Zadanie Urodziny –LOGIA 19 SP (2018/19), etap 3

Treść zadania

Bartek co roku obchodzi hucznie swoje urodziny. W tym roku postanowił, że będzie także celebrował swoje urodziny w te dni, które są odległe od jego dnia urodzin o wielokrotność 1000 dni. Niestety wyznaczenie dat, w których obchodziłby kolejne „tysiącznice" jest poza jego możliwościami. Szczególnie, że musi uwzględnić lata przestępne tj. takie, które są podzielne przez 4 i nie są podzielne przez 100 lub są podzielne przez 400. Bartek zapisuje daty w postaci słowa *uddmmrrrr*, gdzie *u* jest stałe, *dd* to dwucyfrowy numer dnia, *mm* – dwucyfrowy numer miesiąca, *rrrr* – czterocyfrowy numer roku (np. u13022004 oznacza datę 13 lutego 2004 roku).

Pomóż Bartkowi i napisz dwuparametrową funkcję **urodziny**, której pierwszym parametrem jest słowo określające datę urodzin Bartka, a drugim liczba nie mniejsza niż 1 i nie większa niż 100 określająca, którą „tysiącznicę" Bartek chce wyznaczyć. Wynikiem jest słowo określające datę, kiedy wypada kolejna „tysiącznica" określona drugim parametrem. Przyjmij, że pierwszy parametr i wynik tworzą poprawną datę z zakresu 01.01.1901 – 31.12.2499.

Przykłady:

wynikiem **urodziny('u13022004', 2)** jest **'u05082009'**, .

wynikiem **urodziny('u15011905', 10)** jest **'u02061932'**.

Omówienie rozwiązania

Zadanie Urodziny porusza często spotykany podczas programowania problem obliczania dat. Rozwiązanie wykorzystujące bibliotekę `datetime` jest proste, ale podczas konkursu nie można z niej korzystać (dozwolone są tylko biblioteki `turtle` i `math`). Z problemem wyliczania dat należy się zmierzyć bez pomocy funkcji bibliotecznych.

Odnotujmy podstawowe fakty o kalendarzu:

- liczba dni w kolejnych miesiącach roku jest różna,
- miesiąc luty dodatkowo może mieć 29 dni. Dzieje się tak w latach, które są podzielne przez 4, ale niepodzielne przez 100 – chyba, że są podzielne przez 400 – czyli lata 2100, 2200, 2300 nie będą przestępnymi.

Zadanie to rozwiążemy pisząc funkcję, która zlicza liczbę dni od pewnej ustalonej daty (np. 1 stycznia roku zerowego, której to daty w rzeczywistości nie było) oraz drugą funkcję, która jest funkcją odwrotną do pierwszej, czyli dla danej liczby dni zwraca datę.

W pseudokodzie pierwsza funkcja mogłaby wyglądać tak:

```
ndnia ← 365*rok + rok div 4 – rok div 100 + rok div 400 + md[m-1]+d
jeśli przestępny(rok) i m < 3
    ndnia ← ndnia – 1
wynik ndnia
```

W pierwszej linii obliczamy liczbę dni od wspomnianej daty 1 stycznia roku zerowego z uwzględnieniem lat przestępnych. Posługujemy się tablicą `md`, której kolejne elementy wynoszą [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365] – jest to liczba dni, jakie upłynęły w poprzednich miesiącach danego roku i tutaj nie liczymy lat przestępnych. Zakładamy, że luty ma 28 dni. W drugiej linii korygujemy `ndnia`, gdy mamy rok przestępny, lecz nie było jeszcze 1 marca.

Funkcja **przestępny(rok)** mogłaby wyglądać tak:

```
zwróć wartość wyrażenia logicznego:
(rok mod 4 = 0) oraz ((rok mod 100 <> 0) lub (rok mod 400 = 0))
```

Wynikiem jest PRAWDA, gdy wartość `rok` odpowiada rokowi przestępnemu. Operatory `div` i `mod`, to odpowiednio dzielenie całkowitoliczbowe i reszta z dzielenia.

Funkcja realizująca obliczenia w drugą stronę, tzn. z danego numeru dnia wyliczająca datę podzielona została na dwie części. W części pierwszej jest wyznaczany rok, w którym data się znajduje – są to kroki 1-4 z listingu poniżej. W drugiej części wyliczany jest miesiąc i dzień tego roku.

```
rok  ← ndnia // 365
dopóki ndnia – (365 * rok + rok // 4 - rok // 100 + rok // 400) <= 0
    rok ← rok – 1
ndnia ← ndnia - (365 * rok + rok div 4 - rok div 100 + rok div 400)
jeśli ndnia > 365
    ndnia ← ndnia – 366
    rok ← rok + 1
jeśli przestępny(rok)
    jeśli ndnia = 31 + 28
          wynik 29, 2, rok
    jeśli ndnia < 31 + 29
          ndnia ← ndnia + 1
miesiąc ← 1
dopóki ndnia – md[miesiąc] > 0
    miesiąc ← miesiąc + 1
ndnia ← ndnia – md[miesiąc – 1]
wynik ndnia, miesiąc, rok
```

Gdyby każdy rok zawierał 365 dni, to obliczenie roku zakończylibyśmy wyrażeniem:

```
rok = ndnia // 365
```

Z uwagi, że część dni z puli o wartości `ndnia` znajduje się w latach przestępnych należy skorygować tak obliczoną wartość zmiennej `rok`. Zrobione to zostało w pętli z kroku 2. Pętla dla podanych zakresów lat wykona się dwa razy – gdyż liczba lat przestępnych w latach 1-1901 i 1-2499 są odpowiednio równe 460 i 606.

Krok 4 zawiera korektę, gdy data wypada 1 stycznia roku przestępnego. Dzień ten już był odejmowany w kroku 3, a w rzeczywistości nie powinniśmy go odejmować. Podobna korekta jest w kroku 5, lecz tu nie musimy już zmieniać roku – wystarczy odjąć jeden od `ndnia`.

W krokach 6-7 wyznaczamy numer miesiąca wykorzystując wcześniej omawianą tablicę `md`. W końcu po odjęciu od `ndnia` liczby dni, jaka upłynęła od początku roku do końca poprzedniego miesiąca, otrzymujemy numer dnia w miesiącu.

Mając te dwie funkcje obliczenie „tysiącznicy" sprowadza się do wywołania:

```
ndnia2data( data2ndnia(data) + n * 1000 )
```

Pozostaje jeszcze sformatowanie zwróconej daty zgodnie ze specyfikacją.

Rozwiązanie w języku Python

```
1.  def data2ndnia(data):
2.    d = int(data[1:3])
3.    m = int(data[3:5])
4.    r = int(data[5:])
5.    ndnia = 365 * r  + md[m - 1] + d + r // 4 - r // 100 + r // 400
6.    if r % 4 == 0 and (r % 100 != 0 or r % 400 == 0) and m < 3: ndnia -= 1
7.    return ndnia
```

W liniach 2-4 pobieramy z łańcucha tekstowego z datą odpowiednie fragmenty z dniem miesiąca, miesiącem i rokiem. W linii 5 obliczamy numer dnia i ewentualnie korygujemy, gdy mamy rok przestępny i data jest przed 1 marca.

Wypełnienie tablicy `md` zrealizowano za pomocą kodu:

```
8.  md = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 ]
9.  for i in range(1, len(md)):
10.   md[i] = md[i - 1] + md[i]
```

Realizacja funkcji odwrotnej, czyli z numeru dnia wyznaczającej datę wygląda następująco:

```
11. def ndnia2data(ndnia):
12.   r = ndnia // 365
13.   while ndnia - (365 * r + r // 4 - r // 100 + r // 400) <= 0:
14.     r -= 1
15.   ndnia = ndnia - (365 * r + r // 4 - r // 100 + r // 400)
16.   if ndnia > 365:
17.     ndnia -= 366
18.     r += 1
19.   if r % 4 == 0 and (r % 100 != 0 or r % 400 == 0):
20.     if ndnia == 31 + 28: return 29, 2, r
21.     if ndnia < 31 + 29: ndnia += 1
22.   m = 1
23.   while ndnia - md[m] > 0: m += 1
24.   ndnia = ndnia - md[m - 1]
25.   return ndnia, m , r
```

Zwracana jest trzyelementowa krotka z dniem miesiąca, miesiącem i rokiem.

Obie wyżej omówione funkcje oraz tablicę `md` umieszczono wewnątrz funkcji **urodziny**. Z krotki, którą zwraca funkcja `ndnia2data`, jest w linii 27 formatowany łańcuch z pierwszym znakiem 'u', a następnie dwie dwuznakowe i jeden czteroznakowy z dopełnianiem zerami wiodącymi np. 01082003. Operator * w funkcji `format` służy do rozpakowania krotki na trzy argumenty.

```python
def urodziny(data, n):
    md = [0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 ]
    for i in range(1, len(md)):
        md[i] = md[i - 1] + md[i]
    def data2ndnia(data):
        d = int(data[1:3])
        m = int(data[3:5])
        r = int(data[5:])
        ndnia = 365 * r  + md[m - 1] + d + r // 4 - r // 100 + r // 400
        if r % 4 == 0 and (r % 100 != 0 or r % 400 == 0) and m < 3: ndnia -= 1
        return ndnia
    def ndnia2data(ndnia):
        r = ndnia // 365
        while ndnia - (365 * r + r // 4 - r // 100 + r // 400) <= 0:
            r -= 1
        ndnia = ndnia - (365 * r + r // 4 - r // 100 + r // 400)
        if ndnia > 365:
            ndnia -= 366
            r += 1
        if r % 4 == 0 and (r % 100 != 0 or r % 400 == 0):
            if ndnia == 31 + 28: return 29, 2, r
            if ndnia < 31 + 29: ndnia += 1
        m = 1
        while ndnia - md[m] > 0: m += 1
        ndnia = ndnia - md[m - 1]
        return ndnia, m , r
    wy = ndnia2data( n * 1000 + data2ndnia(data))
    return 'u{:02}{:02}{:04}'.format(*wy)
```

Testy

Testy zostały tak dobrane, by sprawdzić obliczenia bez lat przestępnych, z latami przestępnymi oraz zawierały lata podzielne przez 100. Były też testy, w których poszukiwane daty wypadały 29 lutego lub 1 stycznia.

Wywołanie – Python	**Wynik**
urodziny("u06042093", 1)	"u01012096"
urodziny("u05041989", 1)	"u31121991"
urodziny("u16072377", 10)	"u01122404"
urodziny("u28071987", 6)	"u31122003"
urodziny("u03062337", 1)	"u28022340"
urodziny("u30122380", 7)	"u29022400"
urodziny("u13121901", 3)	"u01031910"
urodziny("u30122000", 7)	"u29022020"
urodziny("u02081991", 94)	"u12122248"
urodziny("u20111976", 11)	"u02012007"