Biblioteka
część II

Uniwersytet Ekonomiczny w Poznaniu

25 kwietnia 2014

Zadanie

Dla podanej wcześniej definicji biblioteki, zaimplementuj

1. Dodawanie historii przeprowadzania akcji na egzemplarzach:
   - dodania
   - wypożyczenia
   - oddania
2. Naliczanie kar bibliotecznych.

Na Moodle dostępne jest rozwiązanie poprzedniego zadania z biblioteki, które można zmodyfikować zgodnie z instrukcjami do tego zadania.

Zadanie 1

- Podczas dodawania egzemplarzy książki, podawana jest także data jego dodania do biblioteki, jako krotka, np. `(2013, 12, 23)`.
- Wyświetl zestawienie egzemplarzy książek, które zostały dodane później, niż podana data (czyli zestawienie **nowszych** egzemplarzy).
- Wejście:
  - W pierwszej linijce podawana liczba książek do dodania (n).
  - W kolejnych n linijkach podane egzemplarze książki.
  - W ostatniej linijce podana data (jako krotka).
- Wyjście: analogiczne zestawienie, jak w poprzednim zadaniu z biblioteki.

Zadanie 1 c.d.

Przykładowe wejście:

```
5
("Chatka Puchatka", "Alan A. Milne", 2014, (2014, 4, 10))
("Quo Vadis", "Henryk Sienkiewicz", 2010, (2014, 1, 15))
("Chatka Puchatka", "Alan A. Milne", 1998, (2013, 12, 31))
("Pan Tadeusz", "Adam Mickiewicz", 2003, (2012, 1, 1))
("Quo Vadis", "Henryk Sienkiewicz", 2010, (2014, 1, 15))
(2013, 12, 31)
```

Przykładowe wyjście:

```
('Chatka Puchatka', 'Alan A. Milne', 1)
('Quo Vadis', 'Henryk Sienkiewicz', 2)
```

Zadanie 2

Zaimplementuj naliczanie kar bibliotecznych.

- Reguły naliczania kar:
  - Każdy egzemplarz ma limit wypożyczenia 7 dni.
  - Za każdy dzień przekroczenia limitu naliczana jest kara w wysokości 50 groszy.
  - Wszystkie kary sumują się.
- Na wejściu podane są akcje z datami ich wykonania; w ostatniej linijce bieżąca (końcowa) data. Akcje podane są chronologicznie.
- Na wyjściu:
  - Najpierw wyświetlone są wartości logiczne, opisujące poprawność wykonania operacji (analogicznie do poprzedniego zadania z biblioteki).
  - Następnie wyświetlone jest zestawienie kar wszystkich czytelników (czytelnicy sortowani są rosnąco po nazwiskach). Jeśli czytelnik nie ma kar, nie jest wyświetlany. Nie trzeba zaokrąglać kar, ani wyświetlać ich w specjalny sposób. Krotki z karami wyświetlane są jedna pod drugą, podobnie jak w przypadku wyświetlania zestawienia egzemplarzy.

Zadanie 2 c.d.

Przykładowe wejście

```
11
("dodaj","Pan Tadeusz","A. Mickiewicz",2000,(2014,1,1))
("dodaj","Quo Vadis","H. Sienkiewicz",2010,(2014,1,1))
("dodaj","Chatka Puchatka","A.A. Milne",1998,(2014,1,1))
("dodaj","Pan Tadeusz","A. Mickiewicz",2000,(2014,1,1))
("dodaj","Chatka Puchatka","A.A. Milne",2014,(2014,1,1))
("wypozycz","Bartek Perkowski","Pan Tadeusz",(2014,1,25))
("wypozycz","Bartek Perkowski","Pan Tadeusz",(2014,2,2))
("wypozycz","Jacek Malyszko","Quo Vadis",(2014,2,12))
("oddaj","Jacek Malyszko","Quo Vadis",(2014,2,17))
("oddaj","Bartek Perkowski","Pan Tadeusz",(2014,2,25))
("wypozycz","Bartek Perkowski","Quo Vadis",(2014,3,5))
(2014,3,13)
```

Zadanie 2 c.d.

Przykładowe wyjście:

```
True                                                    1
True                                                    2
True                                                    3
True                                                    4
True                                                    5
True                                                    6
False                                                   7
True                                                    8
True                                                    9
True                                                    10
True                                                    11
('Bartek Perkowski', 12.5)                              12
```

Wskazówki

- Warto wykorzystać bibliotekę do obsługi dat.
- Dokumentacja:
  `https://docs.python.org/3.0/library/datetime.html`
- Tutorial: `http://pymotw.com/2/datetime/` (tutorial jest dla Pythona 2, ale moduł działa podobnie dla Pythona 3)
- Przykład wykorzystania w konsoli Pythona 3:

```
>>> import datetime
>>> d1 = datetime.date(2008, 3, 12)
>>> print(d1)
2008-03-12
>>> d2 = datetime.date(2009, 1, 21)
>>> (d1 - d2).days
-315
```

Sposób rozwiązania zadania 1

Ponieważ w trzech grupach zadanie pierwsze rozwiązywane było podczas zajęć, na kilku kolejnych slajdach zaprezentowano krok po kroku przykładowy sposób rozwiązania tego zadania. Opis powinien być bardzo przydatny studentom, którzy nie mieli tych zajęć (godziny rektorskie) oraz dla tych, którzy nie zrobili tego podczas zajęć.

Krok 1. Wczytywanie danych

W pliku z rozwiązaniem poprzedniego zadania, który znajduje się na Moodle, są następujące linijki:

```
biblioteka = Biblioteka()
for i in range(int(input())):
    t = eval(input())
    if t[0] == "dodaj":
        print(biblioteka.dodaj_egzemplarz_ksiazki(t[1], t[2], t[3]
    elif t[0] == "wypozycz":
        print(biblioteka.wypozycz(t[1], t[2]))
    elif t[0] == "oddaj":
        print(biblioteka.oddaj(t[1], t[2]))
```

Kod ten jest przystosowany do zadania nr. 2 z poprzedniej biblioteki. Najpierw tworzona jest instancja biblioteki, a następnie wczytywane są wszystkie krotki z opisywanymi akcjami, po czym określone akcje są wykonywane i wyświetlany jest ich rezultat (czy się powiodły, czy nie).

Krok 1. Wczytywanie danych

Linie od 4 do 9 z poprzedniego slajdu najlepiej zakomentować. Przydadzą się później w zadaniu z naliczaniem kar bibliotecznych (najlepiej wykorzystać działający kod z zadania 1 w zadaniu 2 – tam też będzie ważne zapisywanie egzemplarzy z datą dodania).

Natomiast po liniach 1 – 3 należy:

1. dodać egzemplarz książki z datą dodania,
2. wczytać datę, dla której generowany będzie raport,
3. wyświetlić raport nowszych egzemplarzy, niż podana data.

Zostanie to szczegółowo omówione na kolejnych slajdach.

Krok 2. Dodawanie egzemplarza książki

Za dodanie egzemplarza do biblioteki służy metoda:

```
dodaj_egzemplarz_ksiazki(self, tytul, autor, rok_wydania)
```

Najlepiej rozszerzyć tę metodę o nowy parametr z datą:

```
dodaj_egzemplarz_ksiazki(self, tytul, autor, rok_wydania, data)
```

Zatem dodanie egzemplarza książki na podstawie krotki `t` mogłoby wyglądać w następujący sposób (podane są dwie alternatywne formy):

```
biblioteka.dodaj_egzemplarz_ksiazki(t[0], t[1], t[2], t[3])
biblioteka.dodaj_egzemplarz_ksiazki(*t)
```

Gwiazdka przed krotką powoduje „rozpakowanie" krotki, czyli w przypadku powyższego wywołania, jeśli krotka `t` ma cztery elementy, to wywołanie `*t` w metodzie `dodaj_egzemplarz_ksiazki` spowoduje podanie tych czterech kolejnych elementów jako cztery kolejne argumenty w metodzie.

Krok 2a. Modyfikacja `dodaj_egzemplarz_ksiazki`

Istniejąca metoda `dodaj_egzemplarz_ksiazki` działa w następujący sposób:

1. Do zmiennej `ksiazka` przypisywana jest książka, która wcześniej została dodana do „bazy" – słownika z książkami.
2. Jeśli ta instrukcja się nie powiedzie – czyli jeśli nie można odnaleźć książki o tym tytule w „bazie", to znaczy, że należy dodać nową książkę. Zatem w kolejnych krokach tworzony jest nowy obiekt książki i przypisywany do zmiennej `ksiazka`. Następnie ten obiekt jest dodawany do „bazy".
3. Każda książka posiada listę egzemplarzy. W kolejnej linijce do takiej listy dodawany jest nowy, utworzony obiekt egzemplarza książki.
4. Zwracane jest `True`, oznaczające, że akcja się powiodła.

W tym wypadku modyfikacja powinna polegać na dodaniu parametru `data` do deklaracji metody oraz na przekazaniu tego parametru do konstruktora egzemplarza. Zatem utworzenie obiektu egzemplarza powinno wyglądać następująco:

```
Egzemplarz(rok_wydania, ksiazka, data)                    1
```

Krok 2b. Modyfikacja konstruktora egzemplarza

W konstruktorze egzemplarza należy dodać nowy parametr: `data` i przypisać go jako atrybut nowego obiektu. To jest też dobre miejsce, żeby skorzystać z rekomendowanej biblioteki do obsługi dat. Zatem na początku pliku należy zaimportować odpowiedni moduł:

```
import datetime                                                    1
```

Z kolei samo utworzenie atrybutu i przypisanie daty mogłoby wyglądać następująco:

```
self.data_dodania = datetime.date(*data)                           1
```

Ponieważ `data` występuje jako krotka z trzema elementami, np. `(2013, 4, 23)`, to użycie gwiazdki przed datą spowoduje, że zostanie wywołane coś podobnego do: `datetime.date(2013, 4, 23)`. Czyli krotka zostanie „rozpakowana".

Po wykonaniu tych kroków nasza biblioteka będzie zapamiętywać daty dodania egzemplarza.

Krok 3. Wczytanie daty

Aby wczytać datę, wystarczy wywołać np.:

```
data = eval(input())
```

Data ta następnie zostanie użyta przy wyświetleniu raportu nowszych książek.

Krok 4. Wyświetlenie raportu nowszych egzemplarzy

Aktualnie w bibliotece znajduje się metoda `raport_ksiazek`:

```
def raport_ksiazek(self):                                          1
    lista = []                                                     2
    for k in self.__ksiazki.values():                              3
        lista.append((k.tytul, k.autor, len(k.egzemplarze)))       4
    return sorted(lista, key = lambda t: t[0])                     5
```

W tej metodzie najpierw tworzona jest pusta lista, a następnie do tej listy dodawane są krotki zawierające tytuł książki, autora książki i liczbę egzemplarzy. Na końcu zwracana lest lista krotek posortowana po pierwszych elementach krotek (czyli po tytułach).

Dobrym sposobem byłoby zmodyfikowanie tej metody, żeby zwrcała raport nie wszystkich egzemplarzy, ale raport egzemplarzy nowszych niż określona data.

Krok 4a. Modyfikacja metody `raport_ksiazek`

Aby zmodyfikować metodę `raport_ksiazek` należy:

1. Dodać nowy argument do metody, reprezentujący datę o nazwie, przykładowo, `data`.
2. Zamiast liczby wszystkich egzemplarzy interesuje nas liczba wszystkich egzemplarzy nowszych niż określona data. Zatem najlepiej zamiast wywołania `k.egzemplarze`, wywołać nową metodę (którą oczywiście należy dopisać do klasy `Ksiazka`). Warto także wynik wywołania tej metody zapisać w nowej zmiennej, np.:

```
nowsze = k.egzemplarze_nowsze_niz(data)                    1
```

3. Dodatkowo, jeśli liczba egzemplarzy nowszych wyniesie 0, nie umieszczamy tej pozycji w raporcie, zatem do zmiennej `lista` krotki powinny być dodawane tylko wtedy, gdy `len(nowsze) > 0`

Krok 4b. Dodanie metody egzemplarze_nowsze_niz

Do klasy `Ksiazka` powinna zostać dodana metoda `egzemplarze_nowsze_niz` o następującej deklaracji:

```
def egzemplarze_nowsze_niz(self, data):
```

Metoda powinna zwracać tylko te egzemplarze, które są nowsze niż podana data, co można osiągnąć w dwóch linijkach, stosując listy składane:

```
data = datetime.date(*data)
return [e for e in self.egzemplarze if e.data_dodania > data]
```

Tym samym metoda `raport_ksiazek` jest już w stanie generować zestawienie egzemplarzy książek nowszych niż podana data. Ostatnim krokiem jest ich prawidłowe wyświetlenie.

Krok 4c. Wyświetlenie krotek z raportu

Metoda `raport_ksiazek` zwracała posortowaną listę krotek. Na wyjściu jednak nie jest oczekiwane wyświetlenie całej listy w jednej linijce, lecz wyświetlenie kolejnych krotek w nowych linijkach. Tym samym należy przeiterować wszystkie krotki z tej listy wyświetlić je w kolejnych wywołaniach funkcji `print`:

```
for x in biblioteka.raport_ksiazek(data):
    print(x)
```

Po wykonaniu tych kroków zgodnie z instrukcjami pierwsze zadanie powinno przejść testy na sprawdzarce.