```
import string, sys


def otevriSoubor(nazevSouboru):

    #r jako read
    soubor = open(nazevSouboru, "r")

    return soubor


def nactiSoubor(soubor):

    #tohle volani nacte cely soubor do pameti
    slova = string.split(soubor.read());

    soubor.close()

    return slova


def spoctiSlova(nazevSouboru, slova):

    #slozene zavorky inicializuji hash mapu/slovnik, zatim prazdna
    vyskyty = {}

    #cyklus for bez zavorek!!
    for slovo in slova:

        upravene = string.lower(string.strip(slovo, "`',.:"))

        # +1 nam inkrementuje
        vyskyty[upravene] = vyskyty.get(upravene, 0) + 1


    print("Soubor {0} obsahuje celkem {1} slov (z toho {2}

            jedinecnych).\n".format(nazevSouboru, len(slova), len(vyskyty)))

    return vyskyty


def sortfce(x,y):

    return cmp(y[1], x[1])


def vypisSlova(vyskyty):

    polozky = vyskyty.items()

    polozky.sort(sortfce)

    for slovo in polozky:

        print("\t{0:15s} - {1:3d}".format(slovo[0], slovo[1]))
```

```
def main():

    #nazevSouboru = raw_input("Zadejme jmeno souboru: ")

    nazevSouboru = sys.argv[1]

    nazevSouboru = string.strip(nazevSouboru)

    soubor = otevriSoubor(nazevSouboru)

    slova = nactiSoubor(soubor)

    vyskyty = spoctiSlova(nazevSouboru,slova)

    vypisSlova(vyskyty)

main()
```

```
# -*- coding: cp1250 -*-

"""
SAX zpracovani souboru bar.xml
Ukol - zjistit, ktery koktejl vynasi nejvic
"""

# potrebne pro tvorbu HTML
from __future__ import with_statement

# potrebne pro praci se vstupem a vystupem, s retezci
import sys, string

# potrebne pro SAX
from xml.sax import handler, make_parser

# potrebne pro validaci
from lxml import etree

# domenova trida - slozka

class Slozka:

    def __init__(self):

        self.nazev = ''

        self.cena = ''


    def setId(self, id):

        self.id = id


    def getId(self):

        return self.id
```

```
    def setNazev(self, nazev):
        self.nazev = nazev

    def getNazev(self):
        return self.nazev;

    def setCena(self, cena):
        self.cena = cena

    def getCena(self):
        return self.cena
```

```
# domenova trida - koktejl
class Koktejl:

    def __init__(self):
        self.nazev = ''
        self.cena = 0
        self.vydaje = 0

    def setId(self, id):
        self.id = id

    def getId(self):
        return self.id

    def setNazev(self, nazev):
        self.nazev = nazev

    def getNazev(self):
        return self.nazev;

    def setCena(self, cena):
        self.cena = cena
```

```
    def getCena(self):
        return self.cena

    def addVydaj(self, slozky, id, mnozstvi):
        for slozka in slozky:
            if (slozka.getId() == id):
                self.vydaje += mnozstvi * slozka.cena
                break

    def getVydaje(self):
        return self.vydaje
```

```
# Definice nove tridy oddedene od ContentHandler
class BarHandler(handler.ContentHandler):

    # konstruktor tridy
    def __init__(self):

        # inicializace dat tridy
        self.inSlozky = False    # indikator stavu
        self.inKoktejly = False # indikatot stavu
        self.inNazev = False # indikatot stavu
        self.inCena = False # indikatot stavu
        self.inSlozka = False # indikatot stavu
        self.koktejlyList = []  # seznam koktejlu
        self.slozkyList = []    # seznam slozek

    # getter pro seznam koktejlu
    def getKoktejlyList(self):
        return self.koktejlyList

    # prekryti metodu startElement(self)
    def startElement(self, name, attrs):
        # ulozeni stavu, kde se prave ve XML nachazime
        if (name == "slozky") and (not self.inKoktejly):
```

```
            self.inSlozky = True
        elif (name == "koktejly"):
            self.inKoktejly = True
        elif (name == "nazev"):
            self.inNazev = True
        elif (name == "cena"):
            self.inCena = True
        elif (name == "slozka"):
            self.inSlozka = True
            if self.inSlozky:
                #vytvoreni instance slozky (jsme uvnitr slozky, ne koktejlu)
                self.slozka = Slozka()
                self.slozka.setId(string.atoi(attrs.get("id")))
            else:
                self.id = string.atoi(attrs.get("id"))
        elif (name == "koktejl"):
            # vytvoreni instance koktejlu
            self.koktejl = Koktejl()
            self.koktejl.setId(string.atoi(attrs.get("id")))

    # prekryti metody endElement(self, name)
    def endElement(self, name):
        # ulozeni stavu, kde se prave ve XML nachazime
        if (name == "slozky") and (not self.inKoktejly):
            self.inSlozky = False
        elif (name == "koktejly"):
            self.inKoktejly = False
        elif (name == "nazev"):
            self.inNazev = False
        elif (name == "cena"):
            self.inCena = False
        elif (name == "slozka"):
            self.inSlozka = False
            if self.inSlozky:
```

```
                self.slozkyList.append(self.slozka)

            else:

                # pridani vydaje k aktualnimu koktejlu
                self.koktejl.addVydaj(self.slozkyList, self.id, self.mnozstvi)

        elif (name == "koktejl"):

            # pridani koktejlu do seznamu
            self.koktejlyList.append(self.koktejl)


    # prekryti metody characters(self, chrs)
    def characters(self, chrs):

        if self.inCena:

            if self.inSlozky:

                self.slozka.setCena(string.atof(chrs))

            else:

                self.koktejl.setCena(string.atof(chrs))

        elif self.inNazev:

            if self.inSlozky:

                self.slozka.setNazev(chrs.encode('windows-1250'))

            else:

                self.koktejl.setNazev(chrs.encode('windows-1250'))

        elif self.inSlozka and self.inKoktejly:

            self.mnozstvi = string.atof(chrs)


# Pomocna funkce pro razeni seznamu koktejlu (dle vydaju)
def sortfunc(x,y):

    if (x.getCena()-x.getVydaje()) < (y.getCena()-y.getVydaje()):
        return 1
    else:
        return -1


# unkce pro tisk seznamu
def printList(koktejlyList):

    print "id\tnazev\tcena\tvydaj\tzisk\t"

    for koktejl in koktejlyList:

        print "%s\t%s\t%s\t%s\t" % (koktejl.getId(), koktejl.getNazev(),

 koktejl.getCena(), koktejl.getVydaje()), koktejl.getCena()-koktejl.getVydaje()


    # tisk do souboru
```

```
    file_html = file("zisk.html", "w")
    #file_html.write(node.pretty().conv().bytes(encoding="windows-1250"))


# Funkce testujici validitu XML dokumentu proti XSD
def validate(xmlName, xsdName):

    #parsovani XML a vytvoreni stromove struktury
    doc = etree.parse(xmlName)

    #parsovani XSD a vytvoreni stromove struktury
    xmlschema_doc = etree.parse(xsdName)

    #vytvoreni schematu
    xmlschema = etree.XMLSchema(xmlschema_doc)

    #validace, vysledek je bud True nebo False
    return xmlschema.validate(doc)


# Hlavni funkce
def main():

    # validace XML proti XSD
    if validate('bar.xml','bar.xsd'):
          print "XML soubor validuje proti XSD"
    else:
          print "XML soubor nevaliduje proti XSD"

          quit()

    # vytvoreni instance handleru
    handler = BarHandler()

    # vytvoreni instance parseru
    parser = make_parser()

    # nastaveni handleru
    parser.setContentHandler(handler)

    # otevreni souboru pro cteni
    inFile = open('bar.xml', 'r')

    # parsovani souboru
    parser.parse(inFile)

    # ziskani seznamu koktejlu
    koktejly = handler.getKoktejlyList()

    # razeni seznamu koktejlu podle funkce sortfunc
    koktejly.sort(sortfunc)

    # tisk koktejlu do konzole
    printList(koktejly)

    # vystup do HTML
    #toHTML(koktejly)

    #zatvreni souboru
    inFile.close()

main()
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<bar>
  <slozky>
    <slozka id="1">
      <nazev>Tequilla</nazev>
      <jednotka>cl</jednotka>
      <cena>4</cena>
    </slozka>
    <slozka id="2">
      <nazev>Pomerančová šťáva</nazev>
      <jednotka>cl</jednotka>
      <cena>0.15</cena>
    </slozka>
    <slozka id="3">
      <nazev>Kostka ledu</nazev>
      <jednotka>ks</jednotka>
      <cena>0.1</cena>
    </slozka>
 <koktejly>
    <koktejl id="1">
      <nazev>Tequilla Sunrise</nazev>
      <cena>60</cena>
      <slozky>
        <slozka id="1">4</slozka>
        <slozka id="2">25</slozka>
        <slozka id="3">4</slozka>
        <slozka id="4">2</slozka>
        <slozka id="5">1</slozka>
      </slozky>
    </koktejl>
    <koktejl id="2">
      <nazev>Koktej ze šampaňského</nazev>
      <cena>65</cena>
      <slozky>
        <slozka id="6">14</slozka>
        <slozka id="7">3</slozka>
        <slozka id="8">1</slozka>
        <slozka id="9">1</slozka>
        <slozka id="5">1</slozka>
      </slozky>
    </koktejl>
    <koktejl id="3">
      <nazev>Tom Collins</nazev>
      <cena>55</cena>
      <slozky>
        <slozka id="10">8</slozka>
        <slozka id="11">1</slozka>
        <slozka id="3">4</slozka>
        <slozka id="12">1</slozka>
        <slozka id="13">30</slozka>
      </slozky>
    </koktejl>
      <slozky>
        <slozka id="14">4</slozka>
        <slozka id="3">4</slozka>
      </slozky>
  </koktejly>
</bar>
```

```
import pyodbc

class Databaze:

    def __init__(self, dburl):

        self.dburl = "DSN=" + dburl

    """ metoda pro pripojeni se k databazi """


    def pripoj(self):

        # pri pripojeni muze dojit k vyjimecne situaci
        try:
            self.db = pyodbc.connect(self.dburl)

        except:
            # neexistence datoveho zdroje
            print "Datovy zdroj neexistuje"

            quit()    # ukonceni programu

    """ metoda pro vypis obsahu tabulky cyklem for """


    def vypis_for(self):

        # pripojeni se k databazi - ulozeno v atributu db
        self.pripoj()

        print "Vypis obsahu tabulky osoby:\n"

        # vytvoreni objektu, kterym bude realizovan SQL prikaz v DB
        cursor = self.db.cursor()

        # sestaveni SQL prikazu - vypis obsahu tabulky osoby
        self.stmt = "SELECT prijmeni, jmeno " \

                    "FROM osoby " \

                   "ORDER BY prijmeni, jmeno"


        # vykonani prikazu a ulozeni celeho obsahu tabulky do promenne rows
        rows = cursor.execute(self.stmt).fetchall()

        # vypisovani jednotlivych zaznamu na obrazovku
        for row in rows:

            print "%s %s" % (row.jmeno, row.prijmeni)

        cursor.close()     # uzavreni objektu typu kurzor - slusnost

        self.db.close()    # odpojeni se od DB

        print "\n"
```

```
""" metoda pro vypis obsahu tabulky cyklem while """
def vypis_while(self):

    # pripojeni se k databazi - ulozeno v atributu db
    self.pripoj()

    print "Vypis obsahu tabulky osoby:\n"

    # vytvoreni objektu, kterym bude realizovan SQL prikaz v DB
    cursor = self.db.cursor()

    # sestaveni SQL prikazu - vypis obsahu tabulky osoby
    self.stmt = "SELECT prijmeni, jmeno " \

                 "FROM osoby " \

                "ORDER BY prijmeni, jmeno"


    # vykonani prikazu a ziskani prvniho zaznamu
    row = cursor.execute(self.stmt).fetchone()

    while row:

        # vypsani pozadovanych hodnot atributu akt. zaznamu
        print "%s %s" % (row.jmeno, row.prijmeni)

        # nacteni dalsiho zaznamu
        row = cursor.fetchone()

    cursor.close()    # uzavreni objektu typu kurzor - slusnost

    self.db.close()   # odpojeni se od DB

    print "\n"


""" metoda pro vlozeni noveho zaznamu do tabulky """

def vloz(self):

    # pripojeni se k databazi - ulozeno v atributu db
    self.pripoj()

    print "Vlozeni Maxe Otty\n"

    # vytvoreni objektu, kterym bude realizovan SQL prikaz v DB
    cursor = self.db.cursor()

    # priprava dat, ktere vlozime do DB
    jmeno = "Max"

    prijmeni = "Otta"

    # sestaveni SQL prikazu s parametry - vlozeni noveho zaznamu do tabulky
    self.stmt = "INSERT INTO osoby (jmeno, prijmeni) VALUES (?, ?)"

    # vykonani prikazu, predani hodnot formalnich parametru
    cursor.execute(self.stmt, (jmeno, prijmeni))
```

```
        self.db.commit()  # potvrzeni transakce

        cursor.close()    # uzavreni objektu typu kurzor - slusnost
        self.db.close()   # odpojeni se od DB


    """ metoda pro zmenu zaznamu v tabulce """
    def zmen(self):

        # pripojeni se k databazi - ulozeno v atributu db
        self.db = pyodbc.connect(self.dburl)

        print "Prejmenovani Maxe na Maxmilliana\n"

        # vytvoreni objektu, kterym bude realizovan SQL prikaz v DB
        cursor = self.db.cursor()

        # priprava dat, podle kterych budeme menit data v DB
        jmeno = "Maxmillian"

        prijmeni = "Otta"


        # sestaveni SQL prikazu s parametry - zmena zaznamu v tabulce
        self.stmt = "UPDATE osoby SET jmeno = ? WHERE prijmeni = ?"


        # vykonani prikazu, predani hodnot formalnich parametru
        cursor.execute(self.stmt, (jmeno, prijmeni))

        self.db.commit()  # potvrzeni transakce

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()   # odpojeni se od DB


    """ metoda pro mazani zaznamu v tabulce """
    def smaz(self):

        # pripojeni se k databazi - ulozeno v atributu db
        self.db = pyodbc.connect(self.dburl)

        print "Zruseni Maxmilliana Otty\n"

        # vytvoreni objektu, kterym bude realizovan SQL prikaz v DB
        cursor = self.db.cursor()

        # priprava dat, podle kterych budeme menit data v DB
        prijmeni = "Otta"

        # sestaveni SQL prikazu s parametry - mazani zaznamu v tabulce
        self.stmt = "DELETE FROM osoby WHERE prijmeni = ?"

        # vykonani prikazu, predani hodnot formalnich parametru
        cursor.execute(self.stmt, (prijmeni))

        self.db.commit()  # potvrzeni transakce

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()   # odpojeni se od DB
```

```
""" hlavni program """
def main():

    mydb = Databaze("osoby")  # vytvoreni instance tridy Databaze

    mydb.vypis_while()        # vypis obsahu tabulky osoby

    mydb.vloz()               # vlozeni noveho zaznamu

    mydb.vypis_for()          # vypis obsahu tabulky osoby

    mydb.zmen()               # zmena nekterych zaznamu v tabulce

    mydb.vypis_for()          # vypis obsahu tabulky osoby

    mydb.smaz()               # mazani nekterych zaznamu v tabulce

    mydb.vypis_for()          # vypis obsahu tabulky osoby


main()  # spusteni hlavniho programu
```

```
# potrebne pro SAX
from xml.sax import handler, make_parser


# potrebne pro validaci
from lxml import etree


# potrebne na razeni ...
from operator import attrgetter
```

```
import string


# trida na jmena
class Jmeno:

    def __init__(self):

        self.jmeno = ''

    def setJmeno(self, jmeno):

        self.jmeno = jmeno

    def getJmeno(self):

        return self.jmeno
```

```
class JmenoHandler(handler.ContentHandler):

    def __init__(self):

        self.inJmeno = False

        self.jmenaList = [] #seznam jmen


    def getJmenaList(self):

        return self.jmenaList


# prekryti metodu startElement(self)
    def startElement(self, name, attrs):

        # ulozeni stavu, kde se prave ve XML nachazime
        if (name == "jmeno"):

            self.inJmeno = True

            self.jmeno = Jmeno()


    # prekryti metody endElement(self, name)
    def endElement(self, name):

        if (name == "jmeno"):

            self.jmenaList.append(self.jmeno)

            self.inJmeno = False


    # prekryti metody characters(self, chrs)
    def characters(self, chrs):

        if self.inJmeno:

            self.jmeno.setJmeno(chrs)

            #self.jmenaList.append(chrs)


def printList(jmenaList):

    for jmeno in jmenaList:

        print(jmeno.getJmeno())


def delkySlov(jmena):

    delky = {} #slovnik

    for jmeno in jmena:

        upravene = len(jmeno.getJmeno())
```

```
        delky[upravene] = delky.get(upravene, 0) + 1

    return delky

def vyskytyPismen(jmenaList):

    vyskyty = {}

    for jmeno in jmenaList:

        upravene = string.lower(jmeno.getJmeno())

        pismeno1 = upravene[0:]

        #vyskyty[pismeno1] = vyskyty.get(pismeno1, 0) + 1
        vyskyty[pismeno1] = pismeno1

    return vyskyty

def sortfce(x, y):

    return cmp(y[0], x[0])
    #return cmp(x[1], x[1])

def printPismena(delky):

    polozky = delky.items()

    polozky.sort(sortfce)

    polozky.reverse()

    for delka in polozky:

        print("{0:1s}".format(delka[0], delka[1]))

def printSlovnik(delky):

    polozky = delky.items()

    polozky.sort(sortfce)

    polozky.reverse()

    for delka in polozky:

        print("{0:5d} - {1:1d}".format(delka[0], delka[1]))

def main():

    handler = JmenoHandler()

    # vytvoreni instance parseru
    parser = make_parser()

    # nastaveni handleru
    parser.setContentHandler(handler)
```

```
    # otevreni souboru pro cteni
    inFile = open('jmena.xml', 'r')

    # parsovani souboru
    parser.parse(inFile)

    jmena = handler.getJmenaList()

    #printList(jmena)
    inFile.close()
    #delky = delkySlov(jmena)

    #printSlovnik(delky)
    delky = vyskytyPismen(jmena)

    printPismena(delky)

main()
```

```
import pyodbc

class Databaze:

    """ konstruktor tridy Databaze """
    def __init__(self, dburl):

        self.dburl = "DSN=" + dburl


    """ metoda pro pripojeni se k databazi """
    def pripoj(self):

        # pri pripojeni muze dojit k vyjimecne situaci
        try:

            self.db = pyodbc.connect(self.dburl)
        except:

            # neexistence datoveho zdroje
            print ("Datovy zdroj neexistuje.")

            quit()    # ukonceni programu


    """ metoda pro vypis obsahu tabulky cyklem for """
    def vypis_for(self):

        # pripojeni se k databazi - ulozeno v atributu db
        self.pripoj()

        print ("Vypis obsahu tabulky osoby:")

        # vytvoreni objektu, kterym bude realizovan SQL prikaz v DB
        cursor = self.db.cursor()

        # sestaveni SQL prikazu - vypis obsahu tabulky osoby
```

```
        self.stmt = "SELECT jmeno " \

                    "FROM jmena " \

                   "ORDER BY jmeno DESC" # pro serazeni vzestupne staci pridat
ASC, sestupne DESC a neni potreba stupidniho serazovani pomoci funkce


        # vykonani prikazu a ulozeni celeho obsahu tabulky do promenne rows
        rows = cursor.execute(self.stmt).fetchall()


        # vypisovani jednotlivych zaznamu na obrazovku
        for row in rows:

            print ("{0}".format(row.jmeno))

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()   # odpojeni se od DB

        print ("\n")


    """ metoda pro vlozeni noveho zaznamu do tabulky """
    def vloz(self):

        self.pripoj()

        cursor = self.db.cursor()

        noveJmeno = "Radek"

        self.stmt = "INSERT INTO osoby (jmeno) VALUES (?)"

        cursor.execute(self.stmt, noveJmeno)

        self.db.commit()  # potvrzeni transakce, zmeny by jinak byly zahozeny!

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()   # odpojeni se od DB


    def zmen(self):

        self.pripoj()

        cursor = self.db.cursor()

        jmeno = "Josef"

        noveJmeno = "Pepa"

        self.stmt = "UPDATE osoby SET jmeno = ? WHERE jmeno = ?"

        # vykonani prikazu, predani hodnot formalnich parametru
        cursor.execute(self.stmt, (noveJmeno, jmeno))

        self.db.commit()  # potvrzeni transakce

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()
```

```
    """ Dulezita funkce pro dalsi manipulaci pro vyskyty !! Zapise obsah
      databaze z urciteho sloupecku do seznamu """
    def zapisDoSeznamuDB(self):

        self.seznam = []

        self.pripoj()

        cursor = self.db.cursor()

        self.stmt = "SELECT jmeno " \

                    "FROM osoby " \

                   "ORDER BY jmeno DESC"

        rows = cursor.execute(self.stmt).fetchall()

        for row in rows:

            self.seznam.append(row.jmeno)

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()

        return self.seznam


    def smaz(self):

        self.pripoj()

        cursor = self.db.cursor()

        jmeno = "Jan"

        self.stmt = "DELETE FROM osoby WHERE jmeno = ?"

        # vykonani prikazu, predani hodnot formalnich parametru
        cursor.execute(self.stmt, (jmeno))

        self.db.commit()  # potvrzeni transakce

        cursor.close()    # uzavreni objektu typu kurzor - slusnost

        self.db.close()


def delkySlov(seznamSlov):

    delky = {} #slovnik

    for slovo in seznamSlov:

        delkaSlova = len(slovo) #delka slova

        delky[delkaSlova] = delky.get(delkaSlova, 0) + 1

    return delky
```

```
""" Nejaka serazovaci funkce """
def sortfce(x, y):

    return cmp(y[1], x[1])


""" Funkce vypise polozky slovniku podle cetnosti vyskytu delek slov od
nejmensiho po nejvetsi"""
def printSlovnik(slovnik):

    polozky = slovnik.items()

    polozky.sort(sortfce)

    polozky.reverse()

    for polozka in polozky:


        print("{0:5d} - {1:1d}".format(polozka[0], polozka[1]))


""" Fuknce vytiskne obycejny seznam, pozn.: modifikovatelne """
def printSeznam(seznam):

    for polozka in seznam:

        print(polozka)


def main():


    mydb = Databaze("osoby2")  # vytvoreni instance tridy Databaze
    #mydb.vypis_for()          # vypis obsahu tabulky osoby


    seznam = mydb.zapisDoSeznamuDB()
    #printSeznam(seznam)

    mydb.vloz()

    mydb.zmen()

    mydb.smaz()

    delky = delkySlov(seznam)

    printSlovnik(delky)

main()
```