ZAJĘCIA NR 2

Zawartość

ZAJĘCIA 2 (3 i 4h) 2

--- Pytania do poprzednich zajęć, powtórzenie treści 2

--- WCZYTYWANIE: raw_input() 2

--- IF 4

➔ Składnia if-else 4

➔ Nierówność (!= lub <>) 6

➔ Przypisywanie wyniku porównania 6

➔ Składnia if-elif-else 7

➔ Rozwiązania niektórych zadań 9

--- WHILE 10

--- Lista (LIST) 12

➔ Przypisywanie, odwoływanie się 12

➔ Dodawanie i usuwanie elementów 14

➔ Rozwiązania niektórych zadań 16

--- * SŁOWNIK (DICT) 17

--- Pętla FOR 18

➔ for 18

➔ range 18

➔ Rozwiązania niektórych zadań 21

Autor: ŁUKASZ CZERWIŃSKI

L.Czerwinski@students.mimuw.edu.pl
CzerwinskiLukasz1@gmail.com

2012-10-17

ZAJĘCIA 2 (3 i 4h)

--- PYTANIA DO POPRZEDNICH ZAJĘĆ, POWTÓRZENIE TREŚCI

--- WCZYTYWANIE: RAW_INPUT()

```
# -*- coding: utf-8 -*-
imie = raw_input('Podaj imię: ')
print 'Cześć, ', imie, "!"
```

```
# coding: utf-8
a = raw_input('1. Podaj liczbę: ')
print 'a =', a
print

a = raw_input('2. Podaj liczbę: ')
print 'a + 2 =', int(a)+2
print

a = raw_input('3. Podaj liczbę: ')
b = int(a)+2
print 'a + 2 =', b
print

a = raw_input('4. Podaj liczbę: ')
a = int(a)+2
print 'a + 2 =', a
print

a = raw_input('5. Podaj liczbę: ')
a = int(a)+2.5
print 'a + 2.5 =', a
print
```

* Zadanie 1.

Wczytaj dwie liczby (np. do zmiennych `a` i `b`) i wypisz ich sumę.

* Zadanie 2.

Wczytaj liczbę `a` i wypisz liczbę do niej przeciwną.

* Zadanie 3.

Wczytaj liczbę `a` i wypisz jej odwrotność. Jakiego typu danych musisz użyć?

* Zadanie 4.

Wczytaj liczbę `r` i wypisz pole i obwód koła o takim promieniu.

 * Zadanie 5.

Wczytaj długości boków prostokąta i wypisz jego obwód.

 ** Zadanie 6.

Wczytaj długość przekątnej kwadratu i wypisz jego pole.

 ** Zadanie 7.

Wczytaj dwie liczby (np. do zmiennych `a` i `b`) i wypisz ich iloraz. Czy Twój program zawsze zadziała?

 ** Zadanie 8.

Dzielenie z resztą wygląda następująco: *dzielna* / *dzielnik* = *wynik* r. *reszta*. Napisz program, który wczyta trzy liczby: wynik ilorazu, dzielnik i resztę z dzielenia, a następnie oblicz dzielną.

 ** Zadanie 9.

Wczytaj dwie liczby (np. do zmiennych `a` i `b`) i wypisz $\sqrt[b]{a}$. Czy Twój program zawsze zadziała? Jeśli nie, to kiedy nie zadziała? Dlaczego?

 ** Zadanie 10.

Korzystając ze wzoru Herona, oblicz pole trójkąta. Poproś użytkownika o podanie niezbędnych danych. Sprawdź, czy podane liczby mogą być bokami trójkąta. Wzór Herona znajdź w Internecie. W razie problemów, poproś o pomoc prowadzącego zajęcia.

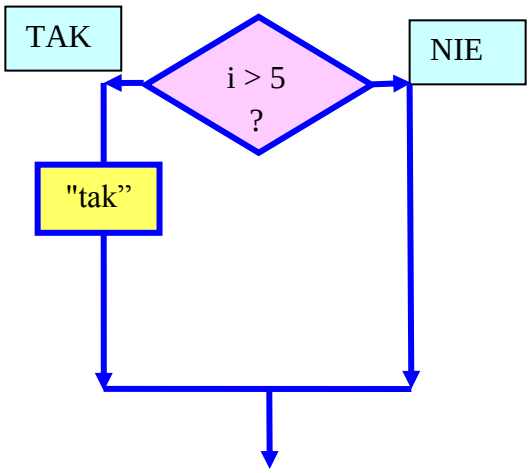
 *** Zadanie 10.

Zastanów się, jak sprawdzać, czy użytkownik rzeczywiście wpisał liczbę, a nie np. same litery. Zadanie wymaga elementów, o których nie było mowy. Zastanów się, czego Ci brakuje i znajdź rozwiązanie, korzystając z Internetu lub wskazówek prowadzącego.

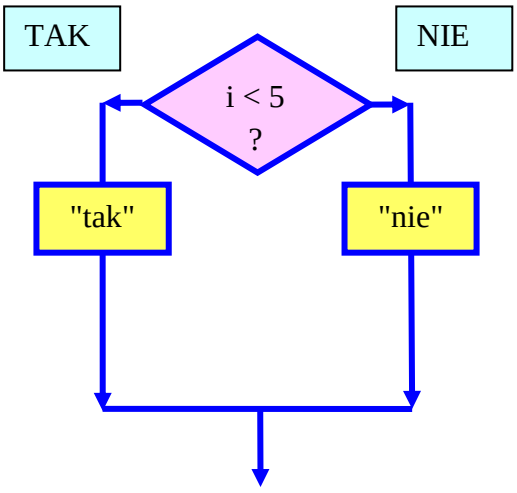
--- IF

➔ *Składnia if-else*

```
# coding: utf-8
i = int(raw_input("Podaj liczbę: "))
if i > 5:
  print "tak"
```

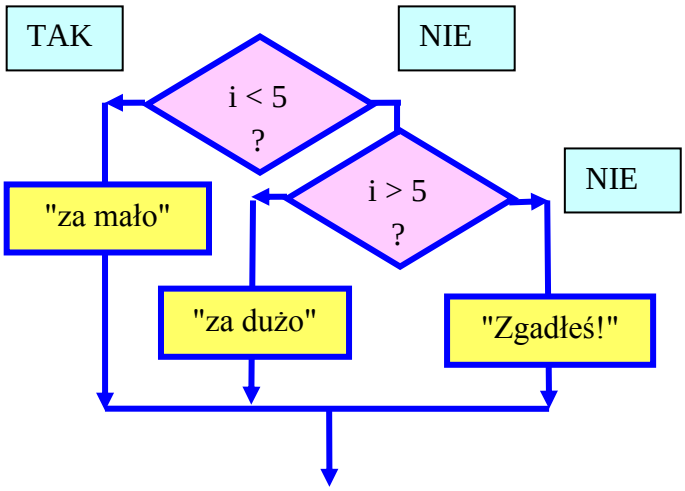



```
# coding: utf-8
i = int(raw_input("Podaj liczbę: "))
if i == 5:
  print "tak"
else:
  print "nie"
```



Konstrukcję if możemy też zagnieżdżać (umieszczać jedną w drugiej):

```
# coding: utf-8
i = int(raw_input('Wpisz liczbę: '))
if i < 5:
   print "za mało"
else:
   if i > 5:
      print "za dużo"
   else:
      print "Zgadłeś! i == 5"
```



* Zadanie 1. Czy zero.

Dla liczby n podanej na wejściu wypisz: „zero” jeśli n równa się zero, w przeciwnym wypadku nic nie wypisuj.

* Zadanie 2. Dodatnia/ujemna.

Dla liczby n podanej na wejściu wypisz: „dodatnia” lub „nieujemna”.

** Zadanie 2b. Dodatnia/ujemna/zero.

Dla liczby n podanej na wejściu wypisz: „dodatnia”, „ujemna” lub „zero”.

* Zadanie 3. Parzysta/nieparzysta.

Dla liczby n podanej na wejściu wypisz: „parzysta” lub „nieparzysta”.

* Zadanie 4. Która większa.

Napisz program, który wczyta dwie liczby i powie, która z nich jest większa.

** Zadanie 5.

Napisz program, który wczyta trzy liczby i powie, która jest największa, która średnia, a która najmniejsza.

** Zadanie 6. Deklinacja.

Napisz program, który dla liczby n z wejścia wypisze: „Mam <n> *literx*”, gdzie <n> oznacza wczytaną liczbę, a *literxx* oznacza słowo „litera” odpowiednio odmienione przez przypadki.

*** Zadanie 7. Porównania.

Zastanów się i odpowiedz bez sprawdzania, jakie wyniki dadzą następujące porównania:
1 == 1.0
"1" == 1
"1.0" == 1.0
Sprawdź, czy intuicja dobrze Ci podpowiedziała. Dlaczego wyniki są takie, a nie inne? Zastanów się, a następnie wyjaśnij to z prowadzącym zajęcia.

* Zadanie 7b. Popraw porównania.

Jak sprawić, by wyniki z poprzedniego zadania były takie, jakich oczekiwaliśmy?

➔ *Nierówność (!= lub <>)*

```
if i != 3:
  print "i różne od 3"
else:
  print "i równa się 3"
```

```
if i <> 3:
  print "i różne od 3"
else:
  print "i równa się 3"
```

➔ *Przypisywanie wyniku porównania*

Wynik porównania można wypisać lub przechować:

```
# coding: utf-8
i = int(raw_input("Podaj liczbę: "))
wynik = i <> 3:
print "Wynik porównania i<>3:", wynik

print "Wynik porównania i==5:", i==5

print "Wynik porównania i>7: ", (i>7)
```

➔ *Składnia if-elif-else*

Jeśli mamy więcej niż jeden warunek do spełnienia, możemy je zagnieździć, tzn. napisać tak:

```
# coding: utf-8
i = int(raw_input('Wpisz liczbę: '))
if i < -10:
   print "dużo za mało"
else:
   if i < 5:
      print "za mało"
   else:
      if i > 50:
         print "dużo za dużo"
      else:
         if i > 5:
            print "za dużo"
         else:
            print "Zgadłeś! i == 5"
```

Ale przy dużej liczbie `if`ów i `else`'ów robi się skomplikowana konstrukcja schodkowa. Można ją łatwo uprościć, zastępując „`else: if` ……:" przez skróconą konstrukcję: „`elif:`" (łatwo skojarzyć – jest to skrót od else if):

```
# coding: utf-8
i = int(raw_input('Wpisz liczbę: '))
if i < -10:
   print "dużo za mało"
elif i < 5:
   print "za mało"
elif i > 50:
   print "dużo za dużo"
elif i > 5:
   print "za dużo"
else:
   print "Zgadłeś! i == 5"
```

Dzięki temu unikamy zbyt dużego zagnieżdżenia poziomów.

 * Zadanie 1.

Napisz program, który poprosi użytkownika o wpisanie imienia, po czym dla kilku wybranych imion wypisze przywitanie ze zdrobnieniem, a dla pozostałych – normalne przywitanie.
Np.:
Bartłomiej lub Bartek -> Cześć, Bartuś!
Agnieszka -> Witaj, Aga!
Dorota -> Hej, Dorotko!
Maciej -> Jak się masz, Maćku!
inne imię -> <imię>, dzień dobry.

 ** Zadanie 2.

Zmodyfikuj warunki z konstrukcji `if-elif-else` z przykładu „za mało”, „dużo za dużo”, „zgadłeś” itp. tak, aby te napisy wystąpiły w innej kolejności, ale żeby działanie programu było identyczne.

 *** S Zadanie 2b.

Na ile sposobów można ustawić trzy napisy na trzech miejscach (np. te z przykładu „za dużo”, „za mało”, „zgadłeś”)? Czy dla każdej z tych kombinacji da się dobrać odpowiednie warunki w `if` i `else`? Uzasadnij.

 *** S Zadanie 3.

Powyżej podano przykład z wczytywaniem liczby i sprawdzaniem czy liczba jest większa od 5, mniejsza od 5 czy równa 5 i wypisywaniem odpowiedniego komunikatu. Załóżmy, że nie wiesz, jaka liczba jest wpisana w programie i chcesz ją zgadnąć. Wiesz tylko, że jest ona z zakresu np. 1-1000. Spróbuj znaleźć ogólny sposób (algorytm, schemat postępowania) zgadywania nieznanej liczby. Zastanów się, jak sprawić, abyś musiał podać jak najmniejszą liczbę strzałów, zanim trafisz.

➔ *Rozwiązania niektórych zadań*

Składnia if-elif-else

Zadanie 2b (szkic):

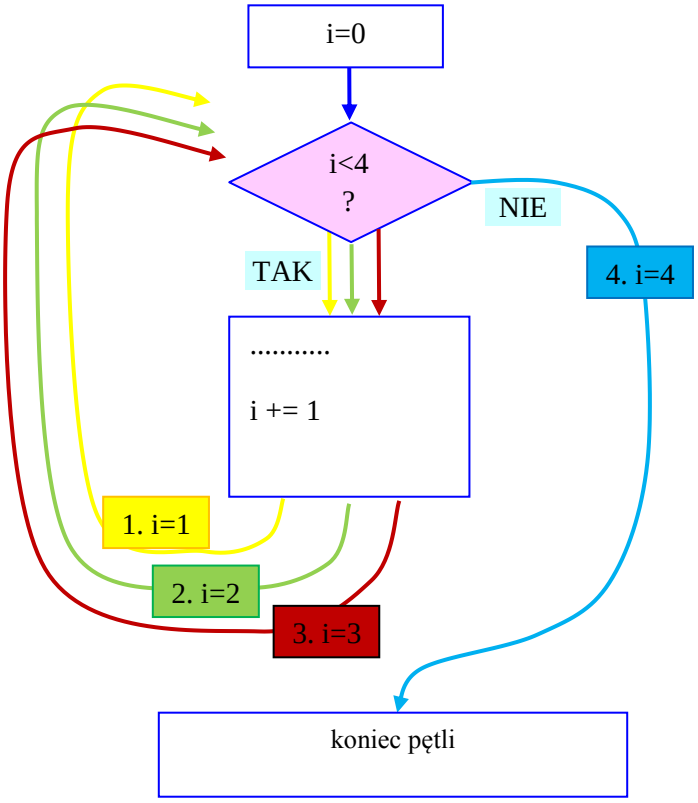
Kombinacji jest 3! = 6 i dla każdej z nich da się dobrać odpowiednie warunki w `if` i `else`.

Zadanie 3 (szkic):

Rozwiązaniem jest wyszukiwanie binarne od 1 do 1000, czyli pytanie zawsze o liczbę ze środka rozpatrywanego przedziału. Każda odpowiedź na pytanie o środkową liczbę zmniejsza nam rozpatrywany przedział do jego lewej lub prawej połowy.

--- WHILE

```
i = 0
while i < 4:
  print i
  i += 1
```




```
i = 0
while i < 5:
  if i == 3:
    print "trzy"
  else
    print i
  i += 1
```

* Zadanie 1.

Wypisz wszystkie liczby od 1 do 15.

* Zadanie 1b.

Wypisz wszystkie liczby nieparzyste od 1 do 15.

* Zadanie 2.

Przy użyciu pętli while wypisz, ile liczb od 1 do 15 dzieli się przez 7.
** Czy umiesz to policzyć bez pętli while?

** Zadanie 3.

Wypisz silnię z liczby n (n – zmienna wczytana z klawiatury).

** Zadanie 4.

Wypisz sumę liczb od 1 do n.

** Zadanie 4b.

Wypisz sumę nieparzystych liczb od 1 do n.

*** Zadanie 4c.

Wypisz sumę liczb pierwszych od 1 do n.

** Zadanie 5.

Fibonacci iteracyjny, n kroków (n – zmienna wczytana z klawiatury).

** Zadanie 6 (ciąg z problemu Collatza)

Dla danego n (wczytanego z klawiatury) wypisz kolejne liczby z ciągu Collatza.

$$c_{n+1} = \begin{cases} \frac{1}{2}c_n & \text{gdy } c_n \text{ jest parzysta} \\ 3c_n + 1 & \text{gdy } c_n \text{ jest nieparzysta} \end{cases}$$

źródło równania: http://pl.wikipedia.org/wiki/Problem_Collatza

** Zadanie 7.

Obliczyć NWD liczb 35 i 49, używając **algorytmu Euklidesa**.

** Zadanie 7b.

Obliczyć NWD liczb 35 i 49, używając **szybkiego algorytmu Euklidesa.**
(http://www.oeiizk.waw.pl/~witek/eliwww/strona2.html)

*** Zadanie 8 (kalkulator).

Napisz kalkulator. W menu głównym wyświetla się lista działań i odpowiadające im kolejne cyfry. Użytkownik wybiera cyferkę oznaczającą, które działanie ma zostać wykonane lub 0, aby wyjść z programu. Po wyborze działania program prosi o podanie pierwszej liczby, a następnie prosi o podanie drugiej liczby, po czym wyświetla wynik i menu główne. W przypadku dzielenia program nie pozwala na wpisanie 0 jako dzielnika.

--- LISTA (LIST)

Do tej pory poznawaliśmy typy proste. Teraz przyszedł czas na listę, zwaną czasem wektorem.

A	B	C	D
1	10.4	"abc"	False

zmienne proste

1	4	6.54	12	34.4	2	"ala"	-10	False	2

X lista, wektor

➔ *Przypisywanie, odwoływanie się*

Lista elementów

```
l = [12, 13, 34]
print type(l)
print len(l)
print l[0] # pierwszy element (numeracja od zera!)
print 13 in l
```

* Zadanie 1.

Na początku programu stwórz listę `elementy` z 8-10 elementami, a następnie wypisz jej pierwszy i trzeci element.

* Zadanie 2.

Na początku programu stwórz listę `elementy` z 8-10 liczbami. W dalszej części programu wypisz sumę drugiego, czwartego i siódmego jej elementu.

* Zadanie 3.

Na początku programu stwórz listę `elementy` z 8-10 elementami. W dalszej części programu wypisz jej pierwszy i ostatni element.

* Zadanie 4.

Na początku programu stwórz listę `elementy` z 8-10 elementami. W dalszej części programu sprawdź, czy znajduje się w niej liczba 24 i wypisz odpowiednio: „TAK” lub „NIE”.

** S Zadanie 4b.

Powyższe zadanie da się wykonać na co najmniej dwa sposoby (istotnie od siebie różne). Znajdź drugi sposób.

** S Zadanie 5.

Na początku programu stwórz listę `elementy` z 8-10 elementami. W dalszej części programu sprawdź pojedynczym porównaniem, czy istnieje w niej liczba 24.3 lub napis "24.3" i wypisz odpowiednio: „TAK” lub „NIE”.

** Zadanie 6.

Na początku programu stwórz listę `elementy` z 8-10 elementami, a następnie wybierz jakiś warunek logiczny, np. liczby większe od 23. W dalszej części programu policz, ile elementów z listy elementy spełnia ten warunek.
Wypisz na ekranie tę liczbę, a następnie spełniające ją elementy.

*** Zadanie 7.

Stwórz dwie listy `k` i `l` o tej samej liczbie elementów. Na liście `k` umieść napisy, a na liście `l` – liczby. Napisz program, który wypisze każdy z napisów z listy `k` tyle razy, ile wynosi odpowiednia liczba z listy `l`, np. dla:

```
k = ['Zajęcia', 'dla', 'ciekawych', 'świata']
l = [1, 4, 2, 5]
```

wypisze:

```
Zajęcia
dla
dla
dla
dla
ciekawych
ciekawych
świata
świata
świata
świata
świata
```

*** S Zadanie 8.

Wpisz do programu listę k o kilku elementach. Napisz program, który odwróci tę listę, tzn. sprawi, że wartość k[0] znajdzie się na ostatnim miejscu listy, k[1] na przedostatnim itd. Czy potrzeba do tego dodatkowej drugiej listy? Czy potrzeba do tego zmiennej pomocniczej?

➔ *Dodawanie i usuwanie elementów*

```
l = [12, "asd", 4.5, True, 3]
l.append(4)
l.append("qwerty")
print l
l.remove(4) # usuwa element o wartości 4 (błąd, gdy takiego elementu nie ma!!)
l.pop(3) # usuwa element o indeksie 3, czyli… czwarty!
print str(l)
```

* Zadanie 1.

Napisz program, który stworzy pustą listę `l` i ją wypisze, po czym doda do niej trzy wybrane przez Ciebie liczby i znowu ją wypisze. Liczby mają być wpisane wprost do kodu programu – bez pytania do użytkownika.

** Zadanie 2.

Napisz program, który do listy `l` będzie wczytywał kolejne dodatnie liczby z wejścia. Podanie `0` ma spowodować zakończenie wczytywania, wypisanie całej listy i jej długości, a następnie zakończenie programu.

Złożoność obliczeniowa

Gdy liczy się czas wykonania programu, podczas projektowania programu mówi się o jego złożoności obliczeniowej. Oznacza ona, jak szybko wzrasta czas działania programu, gdy zwiększymy ilość danych.
Jeśli na przykład dla danych 2 razy większych (np. dla 2 razy dłuższej listy liczb) program działa 2 razy dłużej, dla 3 razy większych – 3 razy, dla 4 razy większych – 4 razy, mówimy o złożoności liniowej. Bardziej ogólnie, mówimy tak, gdy dla n razy większych danych program działa c*n razy dłużej, gdzie c jest z góry określoną stałą, wartością niezmienną, niezależną od ilości danych.

Jeśli dla danych n razy większych program działa tak samo długo, mówimy o złożoności stałej. Jest to idealna sytuacja dla programisty. Niestety bardzo niewiele problemów da się rozwiązać w czasie stałym.
Jeśli dla danych n razy większych program działa $c*n^2$ razy dłużej, mówimy o złożoności kwadratowej. Analogicznie mówimy o sześciennej.

W podobny sposób można zdefiniować też inne złożoności, np. wykładniczą lub logarytmiczną.

Analogicznie można zdefiniować termin złożoność pamięciowa, opisując, ile maksymalnie pamięci zużywa program.

*** Zadanie 3. Usuwanie duplikatów z listy.

Napisz program, który z listy l usunie wszystkie powtórzenia elementów.
Oceń, jaka jest złożoność obliczeniowa Twojego programu. Czy na pewno masz wszystkie potrzebne dane? Jeśli nie, zapytaj prowadzącego.

„na sztywno”

W żargonie informatyków na dane wpisane w kod programu (w odróżnieniu od danych, o które pyta się użytkownika) mówi się często, że są wpisane „na sztywno” w kod programu.

*** Zadanie 4.

Napisz program z wpisaną „na sztywno” listą `l`, który będzie w pętli: pytał użytkownika, który element listy wypisać i wypisywał ten element listy. Ustal jedną wartość, która zamiast oznaczać indeks listy będzie oznaczać wyjście z programu. Jaką liczbę najlepiej do tego wybrać?
Zastanów się, czy wszystkie liczby wpisane przez użytkownika będą poprawne. Jeśli nie, niech program odpowiednio zareaguje na niepoprawne wartości.

Uwaga do powyższego zadania

Przez niepoprawne wartości rozumiemy liczby, które nie mają sensu dla indeksów (ujemne lub wykraczające poza długość listy), a nie o napisy, które nie mają wartości liczbowej, np. "Ala".
Z napisami poradzilibyśmy sobie dopiero obsługując wyjątek (poza podstawowym zakresem naszych zajęć) lub używając funkcji sprawdzającej, czy podano same cyfry.

➔ *Rozwiązania niektórych zadań*

Przypisywanie, odwoływanie się

Zadanie 4b (szkic):

Chodzi o rozwiązania: `24 in elementy` oraz `while z if`em.

Zadanie 5 (szkic):

Użyć while'a z warunkiem: str(a) == "24.3".

Zadanie 8 (szkic):

Oczywiście da się odwrócić listę w miejscu, zamieniając miejscami skrajne elementy.

„w miejscu"

O programie lub algorytmie, który, nie licząc pamięci na dane wejściowe, potrzebuje zawsze stałej dodatkowej wielkości pamięci, bez względu na rozmiar tych danych. Algorytm w powyższym zadaniu będzie w miejscu, jeśli nie będzie korzystał z dodatkowe listy, a jedynie ze zmiennej pomocniczej (lub więcej niż jednej zmiennej pomocniczej, ale ich liczba jest niezależna od długości listy).

--- * SŁOWNIK (DICT)

```
b = {"franek": 23, "bodzio": 345, 4: 19}
print type(b) # <type 'dict'>
print len(b) # 3
print b["franek"] # 23
print b["karol"] # KeyError: 'karol'
b["karol"] = 48 # w ten sposób można dodać nowy klucz
print b["karol"] # 48
print b[4]
print (4 in b) # True
print (5 in b) # False

for i in b:
   print i

# Kolejność kluczy nie jest zachowana:
# karol
# franek
# 4
# bodzio
```

*** Zadanie 1. Zliczanie wystąpień

Napisz program, który wczyta napis, a następnie wypisze częstość występowania liter (bez rozróżniania dużych i małych liter). Wypisz tylko te litery, które faktycznie występowały w tekście.

*** Zadanie 1b.

Zmodyfikuj program, aby wyświetlał także zera wystąpień dla liter, cyfr i znaku spacji.

*** Zadanie 2. Ulepszone usuwanie duplikatów

Używając słownika, popraw złożoność obliczeniową programu z działu z listami. Zakładając, że wstawienie do słownika, usunięcie ze słownika, sprawdzenie istnienia elementu oraz odczytanie wartości elementu to operacje ze złożonością logarytmiczną, powiedz, jaką złożoność ma Twój nowy algorytm.

*** Zadanie 3.

Korzystając z iteritems() (patrz dokumentacja: http://docs.python.org/library/stdtypes.html#dict.iteritems), stwórz słownik, który będzie miał zamienione klucze z wartościami. Jaki problem się pojawia? Jak proponujesz go rozwiązać?

--- PĘTLA FOR

➔ *for*

Podstawowa konstrukcja

```
l = [12, 13, 34]
for a in l:
  print a
```

 * R Zadanie 1.

Długość napisu przechowywanego w zmiennej `slowo` otrzymuje się (analogicznie do długości listy), pisząc: `len(slowo)`. Wiedząc o tym, wypisz listę słów z listy `l` wraz z ich długościami.

 ** Zadanie 2.

Napisz program, który z listy `l = [7, 8, 10, 15, 43, 48]` wypisze tylko liczby podzielne przez 5.

 *** Zadanie 3.

Napisz program, który wypisze tabliczkę mnożenia 10x10.

➔ *range*

1 argument

`range(<wartosc>)` generuje listę `<wartosc>` elementów. Np.:

```
print range(10) # lista 10 elementów: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

 ** R Zadanie 1.

Wypisz 15 gwiazdek (znaków „ * ”).

 * R Zadanie 2.

Ustaw zmienną `m` na jakąś wartość. Wypisz tyle małp („@”), ile wynosi wartość zmiennej `m`.

 * S Zadanie 3.

Jak w zad. 2, ale wypisz małpy w jednej linii.

 ** S Zadanie 4.

Jak w zad. 2, ale wypisz małpy w jednej linii i bez spacji pomiędzy nimi.

** Zadanie 5.

Napisz program, który ze znaczków „*” i „#” stworzy poziomą flagę o wymiarach podanych przez użytkownika. Na przykład dla liczb 2, 3, 15 narysuje:

```
###############
###############
***************
***************
***************
```

** R Zadanie 6.

Jak w zadaniu powyżej, ale flaga jest trójkolorowa: „*”, „#” i „.”(kropka). Możesz dla uproszczenia przyjąć, że wszystkie trzy paski mają taką samą wysokość.

Jak można prościej wyświetlić rządek:

```
print ''.ljust(n, '*') # tak można wyświetlić rządek
```

** Zadanie 7.

Napisz program, który ze znaczków „*” i „#” stworzy pionową flagę o wymiarach podanych przez użytkownika.

** Zadanie 8.

Jak w zadaniu powyżej, ale flaga jest trójkolorowa: „*”, „#” i „.” (kropka).

*** Zadanie 9.

Napisz program, który stworzy ramkę ze znaków: minus („-”), plus („+”) i pionowa kreska („|”). Poproś użytkownika o podanie długości i szerokości ramki. Nie pozwól na wprowadzenie niepoprawnych liczb.

2 i 3 argumenty

1) `range(<poczatek>, <koniec>)` generuje ciąg liczby od `<poczatek>` do **`<koniec> - 1`** co 1, np.:

```
print range(1, 11) # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

2) `range(poczatek, koniec, co_ile)` generuje ciąg liczby od `<poczatek>` do **`<koniec> - 1`** co `<co_ile>`, np.:

```
print range(15, 35, 7) # [15, 22, 29]
```

* R Zadanie 1. Liczby od 1 do 15.

Wypisz jeszcze raz wszystkie liczby od 1 do 15.

* R Zadanie 2. Liczby nieparzyste od 1 do 15.

Wypisz jeszcze raz wszystkie nieparzyste liczby od 1 do 15.

** Zadanie 3. Silnia.

Wypisz silnię z liczby n (n – wczytana z klawiatury).

** Zadanie 4. Suma liczb.

Wypisz sumę liczb od 1 do n.

** Zadanie 5. Suma nieparzystych.

Wypisz sumę nieparzystych liczb od 1 do n.

** Zadanie 6. Suma pierwszych.

Wypisz sumę liczb pierwszych od 1 do n.

➔ *Rozwiązania niektórych zadań*

for

Zadanie 1:

```
l = ["Ala","ma","kota","Reksia"]
for a in l:
  print a, ": ", len(a)
```

range – 1 argument

Zadanie 1:

```
for a in range(15):
  print "*"
```

Zadanie 2:

```
m = 123
for a in range(m):
  print "@"
```

Zadanie 3 (szkic):

print z przecinkiem na końcu

Zadanie 4 (szkic):

Konkatenacja (łączenie) napisów.

Zadanie 6:

```
szer = int(raw_input("Podaj szerokosc flagi: "))
wys = int(raw_input("Podaj wysokosc jednego koloru flagi: "))
rzadek = ['', '', '']
for a in range(szer):
        rzadek[0] += '*'
        rzadek[1] += '#'
        rzadek[2] += '.'

for i in range(len(rzadek)):
        for a in range(wys):
                print rzadek[i]
```

range – 2 i 3 argumenty

Zadanie 1:

```
print range(1, 16)
```

Zadanie 2:

```
print range(1, 16, 2)
```