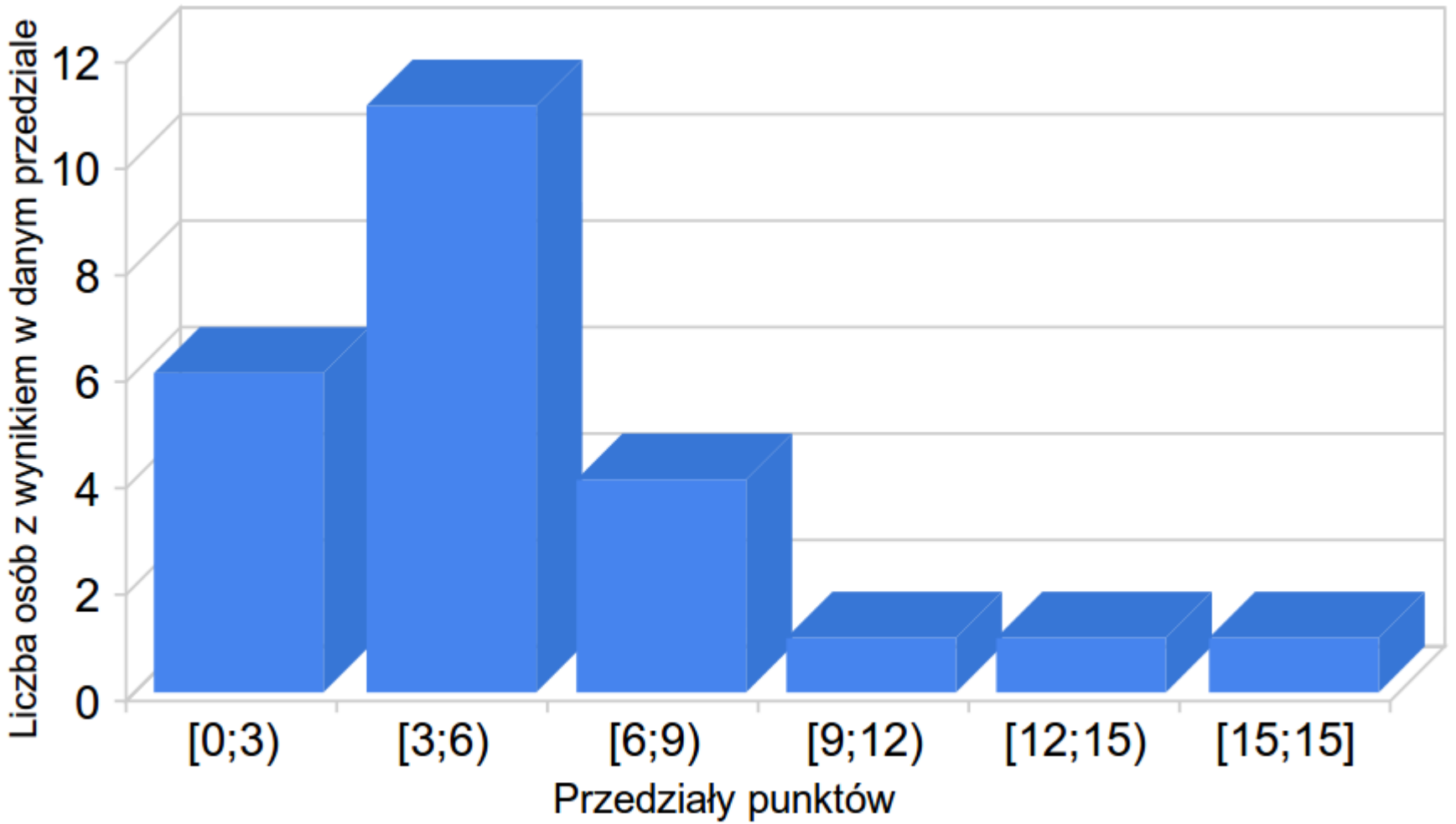
Wstęp do Informatyki dla Bioinformatyków

Wykład 5: Zliczanie i porządkowanie, Czyli o słownikach i zbiorach,

Bartek Wilczyński

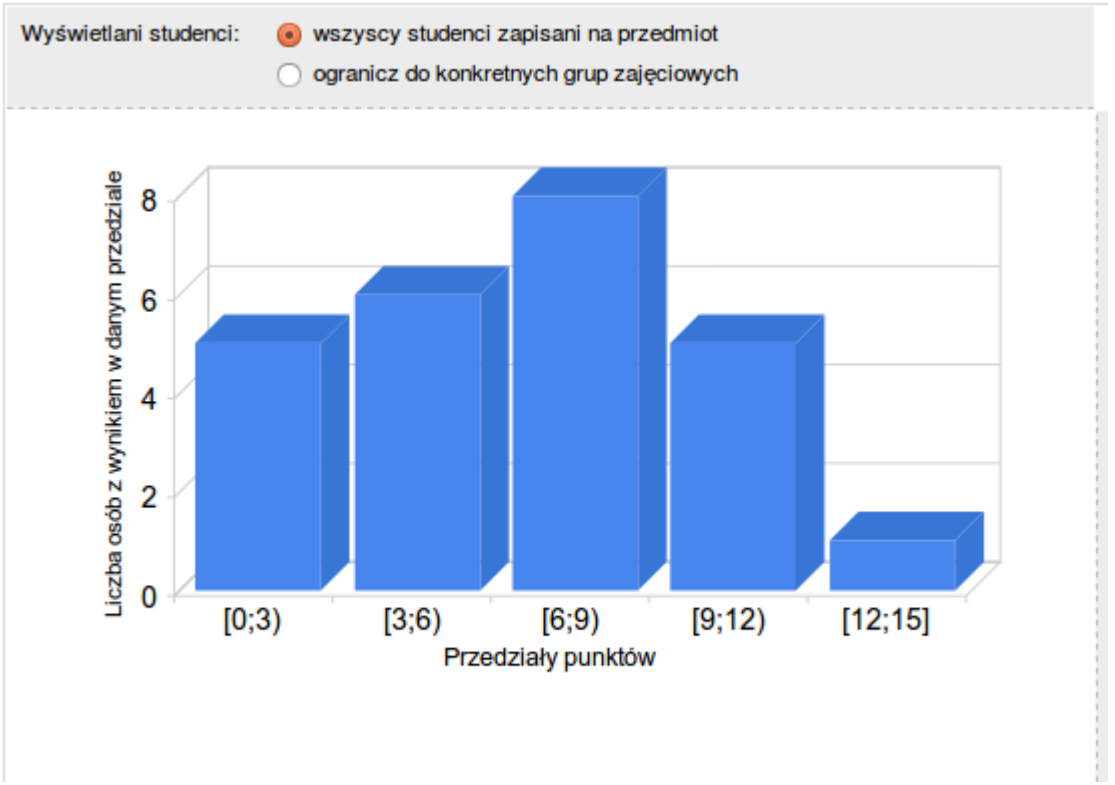
16.11.2015

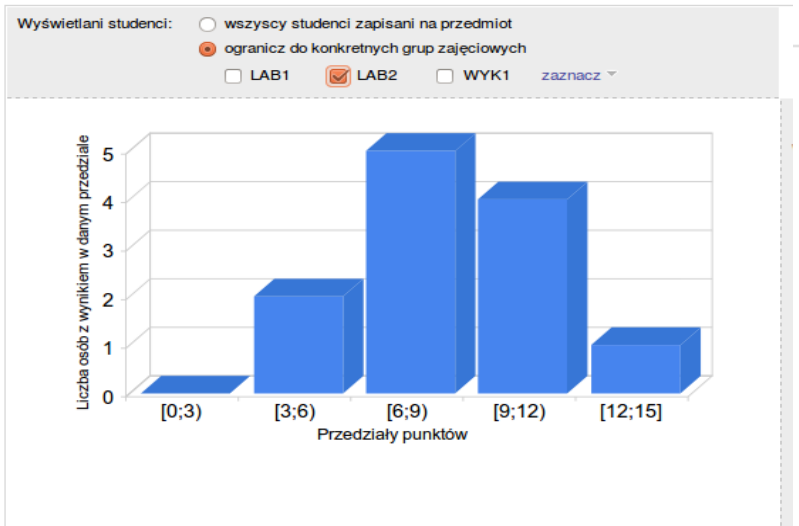
Wyniki kolokwium I

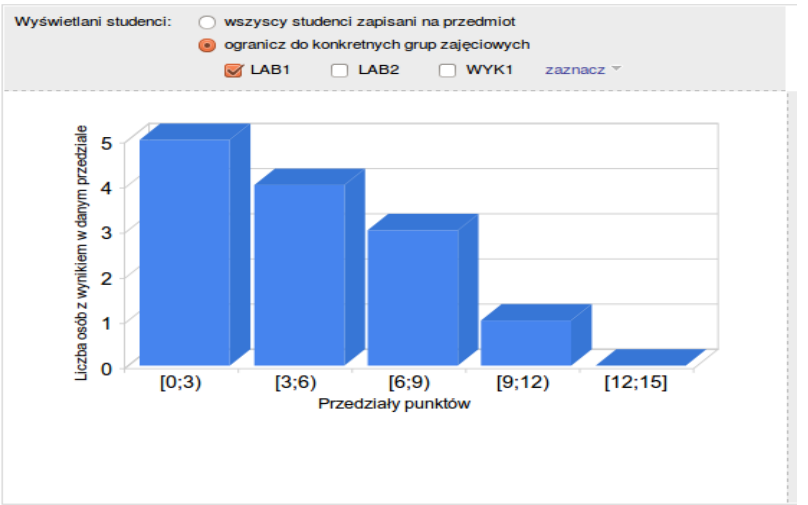


Z zeszłego roku...

Wyniki kolokwium z tego roku







Zadanie rozgrzewkowe:

- Mamy plik tekstowy, policzmy ile razy występuje w nim każde słowo.

```
def count_words(file):
    """counts words in a file"""
    lns=[ln.strip().split() for ln in file]
    words=[]
    counts=[]
    for line in lns:
        for word in line:
            if word not in words: # new word =>counts=1
                words.append(word)
                counts.append(1)
            else: #existing word
                i=words.index(word) #find it
                counts[i]+=1 #increase the counter
    return zip(words,counts)
```

Czy nie możnaby prościej?

- Często napotykamy na różne warianty problemu zliczania elementów
- Kiedy z góry nie można określić jakie elementy wystąpią w zliczaniu, albo gdy przestrzeń możliwych wartośći jest duża, trzeba takie dane składować w sposób dynamiczny, dostosowany do zmiennych danych
- Listy są bardzo nieefektywnym sposobem przechowywania takich zliczeń

Uogólnijmy nasz problem...

- Mamy do czynienia z sytuacją w której potrzebujemy **przechowywać** i **modyfikować** pewnego rodzaju **wartości** powiązane z **rozróżnialnymi** wartościami kluczowymi (“**kluczami**”)
- W przypadku zliczania, kluczami są słowa a wartościami liczby wystąpień; W książce telefonicznej kluczami są nazwiska a wartościami są numery telefonów, itp.
- Taką strukturę danych nazwiemy słownikiem

Słowniki w pythonie

- Możemy tworzyć słowniki przy użyciu nawiasów klamrowych i dwukropków

```
In [2]: d= { "ala" : 1234, "ela" : 4321}

In [3]: d["ala"]
Out[3]: 1234

In [4]: d["ula"]
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/home/bartek/<ipython-input-4-a534db8d13b8> in <module>()
----> 1 d["ula"]

KeyError: 'ula'

In [5]:
```

Sprawdzanie i modyfikacja wartości

- Możemy odwoływać się do wartości tak jak w listach, z tym że zamiast indeksu używamy klucza
- Możemy sprawdzać obecność klucza konstrukcją “in”

```
In [5]: "ula" in d
Out[5]: False

In [6]: d["ula"]=5678

In [7]: "ula" in d
Out[7]: True

In [8]: d["ula"]
Out[8]: 5678

In [9]: d["ula"]+=1

In [10]: d["ula"]
Out[10]: 5679
```

Słowniki umożliwiają też iteracje

- Dla słownika `d` możemy przeglądać jego
  - klucze,
  - wartości,
  - pary (klucz,wartość)

```
In [14]: for x in d:
    ....:     print x,
    ....:
ula ela ala

In [15]: for x in d.values():
    ....:     print x,
    ....:
5679 4321 1234

In [16]: for x in d.items():
    ....:         print x,
    ....:
('ula', 5679) ('ela', 4321) ('ala', 1234)
```

Inne operacje na słownikach

```
In [26]: d=dict([('ula', 5679),('ela', 4321),('ala', 1234)])
```

```
In [27]: d
Out[27]: {'ala': 1234, 'ela': 4321, 'ula': 5679}

In [28]: d.update({"ala":2345,"Hala":997})

In [29]: d
Out[29]: {'Hala': 997, 'ala': 2345, 'ela': 4321, 'ula': 5679}

In [30]: del d["ula"]

In [31]: d
Out[31]: {'Hala': 997, 'ala': 2345, 'ela': 4321}
```

```
In [32]: d.clear()

In [33]: d
Out[33]: {}
```

Czasem nie musimy zliczać...

- Jeśli interesuje nas tylko wyliczenie elementów, np. słów występujących w tekście, możemy użyć zbiorów
- Zbiory tworzy się łatwo z list z powtórzeniami:
  - `z=set([1, 2, 3, 1, 4])` → `set([1,2,3,4])`
- Można wykonywać operacje na zbiorach:
  - `z1 | z2` → suma,
  - `z1 & z2` → przecięcie
  - `z1 - z2` → różnica
  - `z1 ^ z2` → różnica symetryczna (xor)

Możemy też używać operacji na zbiorach w postaci przypisania

```
In [55]: s="ala ela ula ala ula hela".split()

In [56]: z=set(s)

In [57]: z
Out[57]: set(['ula', 'ela', 'hela', 'ala'])

In [58]: z|=set(["ula","lila"])

In [59]: z
Out[59]: set(['ela', 'hela', 'lila', 'ula', 'ala'])

In [60]: z-=set(["hela","lala"])

In [61]: z
Out[61]: set(['ela', 'lila', 'ula', 'ala'])
```

Nasze zadanie jeszcze raz:

```
def count_words(file):
    """count words in a file"""
    lns=[ln.strip().split() for ln in file]
    word_counts={}
    for line in lns:
        for word in line:
            if word not in word_counts:
                word_counts[word]=1
            else:
                word_counts[word]+=1
    return word_counts.items()
```

Przydatny moduł collections

- Ciekawe wersje słowników
  - collections.Counter
  - collections.OrderedDict
  - collections.defaultdict

Zliczanie z collections.Counter

```
def count_words(file):
    import collections
    lns=[ln.strip().split() for ln in file]
    word_counts=collections.Counter()
    for line in lns:
        for word in line:
            word_counts[word]+=1
    return word_counts.items()
```

Zliczanie z collections.defaultdict

```
def count_words_in_lines(file):
    import collections
    lns=[ln.strip().split() for ln in file]
    word_counts=collections.defaultdict(list)
    for i,line in enumerate(lns):
        for word in line:
            word_counts[word].append(i)
    return word_counts.items()
```

```
In [95]: count_words_in_lines(s)
Out[95]: [('ula', [0, 1]), ('ela', [0]), ('hela', [1]), ('ala', [0, 1])]
```