Task 1. SumThreeFive

Напишите функцию, которая принимает один аргумент n и возвращает сумму всех натуральных чисел, меньших n и таких, которые делятся на 3 или на 5.

In [1]:
```
def sumThreeFive(n):
    sum35 = 0
    for i in range(1,n):
        if (i%3==0 or i%5==0):
            sum35 += i
    return sum35

sumThreeFive(10000)
```

Out[1]: 23331668

Task 2. Fibo

Напишите функцию, которая принимает один аргумент n и возвращает n-й член последовательности Фибоначчи. При этом запрещено использовать циклы (т. е. использование рекурсии обязательно).

In [2]:
```
def fibo(n):
    if (n == 1):
        return (1)
    elif (n == 2):
        return (2)
    return (fibo(n-1)+fibo(n-2))

# It takes too long time to calculate fibo(200) with recursive function,
 so I will calculate fibo(20)
fibo(20)
```

Out[2]: 10946

Task 3. Anagrams

Напишите программу, которая загружает список слов из файла words-list-russian.txt и выводит на экран все классы анаграмм, состоящие не менее, чем из четырех слов.

In [3]:
```
def anagrams(name):

    # Read table
    import pandas as pd
    df = (pd.read_table(name, header = None))[0].tolist()
    data = []
    anagr = []

    # Sort letters in these words in alphabetic order
    for word in df:
            word.split()
            word = ''.join(sorted(word))
            data.append(word)

    # Find words with similar letters and include only class of words wi
th at least 4 anagrams
    for i in range(len(df)):
        anagr1 = [df[i]]
        for j in range(i+1,len(df)):
            if data[i] == data[j]:
                anagr1.append(df[j])
        if len(anagr1) >= 4:
            anagr.append(anagr1)

    # Clear from repeated anagram classes identifying repetitions by las
t element
    final = []
    for i in range(len(anagr)):
        if all(list(set(anagr[j])&set(anagr[i]))==[] for j in range(i)):
            final.append(anagr[i])

    return final

anagrams("words-list-russian.txt")
```

Out[3]:
```
[['аборт', 'обрат', 'табор', 'торба'],
 ['автор', 'втора', 'отвар', 'рвота', 'тавро', 'товар'],
 ['арак', 'арка', 'кара', 'рака'],
 ['калан', 'канал', 'ланка', 'накал'],
 ['корт', 'крот', 'торк', 'трок'],
 ['рост', 'сорт', 'торс', 'трос'],
 ['секта', 'сетка', 'стека', 'тесак']]
```

Task 4. TypeSetter

Игра состоит в том, чтобы найти как можно больше слов, которые можно составить из букв, входящих в заданное слово. Например, из букв слова «кильватер» можно составить такие слова как «киль», «кит», «литр» и другие. Напишите программу, которая загружает список слов из файла words-list-russian.txt и выводит на экран все слова, которые можно сложить из букв слова «лекарство».

In [4]:
```
def TypeSetter(source, word):

    # Read table & the base word
    import pandas as pd
    df = (pd.read_table(source, header = None))[0].tolist()
    letters = []

    # Let's count number of certain letters in the base word
    word_dic = {}
    for i in word:
        if i not in word_dic:
            word_dic[i] = 1
        if i in word_dic:
            word_dic[i] =+ 1

    # Now let's make a loop and compare words from list with our base wo
rd
    final = []
    for i in df:
        letters = {}
        count = 0
        for j in i:
            if j not in letters:
                letters[j] = 1
            elif j in letters:
                letters[j] += 1
        for k,v in letters.items():
            if (k in word_dic) and (v <= word_dic[k]):
                count += 1
        if count == len(i):
            final.append(i)
    print('Число слов, составленных из слова \'%s\': %s' % (word, len(final
)))
    return final

TypeSetter("words-list-russian.txt", "лекарство")
```

```
Число слов, составленных из слова 'лекарство': 147
```

```
Out[4]: ['автол',
         'автор',
         'акр',
         'ар',
         'арест',
         'арк',
         'аскер',
         'ваер',
         'валет',
         'вар',
         'век',
         'веко',
         'вектор',
         'вера',
         'верк',
         'верстак',
         'вес',
         'весло',
         'вест',
         'ветка',
         'ветла',
         'влас',
         'вокал',
         'вол',
         'вор',
         'ворс',
         'воск',
         'втора',
         'ер',
         'кал',
         'карел',
         'карст',
         'карт',
         'картвел',
         'кастор',
         'кат',
         'катер',
         'квас',
         'квестор',
         'квота',
         'кларет',
         'клот',
         'ков',
         'кол',
         'кола',
         'колет',
         'кора',
         'корвет',
         'корсет',
         'корт',
         'костра',
         'кот',
         'креол',
         'кресло',
         'крест',
         'кроат',
         'кров',
```

```
            'крот',
            'лавр',
            'лар',
            'латекс',
            'левкас',
            'лек',
            'лектор',
            'лерка',
            'лес',
            'леска',
            'лестовка',
            'лето',
            'леток',
            'лов',
            'лоск',
            'овал',
            'окрас',
            'орс',
            'орт',
            'оса',
            'осек',
            'оскал',
            'ост',
            'отвал',
            'отвар',
            'отвес',
            'отсев',
            'отсек',
            'рвота',
            'река',
            'рекостав',
            'ров',
            'рол',
            'роса',
            'рост',
            'рота',
            'сверка',
            'сверло',
            'сев',
            'севр',
            'секатор',
            'секта',
            'село',
            'сера',
            'серв',
            'сет',
            'сетка',
            'скало',
            'скатол',
            'сквер',
            'склера',
            'скол',
            'скот',
            'словак',
            'сова',
            'совет',
            'сок',
```

```
'солевар',
'солка',
'сор',
'сорт',
'срок',
'старое',
'створка',
'стек',
'стека',
'стекло',
'стекловар',
'стела',
'сток',
'стокер',
'стрела',
'стрелок',
'тавр',
'тавро',
'тал',
'талес',
'тело',
'тесак',
'тесло',
'товар',
'тол',
'торк',
'торс',
'трак',
'трал',
'трек',
'треск',
'трок',
'трос']
```

Task 5. GuessWord 1

In [5]:
```
def guessword(source):

    import random
    import pandas as pd
    df = (pd.read_table(source, header = None))[0].tolist()

    # Choose only words with length 5
    df = [i for i in df if len(i) == 5]

    playing = True
    word = random.sample(df,1)
    guesses = 0

    print(word)
    while playing:
        count = 0
        user_guess = input('Введите слово: ')
        if user_guess == 'закончить':
            break
        elif user_guess not in df:
            print('Не знаю такого слова')
            guesses += 1
            continue
        for element in user_guess:
            if element in [i for i in str(word)]:
                count += 1
        guesses += 1
        if count == 5:
            playing = False
            print('!')
            print('Угадали с попытки номер %s' %guesses)
        else: print(count)

    return word

guessword("words-list-russian.txt")
```

```
['багор']
Введите слово: книга
2
Введите слово: забор
Не знаю такого слова
Введите слово: багор
!
Угадали с попытки номер 3
```

Out[5]:
```
['багор']
```

Task 6. GuessWord 2

In [6]:
```
def guessword(source):

    import random
    import pandas as pd
    df = (pd.read_table(source, header = None))[0].tolist()

    # Choose only words with length 5
    df = [i for i in df if len(i) == 5]

    playing = True
    word = random.sample(df,1)
    guesses = 0

    print(word)
    print('Начинаю угадывать!')
    while playing:
        count = 0
        user_guess = input()
        if user_guess == 'закончить':
            break
        elif user_guess not in df:
            print('Не знаю такого слова')
            guesses += 1
            continue
        for element in user_guess:
            if element in [i for i in str(word)]:
                count += 1
        guesses += 1
        if count == 5:
            playing = False
            print('!')
            print('Я угадал с попытки номер %s' %guesses)
        else: print('Введите число совпадающих букв: %s' %count)

    return word

guessword("words-list-russian.txt")
```

```
['слизь']
Начинаю угадывать!
слезь
Не знаю такого слова
книга
Введите число совпадающих букв: 1
забор
Не знаю такого слова
багор
Введите число совпадающих букв: 0
слизь
!
Я угадал с попытки номер 5
```

Out[6]:
```
['слизь']
```

Task 7. Web importing

In [7]:
```
def BelarusGDP():

    import urllib.request
    from bs4 import BeautifulSoup

    url = 'http://www.belstat.gov.by/ofitsialnaya-statistika/makroekonomika-i-okruzhayushchaya-sreda/natsionalnye-scheta/godovye-dannye_11/proizvodstvo-valovogo-vnutrennego-produkta/'
    html = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(html, 'html.parser') #class object creation
    tags = soup.find_all('p', href=False)

    # First line
    data = []
    for i in tags[2:11]:
        data.append(i.contents[0].strip())
    print('\t'.join(data))

    # Second line
    data = []
    start_of_second = tags[11].contents[0].strip()
    for i in tags[20:29]:
        data.append(i.contents[0].strip())
    data[0] = start_of_second+" "+data[0]
    for i in range(1,len(data)):
        data[i] = data[i].replace(" ","").replace(u'\xa0', '')
    print('\t'.join(data))

    # Third line
    data = []
    for i in tags[29:38]:
        data.append(i.contents[0].strip())
    for i in range(1,len(data)):
        data[i] = data[i].replace(",",".")
    print('\t'.join(data))

    # Forth line
    data = []
    for i in tags[38:47]:
        data.append(i.contents[0].strip())
    for i in range(1,len(data)):
        data[i] = data[i].replace(u'\xa0', '')
    print('\t'.join(data))

BelarusGDP()
```

```
        2009    2010    2011    2012    2013    2014    2015    2016
Валовой внутренний продукт в текущих ценах,    142091  170466  307245  547617  670688  805793  899098  94949
в сопоставимых ценах,       x       107.7   105.5   101.7   101.0   101.7   96.2    97.5
Валовой внутренний продукт на душу населения, тыс. руб. 14946   17962   32433   57860   70852   85048   94745   9993
```