – послідовностями символів. Для текстових файлів автоматично виконується перетворення символів кінця рядка, а до змісту файлів автоматично застосовуються операції кодування/декодування: дані кодуються в двійкове зображення при записуванні в файл і декодуються назад в Юнікод при читанні з файла, відповідно з вказаним або використаним за замовчуванням кодуванням.

● Двійкові файли зберігають звичайні 8-бітові байти. Зміст двійкових файлів в сценаріях завжди зображений рядками байтів, зазвичай об'єктом типу bytes. Для двійкових файлів не передбачено жодних перетворень даних при читанні і записуванні.

На практиці текстові файли використовують для зберігання дійсних текстових даних, а двійкові – для таких елементів, як запаковані двійкові дані, зображення, аудіодані, виконавчий програмний код тощо. Програмно ці два типи файлів відрізняють аргументом з рядком режиму, який передають функції open. Наприклад, 'rb', 'wb' означає, що файл має двійкові дані. Для будови нового змісту текстових файлів використовують звичайні рядки ('spam' чи bytes.decode()), а для двійкових файлів – рядки байтів (b'spam' чи str.encode()).

Об'єкт файла отримують функцією open(), і використовують методи читання даних (read, readline, readlines), запису даних (write, writelines), звільнення ресурсів (close), пересування по файлу (seek), примусового виштовхування буферів на диск (flush), отримання дескриптора файла (fileno), та інші.

Деякі інструменти для роботи з файлами і каталогами

Демонстрація функцій роботи з файлами і каталогами. Для демонстрації обрану лише частину функцій, які є в зазначених далі модулях. Для точного визначення функцій і повного розуміння їх поведінки треба переглядати офіційну документації від авторів таких функцій.

Далі за текстом рамкою позначені результати функцій, отримані відповідними операторами print().

```
#  модулі для роботи з файлами та ресурсами операційної системи
import os
import os.path
import glob

#  -------- операції з шляхами ----------

dirname = os.path.abspath('.')  # шлях від кореня до поточної папки
print(dirname)
```

```
D:\V_V\Python Lecture\PythonTest
```

```
filename = dirname + r'\test1.txt'   # об'єднати шлях з файлом в поточній папці
# або так: об'єднати параметри в шлях, використовуючи знак os.sep роздільника:
#filename = os.path.join(dirname, 'test1.txt')
print(filename)
```

```
D:\V_V\Python Lecture\PythonTest\test1.txt
```

```
head, tail = os.path.split(filename)  # розділити на шлях та ім'я файла
print(head, "    ", tail)
```

```
D:\V_V\Python Lecture\PythonTest      test1.txt
```

```
drive, tail = os.path.splitdrive(filename)  # відділити ім'я диску
print(drive, "    ", tail)
```
```
D:      \V_V\Python Lecture\PythonTest\test1.txt
```

```
root, ext = os.path.splitext('test1.txt')  # розділити ім'я файла
print(root, "    ", ext)
```
```
test1      .txt
```

```
#  або для цілого шляху:
root, ext = os.path.splitext(filename)  # відділити розширення імені
print(root, "    ", ext)
```
```
D:\V_V\Python Lecture\PythonTest\test1      .txt
```

```
#  ---------- операції з списками файлів -------------

ls = os.listdir('.')  # список всіх файлів і папок, які є в поточній папці
print(ls)
#  [. . . . .]  багато ☺

part = glob.glob('*.txt')  # список файлів, які відповідають заданому шаблону
print(part)
```
```
['increase.txt',  'kube.txt',  'L18-1.txt',  'minmaxmtdata.txt',  'mtdata.txt',
'test1.txt', 'uniontry.txt']
```

```
# -------------- характеристики файла --------------------

#  os.path.getsize(path)   # розмір файла в байтах
filename = os.path.join(os.path.abspath('.'), 'test1.txt')
sz = os.path.getsize(filename)
print("Розмір файла:", sz)
```
```
Розмір файла: 31
```

```
import time
#  os.path.getmtime(path)   # час останньої модифікації файла:
gt = os.path.getmtime(filename)
print("Зведений формат часу:", gt)  # зведений формат часу - треба перетворити
до іншої форми - модуль time
```
```
Зведений формат часу: 1446497903.62794
```
```
# перетворити в структуровану форму:
## sttm = time.gmtime(gt)  # не плутати - це глобальний час UTC
#  UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or
GMT)
sttm = time.localtime(gt)  # а це локальний - структура time.struct_time
print("Загальний список показників часу:\n", sttm)  # друкує загальний список
показників
```
```
Загальний список показників часу:
 time.struct_time(tm_year=2015,  tm_mon=11,  tm_mday=2,  tm_hour=22,  tm_min=58,
tm_sec=23, tm_wday=0, tm_yday=306, tm_isdst=0)
```
```
# друкуємо по одному показнику, використовуючи структуру  time.struct_time
#  структура time.struct_time - це об'єкт з інтерфейсом іменованого кортежу
print("Файл створений: рік {0}, місяць {1}, день {2}".format(sttm.tm_year,
sttm.tm_mon, sttm.tm_mday))
# див. визначення структури time.struct_time
```
```
Файл створений: рік 2015, місяць 11, день 2
```

```
# ---------------- поточний час ----------------------

import datetime  # розділені показники часу
import time
# today()  - на даний момент
print("Сьогодні: рік {0}, місяць {1}, день {2}".
       format(datetime.date.today().year,   \
       datetime.date.today().month, datetime.date.today().day))
```

```
Сьогодні: рік 2017, місяць 10, день 1
```

```
#  а також  hour, minute, second
# поточний час може бути потрібним для обчислення різниці від часу останньої
#   модифікації файла
#  time.time()  - поточний час зведений
filename = os.path.join(os.path.abspath('.'), 'test1.txt')
# обчислюємо різницю в часі:
timediff = time.time() - os.path.getmtime(filename)
# так само переводимо в структуровану форму:
diff = time.localtime(timediff)
# треба ще відняти базовий час системи
basetime = time.localtime(os.times().user)
#print(basetime.tm_year,  basetime.tm_mon,  basetime.tm_mday)    #  базовий  час
системи
#print(basetime)
print("Минуло  часу:  років  {0},  місяців  {1},  днів  {2}".format(diff.tm_year-
basetime.tm_year,   \
      diff.tm_mon-basetime.tm_mon, diff.tm_mday-basetime.tm_mday))
```

```
Минуло часу: років 1, місяців 10, днів 29
```