8 października 2008

Kurs języka Python

Lista 1.

Zadanie 1. Zaprogramuj

1. funkcję Ackermanna

$$Ack(n,m) = \begin{cases} m+1 & \text{gdy } n = 0 \\ Ack(n-1,1) & \text{gdy } m = 0 \\ Ack(n-1, Ack(n, m-1)) & \text{w pozostałych przypadkach} \end{cases}$$

2. trójargumentową funkcję, gdzie pierwszy argument to rok, drugi: miesiąc, trzeci: dzień miesiąca (podawane jako liczby całkowite), która oblicza ile zostało dni do Sylwestra 2006 od daty podanej jako argument;

3. funkcję, której argumentem jest lista list liczb całkowitych, a wynikiem suma wszystkich liczb. Nie jest wymagana konrola poprawności typów;

4. funkcję, której argumentami są dwie trójelementowe listy liczb całkowitych. W listach pamiętane są daty w postaci *[dd, mm, rrrr]*. Wynikiem jest liczba dni dzielących te daty. Możesz przyjąć, że argumenty są poprawnymi datami;

5. funkcję, której argumentami są dwa wektory podane jako listy liczb identycznej długości, a wynikiem iloczyn skalarny tych wektorów. Nie musisz kontrolować, czy listy są faktycznie równe;

6. funkcję, której argumentem jest liczba, a wynikiem słowny zapis liczby. Możesz przyjąć, że wynik jest uproszczony, np. wynikiem $f(123)$ jest *jeden dwa trzy*;

7. funkcję, której argumentem jest lista słów (stringów), gdzie słowa są z ustalonego zbioru słów *{ zero, jeden, dwa, trzy, cztery, ... , dziewięć }*. Wynikiem ma być liczba, np. *f(["trzy", "pięć", "siedem"])* ma dać 357;

8. funkcję, której argumentem jest książka telefoniczna, tj. słownik którego argumentem jest imię kolegi lub koleżanki, a wartością numer telefonu. Zadaniem funkcji jest wypisać numery tylko kolegów lub tylko koleżanek (do wyboru). Regułą do rozróżniania płci może być fakt, że imiona żeńskie kończą się zwykle literą 'a'. Funkcja ta nie powinna zwracać żadnej wartości;

9. funkcję, której argumentami są: tabela kursów walut pamiętanych jako słownik *(symbol waluty, przelicznik)*, symbol waluty ze słownika oraz kwota (w złotówkach); wynikiem jest wartość podanej kwoty we wskazanej walucie;

10. funkcję, której argumentami są: cennik książek pamiętany jako słownik *(tytuł, cena)* oraz lista tytułów. Wynikiem jest cena zakupu książek z listy tytułów przy założeniu, że każdy tytuł jest kupowany w jednym egzemplarzu;

Najwygodniej jest umieścić wszystkie rozwiązania w jednym pliku, a na końcu umieścić kilka testów prezentujących możliwości zaimplementowanych funkcji. Za każde zadań można otrzymać 0,5 punkta, jednak za całą listę nie można otrzymać więcej niż 2 punkty (a więc do oceny można przedstawić co najwyżej 4 funkcje). Termin: zajęcia w przyszłym tygodniu.

Marcin Młotkowski

15 października 2008

Kurs języka Python

Lista 2.

Zadanie 1. Zaprojektuj strukturę danych pamiętającą rozkład jazdy pociągów: stacje początkowe, godziny odjazdu, stacje końcowe i godziny przyjazdu. Zakładamy tu, że pociągi kursują tylko między stacjami dwiema stacjami, bez stacji pośrednich.

Napisz funkcję, która dla zadanej stacji początkowej i końcowej zaproponuje jakąś trasę (niekoniecznie optymalną) lub napisze, że nie jest możliwe znalezienie żądanej trasy.

Zadanie 2. Pewien prosty system zapisów na zajęcia zapamiętuje dwa rodzaje danych: listę osób zapisanych na każde zajęcia (oraz godziny tych zajęć), oraz listę zajęć prowadzonych w danej sali. Jednak konieczne jest sprawdzenie, czy informacje są pełne i niesprzeczne. Napisz trzy funkcje:

1. sprawdzającą, czy przypadkiem ktoś nie zapisał się na zajęcia odbywające się w tym samym czasie;
2. sprawdzającą, czy wszystkie zajęcia są przypisane do jakiejś sali;
3. sprawdzającą, czy zajęcia przypisane salom się nie nakładają.

Oczywiście, wcześniej trzeba zaproponować struktury danych przechowujące te dane.

Zadanie 3. Wiele osób twierdzi, że swoje sukcesy osiągnęły dzięki planowaniu swoich zajęć. Zajęcia można podzielić na dwie podstawowe kategorie: zajęcia o ustalonych godzinach rozpoczęcia i zakończenia (tak jak wykład z Pythona ;), oraz zajęcia, na które trzeba przeznaczyć określoną ilość czasu.

Zaprojektuj strukturę danych przechowującą informacje o przewidywanych zajęciach. Napisz funkcję, która ułoży plan zajęć na najbliższy czas dbając, aby nie zaczynać zbyt wcześnie pracy i aby nie pracować dłużej niż do jakiejś godziny. Jeżeli będą wolne luki w planie, niech ta funkcja zaproponuje jakieś miłe i relaksujące zajęcia.

Zadanie 4. Pewien matematyk zapisał (jako listę w Pythonie) macierz w następujący sposób:

```
[ "12.34, 67.89, 56.21", "22.01, 11.0,    34.56", "20.0, 56, 0"]
```

Napisz funkcję obliczającą transpozycję tej macierzy i zwracającą macierz wynikową w takim formacie, jak macierz wejściowa.

Każde zadanie jest warte 2 punkty, na zajęciach oddaje się jedno zadanie.

Marcin Młotkowski

22 października 2008

Kurs języka Python

Lista 3.

Zaprogramuj dwa z poniższych zadań. Każde zadanie jest warte 1 pkt, można więc za tę listę uzyskać maksymalnie 2 pkt.

Zadanie 1. Napisz jednoargumentową funkcję `pierwsza(n)`, która zwraca listę liczb pierwszych nie większych niż n, na przykład

```
>>> pierwsza(20)
[2, 3, 5, 7, 11, 13, 17, 19]
```

Zadanie 2. Napisz jednoargumentową funkcję `doskonale(n)`, która zwraca listę liczb doskonałych nie większych niż n, na przykład

```
>>> doskonale(1000)
[6, 28, 496, 8128]
```

Zadanie 3. Napisz jednoargumentową funkcję `rozklad(n)` która oblicza rozkład liczby n na czynniki pierwsze i zwraca jako wynik listę par $[(p_1, w_1), (p_2, w_2), \ldots, (p_k, w_k)]$ taką, że
$n = p_1^{w_1} * p_2^{w_2} * \ldots * p_k^{w_k}$ oraz $p_1, \ldots, p_k$ są różnymi liczbami pierwszymi. Na przykład

```
>>> rozklad(756)
[(2, 2), (3, 3), (7, 1)]
```

Zadanie 4. Napisz jednoargumentową funkcję `zaprzyjaznione(n)`, która zwraca listę par liczb zaprzyjaźnionych nie większych niż n, na przykład

```
>>> zaprzyjaznione(1300)
[(220, 284), (1184, 1210)]
```

Odpowiednie definicje można znaleźć np. w polskiej Wikipedii. Implementacje poszczególnych funkcji powinne być w postaci jednej listy składanej, zawierającej być może inną listę składaną. W przypadku bardzo długich wyrażeń akceptowane będzie wydzielenie podlisty składanej

```
def zadana_funkcja(n):
    lista_tymcz = [ lista skladana ]
    return [ lista_składana_zawierająca lista_tymcz ]
```

Do zaprogramowania powyższych zadań wystarczą standardowe funkcje i operatory, nie ma potrzeby korzystania z dodatkowych modułów.

Marcin Młotkowski

29 października 2008

Kurs języka Python

Lista 4.

Zadanie 1. Zaprogramuj klasę ***Wyrazenie***, której obiekty reprezentują wyrażenia arytmetyczne (bądź logiczne, do wyboru). Przyjmij, że wyrażenia zawierają stałe, zmienne oraz podstawowe operatory. Przyjmij, że wyrażenia są pamiętane w postaci drzew, gdzie węzłami są obiekty różnych klas.

Zaprogramuj w klasie ***Wyrazenie*** metody: obliczania wyrażeń, nadawania i zmiany wartości zmiennych, oraz drukowania w ładny sposób drzewa wyrażeń. Zadbaj, aby było możliwe przeglądanie drzewa za pomocą instrukcji `for-in`.

Zadanie 2. Zadeklaruj klasę ***IntStream*** implementującą strumień liczb całkowitych, która implementuje metody:

```
int next();
bool eos();
void reset();
```

gdzie metoda `eos()` oznacza koniec strumienia, a `reset()` inicjuje na nowo strumień. Zadeklaruj dwie podklasy

- ***PrimeStream*** implementującą strumień liczb pierwszych, tj. wartościami kolejnych wywołań metody `next()` są kolejne liczby pierwsze. Oczywiście ze względu na ograniczony rozmiar typu `int` możliwe jest jedynie zwrócenie liczb pierwszych mniejszych niż rozmiar typu. Gdy nie jest możliwe obliczenie kolejnej liczby pierwszej, wartość `eos()` powinna być `true`.
- klasę ***RandomStream***, w której metoda `next()` zwraca liczby losowe. W takim wypadku `eos()` jest zawsze fałszywe.

Wykorzystaj te klasy do implementacji klasy ***RandomWordStream*** realizującej strumień losowych słów o długościach równych kolejnym liczbom pierwszym.

Zadanie 3. Zaprogramuj zbiór klas (przynajmniej trzy klasy) implementujących różne rodzaje okienek ekranowych, np. okna, okna z tytułem, okna z komunikatem itp. wraz z typowymi operacjami takimi jak narysowanie okna, przesunięcie itp. Przyjmij, że urządzenie graficzne jest reprezentowane przez klasę ***Ekran*** (zaprogramowaną przez Ciebie). Dla ułatwienia możesz przyjąć, że realizacja okien będzie w trybie tekstowym za pomocą znaków semigraficznych, a odpowiednie metody klasy ***Ekran*** wyświetlają stan ekranu na konsoli wiersz po wierszu. Możesz także przyjąć, że po każdej zmianie wszystkie okna są rysowane na nowo.

Zadanie 4. Zdefiniuj klasę ***Wektor*** implementujący wektory swobodne o współrzędnych operacjami dodawania wektorów (o ile mają tyle samo współrzędnych), iloczynu skalarnego czy mnożenia przez skalar. Zadbaj o to, aby można było się odwoływać do kolejnych współrzędnych wektora za pomocą instrukcji `for-in` bez ujawniania wewnętrznej implementacji.

Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 2 punkty. Proszę zadbać też o komentarze w swoich programach tak, aby (help) dawał ładnie sformatowny opis klasy.

Marcin Młotkowski

5 listopada 2008

Kurs języka Python

Lista 5.

Zadanie 1. Zaprogramuj interfejs graficzny (okno) który zawiera płótno (***Canvas***) po którym będzie chodził żółw (***RawPen***), oraz instrukcje sterujące żółwiem (jako opcje wybierane z menu; parametry typu długość kroku czy obrót powinne być podawane poprzez kontrolkę ***Entry***).

Zadanie 2. Zaprogramuj prostą przeglądarkę obrazków (w formatach akceptowanych przez instytutową instalację Pythona). Przyjmij, że program może albo wyświetlać wszystkie obrazki z podanego (w ***Entry***) katalogu, albo poprzez jawne wskazanie pliku graficznego (do tego może się przydać biblioteka *Tix*).

Zadanie 3. Zaprogramuj program rysujący wykresy kilku ustalonych funkcji. Przyjmij, że wskazanie funkcji następuje w menu, natomiast wartości *od do* przedziału rysowania są określane w kontrolkach ***Entry***.

Zadanie 4. Zaprogramuj następującą prostą grę: na rysunku jest armata, która ma regulowany kąt wystrzału i prędkość początkową pocisku, oraz cel (odległość między armatą i celem może być losowa). Kąt wystrzału oraz prędkość pocisku powinna być zadawana przez użytkownika, np. za pomocą kontrolek ***Entry***. Zadanie polega na takim wybraniu kąta i prędkości, aby pocisk trafił w cel. Korzystając z prostych praw fizyki narysuj tor pocisku oraz oblicz, czy pocisk trafił w cel.

Zadanie 5. Bardzo ładnymi figurami geometrycznymi są fraktale. Sporo materiałów o nich można znaleźć w internecie (są również książki o fraktalach w naszej bibliotece). Zadanie polega na zaprogramowaniu kilku fraktali. Fraktale zwykle mają parametry, które powinne być podawane np. poprzez kontrolki ***Entry***.

Zadania proszę wykonać za pomocą biblioteki Tkinter i pochodnych (Tix czy turtle). Najwygodniej jest zaprogramować aplikacje tak, że w jednym oknie znajduje się menu oraz kontrolka ***Canvas***, na której będą rysowane obrazki. Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 3 punkty. Proszę zadbać też o komentarze w swoich programach.

Marcin Młotkowski

12 listopada 2008

Kurs języka Python

Lista 6.

Zadanie 1. Wiele zestawień (np. notowania giełdowe) jest dostępnych w internecie w formacie pliku CSV. Napisz program, który na podstawie tych danych narysuje wykresy notowań, rysując poszczególne notowania w w różnych kolorach. Zadbaj, aby w łatwy sposób dało się zmienić źródło danych (internet czy plik dyskowy).

Zadanie 2. Napisz program obsługujący prostą bazę danych typu spis książek czy płyt CD. Przyjmij, że dane są wprowadzane i modyfikowane za pomocą interfejsu graficznego, oraz że są one zapamiętywane w pliku, np. jako rekordy, w pliku CSV czy też obiekty zachowywane są w plikach za pomocą modułu `pickle`.

Zadanie 3. Napisz program, który próbuje weryfikować autorstwo zadanego tekstu. Na początku za pomocą zadanego tekstów znanych autorów program "uczy się" zapamiętując statystyki (wymyśl sam, jakie to mogą być statystyki). Ta wiedza jest zapamiętywana w pliku. Następnie można podać tekst i sprawdzić, kto może być potencjalnym autorem. Wypróbuj swój program na autentycznych tekstach.

Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 4 punkty. Proszę zadbać też o komentarze w swoich programach.

Marcin Młotkowski

19 listopada 2008

Kurs języka Python

Lista 7.

Zadanie 1. Zaprogramuj własny organizator swojego czasu zawierający planowane spotkania (od-do), sprawy do załatwienia (do czasu), wraz z opcją przypominania. Dane niech będą przechowywane w bazie danych (typu dbm, pickle czy SQLite). Dodaj do tego interfejs graficzny.

Zadanie 2. Napisz program, który przechowuje w swojej lokalnej bazie danych informacje o posiadanych płytach z muzyką (identyfikator płyty, lista utworów i autorzy) wraz z informacjami o wypożyczeniu płyty znajomym. Uzupełnij swój program o interfejs dostępu do danych.

Zadanie 3. Zaprogramuj własny notatnik z kontaktami do znajomych zawierający ich numery telefonów, adresy email czy gg. Dane niech będą przechowywane w bazie danych (typu dbm, pickle czy SQLite). Dodaj do tego interfejs graficzny.

Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 4 punkty. Proszę zadbać też o komentarze w swoich programach.

Marcin Młotkowski

26 listopada 2008

Kurs języka Python

Lista 8.

Zadanie 1. Zaprogramuj moduł służący do przeglądania stron WWW i wykonujący dla każdej strony pewną akcję. Przyjmij, że przeglądanie rozpoczyna się od strony, której adres jest podany jako argument wywołania. Następnie przeszukiwane są te strony, do których linki znajdują się na bieżącej stronie. Zadbaj o to, aby przeszukiwanie się nie zapętlało. Przyjmij, że głębokość przeszukiwań jest ograniczona i zadawana każdorazowo przy inicjowaniu przeszukiwnania. Również akcja, która ma być wykonywana dla każdej strony winna być parametrem wywołania odpowiedniej funkcji modułu.

Zaprezentuj wykorzystanie modułu do wyszukiwania zdań zawierających słowo *Python*.

Zadanie 2. Napisz pakiet użytecznych narzędzi webowych, które pomogą pielęgnować i ulepszać prowadzony serwis

- moduł przeglądający strony WWW (pliki *.html) podanym katalogu i podkatalogach i sprawdzający, czy odnośniki do innych stron czy obrazków są aktywne.
- moduł przeglądający strony serwisu (również przeglądając pliki) i dla każdej strony (pliku) wypisujący, w których plikach są odnośniki do niej.

Zadanie 3. Napisz własną miniwyszukiwarkę internetową, tj. pakiet który

- przegląda strony i zapamiętuje liczbę wystąpień poszczególnych słów na poszczególnych stronach;
- przy wywołaniu programu z podanym słowem podaje strony na których to słowo występuje. Strony powinny być uszeregowane malejąco względem podanej liczby wystąpien. Możesz też zaproponować własną strategię rankowania stron.

Wystarczy, że indeksowane będą tylko wybrane strony, np. znajdujące się we wskazanych podkatalogach albo tylko strony do których da się dojść po odwołanaich w nie więcej niż k krokach od zadanej strony początkowej.

Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 4 punkty. Proszę zadbać też o komentarze w swoich programach.

Marcin Młotkowski

3 grudnia 2008

Kurs języka Python

Lista 9.

Zmodyfikuj zadanie z poprzedniej listy tak, aby poszczególne „podzadania” przeglądania stron były wykonywane w wątkach. Sprawdź, czy wykorzystywane w programie biblioteczne struktury danych są bezpieczne ze względu na wątki i odpowiednio zmodyfikuj operacje na nich, jeżeli nie nadają się do programów wielowątkowych.

Zadanie jest warte 3 pkt.

To już jest ostatnie zadanie. Na stronie wykładu znajdują się propozycje projektów oraz zasady ich oceniania.

Marcin Młotkowski