Wstęp do programowania w języku Python
Kolokwium
24 stycznia 2018

Za kolokwium można dostać 103 punkty (100 zapowiedzianych oraz 3 bonusowe). W rozwiązaniach istotna jest przede wszystkim czytelność kodu.

Zadanie 1.(15pkt) Napisz w Pythonie funkcję `kratka(N)`, która za pomocą instrukcji `print` wypisuje kwadratową kratkę o boku N.

```
########
# # # # #
########
# # # # #
########
# # # # #
########
# # # # #
########
```

Poprawny program wart jest 15 punktów.

Zadanie 2.(5pkt) Zmodyfikuj powyższy program w ten sposób, że funkcja będzie miała dwa argumenty (`kratka(N, napis)`), i w miejsce pustych kratek wpisywane będą kolejne znaki argumentu `napis`. Jeżeli napis będzie za długi, to zmieści się tylko pewna liczba początkowych znaków, jeżeli napis będzie za krótki, to powinieneś go uzupełnić spacjami.

Zadanie 3.(18pkt) Przeanalizuj poniższe polecenia i opisz, jaką wartość ma lista `L` po zakończeniu działania programu. Jeżeli uważasz, że któreś z poleceń spowoduje błąd, to napisz to i podaj wartość listy `L` przy założeniu, że tego polecenia nie ma.

```
def f(s):
   R = []
   napis = ''
   liczba = 0
   for i in range(len(s)):
     if s[i] not in '0123456789':
       napis += s[i]
     else:
       liczba = 10 * liczba + int(s[i])
       if napis:
         R.append(napis)
         napis = ''
   return R + [liczba]

L = 3 * [0,1]
L.append( (2>1) or (1/0 < 0))
L =  L + [L[2:4]]
L[0] = f('alama12kota34')
L += [ ['x'] ]
L += f('abc')
```

Zadanie 4.(20pkt) W tym zadaniu będziemy analizować plik tekstowy, składający się z wierszy o następującej postaci: `miasto1 miasto2 odległość-między-nimi` Przykładowo:

```
Wrocław Opole 74
Opole Katowice 111
Wrocław Gdańsk 356
```

Miasta są jednym wyrazem. Odległość nie musi być symetryczna (bo np. mogą być jednokierunkowe drogi).

a) Zakładamy, że informacje o odległościach będziemy przechowywać w słowniku. Napisz dokładnie, co w tym słowniku powinno być kluczem, a co wartością. **2**

b) Napisz fragment programu, który wczytuje ten słownik **3**

c) Napisz fragment programu, który wypisuje największą odległość między miastami, i wszystkie pary miast, które tę odległość realizują. **4**

d) Napisz funkcję `propozycje_wycieczek(D)`, która dla słownika `D` z punktu a) wypisuje dla każdego miasta wypisuje jedno z miast, które jest najbliżej. **5**

e) Zmodyfikuj wszystkie poprzednie programy tak, aby nazwy miast mogły składać się z więcej niż jednego wyrazu. **6** Poprawne wielowyrazowe (być może więcej niż dwuwyrazowe) miejscowości znajdują się w pliku `miejscowosci.txt`, możesz założyć, że jeżeli mamy wiersz:

```
Gąbki Wielkie Maciejowice 12
```

to w pliku z miejscowościami znajdują się albo *Gąbki Wielkie*, albo *Wielkie Maciejowice*, ale nie obie te miejscowości. Jeżeli jakiś kawałek kodu działa poprawnie dla wielowyrazowych miejscowości, to oczywiście nie powinieneś go przepisywać (wystarczy stwierdzenie, że kod w danym podpunkcie jest dobry również dla rozszerzonego zakresu działania).

Zadanie 5.(15pkt) W języku LISP wyrażenia zapisuje się w notacji prefiksowej, w której najpierw pojawia się znak działania, a po nim ciąg argumentów. Przykładowo `1+2+3*4` zapisalibyśmy jako `(+ 1 2 (* 3 4))`. Powinieneś napisać interpreter wyrażeń lispowych, wykonując dwa podpunkty:

a) Napisz funkcję, przekształcającą wyrażenie lispowe do odpowiedniego wyrażenia Pythonowego (w powyższym przykładzie do `["+", 1, 2, ["*", 3, 4]]` (Możesz założyć, że spacje stawiane są tak jak w powyższym przykładzie.)

b) Napisz rekurencyjną funkcję, która wylicza wartość wyrażenia Pythonowego.

Zadanie 6.(30pkt) W tym zadaniu powinieneś napisać definicje kilku opisanych niżej funkcji. Każdą z nich da się napisać w następującej *wyrażeniowej* postaci:

```
def f(...):
   return wyrażenie-w-pythonie
```

i taka postać jest w tym zadaniu preferowana, tzn. poprawne napisanie definicji zgodnej z tą postacią warte jest 4.5 lub 5p (4.5 w przypadku, gdy rozwiązanie będzie dużo bardziej skomplikowane od rozwiązania modelowego). Poprawna definicja korzystająca z innych konstrukcji (pętle, instrukcje podstawienia, ...) jest warta 4p. Może przydać się wiedza, że `'ŻÓŁta FEBrA'.lower() == 'żółta febra'`

a) `suma_dwucyfrowych_parzystych(L)`, która dla listy liczb nieujemnych L zwraca sumę dwucyfrowych liczb parzystych zawartych w L

b) `maksymalny_jednorodny(s)`, która dla napisu zwraca najdłuższy ciąg postaci `dd...d`, gdzie d jest znakiem odpowiadającym którejś cyfrze. Jeżeli ciągów najdłuższych jest więcej, funkcja powinna zwracać ciąg z największą cyfrą.

c) `lista_w_liscie(A,B)`, która zwraca wartość logiczną, mówiącą czy lista liczb A jest spójną podlistą listy B (czyli B == X + A + Y, dla pewnych list X i Y)

d) `liczba_roznych_slow(L)` która zwraca liczbę różnych słów na liście L, przy czym uznajemy za jednakowe (czyli nie różne) słowa, które różnią się jedynie wielkością liter.

e) `pary_dzielnikow(A,B)`, która dla list liczb zwraca listę takich par `(a,b)`, że `a` dzieli `b`, `a` należy do listy `A`, `b` należy do listy `B` i obie te liczby znajdują się na swoich listach na tej samej pozycji.

f) `prawie_rosnacy(L)`, która dla listy liczb zwraca True, wtedy i tylko wtedy, gdy listę da się podzielić na dwie spójne części (L == A + B), takie że zarówno A, jak i B są niemalejące.