Data Analysis in Python, Home Assignment 2. By Vanchev Lev.

Отчет.

Задание 1:

Функция **factorize**(a) возвращает все простые делители числа a. Внутри задействована функция **isprime**(a), определяющая, является ли число простым. Отдельно рассмотрен случай для a = 2.

Результат применения функции:

In:

```
factorize(123456789)
```

Out:

```
[3, 3, 3607, 3803]
```

In:

```
factorize(12345678901234567890123456789012345678901234567890)
```

Out:

```
[2, 3, 3, 5, 73, 101, 137, 3541, 3607, 3803, 27961, 1676321, 5964848081]
```

Код:

```
def isprime(a):
    c=int(round(a**0.5))
    if a<=1:
        return False
    else:
        for i in range(2, c+1):
            if a%i == 0:
                return False
        return True


def factorize(a):
    divs=[]
    n=a
    while n!=1:
        if n==2:
            divs.append(2)
            return divs
        c=int(round(n**0.5))
        for i in range(2, c+1):
```

```
        if n%i == 0 and isprime(i):
            divs.append(i)
            n=n//i
            break
        if i==c:
            divs.append(n)
            return divs
```

Задание 2:

Функция **queens**() расставляет 8 ферзей на доске 8x8 так, чтобы ни один из них не был под атакой.

Результат применения функции:

In:

queens()

Out:

.......Q

Q.......

....Q...

..Q.....

......Q.

...Q....

.....Q..

.Q......

Код:

```
def queens():
    import random
    a=random.randint(1,8)
    b=random.randint(1,8)
    c=random.randint(1,8)
    d=random.randint(1,8)
    e=random.randint(1,8)
    f=random.randint(1,8)
    g=random.randint(1,8)
    h=random.randint(1,8)
    while b==a or b-1==a or b+1==a:
```

```
    b=random.randint(1,8)
  while c==a or c==b or c-1==b or c+1==b:
    c=random.randint(1,8)
  while d==a or d==b or d==c or d-1==c or d+1==c:
    d=random.randint(1,8)
  while e==a or e==b or e==c or e==d or e-1==c or e+1==c:
    e=random.randint(1,8)
  while f==a or f==b or f==c or f==d or f==e or f-1==e or f+1==e:
    f=random.randint(1,8)
  while g==a or g==b or g==c or g==d or g==e or g==f or g-1==f or g+1==f:
    g=random.randint(1,8)
  while h==a or h==b or h==c or h==d or h==e or h==f or h==g or h-1==g or h+1==g:
    h=random.randint(1,8)
  k='.......'
  s1=k[:a-1]+"Q"+k[a-1:]
  s2=k[:b-1]+"Q"+k[b-1:]
  s3=k[:c-1]+"Q"+k[c-1:]
  s4=k[:d-1]+"Q"+k[d-1:]
  s5=k[:e-1]+"Q"+k[e-1:]
  s6=k[:f-1]+"Q"+k[f-1:]
  s7=k[:g-1]+"Q"+k[g-1:]
  s8=k[:h-1]+"Q"+k[h-1:]
  print(s1)
  print(s2)
  print(s3)
  print(s4)
  print(s5)
  print(s6)
  print(s7)
  print(s8)
```

Задание 3:

Функция **frequencies**(a) открывает указанный текстовый файл и проводит частотный анализ.

Результат применения функции:

In:

frequencies('tolstoi.txt')

Out:

а : 8.19 %

б : 1.75 %

в : 4.65 %

г : 2.00 %

д : 2.93 %

е : 8.56 %

ё : 0.00 %

ж : 1.03 %

з : 1.70 %

и : 6.55 %

й : 1.14 %

к : 3.41 %

л : 5.09 %

м : 3.17 %

н : 6.50 %

о : 11.42 %

п : 2.51 %

р : 4.25 %

с : 5.46 %

т : 5.89 %

у : 2.68 %

ф : 0.13 %

х : 0.83 %

ц : 0.31 %

ч : 1.46 %

ш : 0.93 %

щ : 0.36 %

ъ : 0.03 %

ы : 1.87 %

ь : 2.05 %

э : 0.33 %

ю : 0.65 %

я : 2.13 %

Код:

```
def frequencies(a):
   file = open(a, 'r', encoding='utf-8')
   s=file.read()
   s=s.lower()
   list=['а','б','в','г','д','е','ё','ж','з','и','й','к','л',\
   'м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ъ','ы',\
   'ь','э','ю','я']
   counts=[]
   freq=[]
   for elem in list:
      counts.append(s.count(elem))
   for elem in counts:
      freq.append(elem/sum(counts)*100)
   for i in range(0, len(freq)):
      print(list[i],":", "%.2f" % freq[i],"%")
   file.close
```

Задание 4:

Функция **decipher**(a, b) проводит частотный анализ (при помощи функции **freq**() происходит запись частот в список) двух текстов, упорядочивает буквы по частоте применения и заменяет их.

Результат применения функции (пример):

In:

decipher(‘tolsoti.txt’, ‘tolstoi.enc’)

Out:

Ао не успел еце Кыер решитысь на ответ, которгй он сделает, как сама ярафинь, в белом атласном халате, шитом серебром, и в простгх волосах (две ояромнге косг en diad?me[6 - Лиademою.] ояибали два раза ее прелестную яолову) вошла в комнату спокойно и величественно; толыко на мраморном, несколыко вгпуклом лбе ее бгла морцинка янева. Эна с своим все вгдерживаюцим спокойствием не стала яовориты при камердинере. Эна знала о дуэли и пришла яовориты о ней. Эна дождаласы, пока камердинер уставил кофей и вгшел. Кыер робко через очки посмотрел на нее, и как заьщ, окруженнгй собаками, прижимаь уши, продолжает лежаты в виду своих враяов, так и он попробовал продолжаты читаты; но чувствовал, что это бессмгсленно и невозможно, и опьты робко взяльнул на нее. Эна не села и с презрителыной улгбкой смотрела на неяо, ожидаь, пока вгйдет камердинер.

– Щто еце что? Ыто вг наделали, ь вас спрашиваю? – сказала она строяо.

– М?.. что? ь… – сказал Кыер

Код:

```
def freq(a):
    file = open(a, 'r', encoding='utf-8')
    s=file.read()
    s=s.lower()
    list=['а','б','в','г','д','е','ё','ж','з','и','й','к','л',\
    'м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ъ','ы',\
    'ь','э','ю','я']
    counts=[]
    freq=[]
    for elem in list:
        counts.append(s.count(elem))
    for elem in counts:
        freq.append(elem/sum(counts)*100)
    file.close
    return freq
#%%
freq('tolstoi.txt')
#%%
def decipher(a,b):
    list=['а','б','в','г','д','е','ё','ж','з','и','й','к','л',\
    'м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ъ','ы',\
    'ь','э','ю','я']
    freqsource=freq(a)
    freqenc=freq(b)
    pmaxsource=100
    psource=0
    lsource=""

    pmaxenc=100
    penc=0
```

```
lenc=""


dct={}
while len(dct)<len(list):
    for i in range(0, len(freqsource)):
        if freqsource[i]>psource and freqsource[i]<pmaxsource:
            psource=freqsource[i]
            lsource=list[i]
    for j in range(0, len(freqenc)):
        if freqenc[j]>penc and freqenc[j]<pmaxenc:
            penc=freqenc[j]
            lenc=list[j]


    dct[lenc]=lsource


    pmaxsource=psource
    pmaxenc=penc


    psource=0
    penc=0


file = open(b,'r',encoding='utf-8')
for line in file:
    for char in line:
        if char in dct.keys():
            print(dct[char], end = '')
        else:
            print(char,end = '')
```

Задание 5:

Класс **Data** определен и выводит статистику списка элементов.

Результат применения функции:

In:

```
x = Data([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
```

print(x)

Out:

Count: 15

Mean: 8.0

Variance: 18.666666666666668

Standard deviation: 4.320493798938574

2nd moment: 82.66666666666667

3rd moment: 960.0

4th moment: 11887.466666666667

Median: 8

25% quantile: 4

75% quantile: 12

Код:

```
class Data():
    def __init__(self, data):
        self.data=data
    def count(self):
        return len(self.data)
    def sum(self):
        return sum(self.data)
    def moment(self, k):
        return sum(elem**k for elem in self.data)/self.count()
    def mean(self):
        return self.moment(1)
    def var(self):
        return sum((elem-self.mean())**2 for elem in (self.data))/self.count()
    def std(self):
        return self.var()**0.5
    def percentile(self, p):
        elem=int(p*self.count())+(self.count()%4 > 0)
        return self.data[elem-1]
    def median(self):
        return self.percentile(0.5)
    def __str__(self):
```

```
        print("Count:", self.count())
        print("Mean:", self.mean())
        print("Variance:", self.var())
        print("Standard deviation:", self.std())
        print("2nd moment:", self.moment(2))
        print("3rd moment:", self.moment(3))
        print("4th moment:", self.moment(4))
        print("Median:", self.median())
        print("25% quantile:", self.percentile(0.25))
        print("75% quantile:", self.percentile(0.75))
        return ""
```

Задание 6:

Функция **CPI**(url) использует **cpitoinf**(cpi) для конвертации индексов цен в инфляцию и применяет класс **Data** для вывода статистики.

Результат применения функции:

In:

```
x=CPI("http://sophist.hse.ru/exes/tables/CPI_Y_CHI.htm")
print(x)
```

Out:

```
Count: 24
Mean: -5.684754367566122
Variance: 499.20607305859136
Standard deviation: 22.342919976104096
2nd moment: 531.5225052781533
3rd moment: -6212.635429937721
4th moment: 1858845.203732865
Median: -0.7162041181736769
25% quantile: -25.97613882863341
75% quantile: -2.481617647058826
```

Код:

```
import urllib.request
from bs4 import BeautifulSoup
import data
def cpitoinf(cpi):
```

```
    inf=[]
    for i in range(1, len(cpi)):
        inf.append((cpi[i]-cpi[i-1]) / cpi[i-1]*100)
    return inf
#%%
def CPI(url):
    with urllib.request.urlopen(url) as resp:
        page=resp.read()
    soup=BeautifulSoup(page, 'html.parser')
    table=soup.find_all("td", {'align': 'right'})
    CPI_row=[]
    for elem in table:
        if ',' in elem.text:
            CPI_row.append(float(elem.text.replace(",", ".")))
    infl = data.Data(cpitoinf(CPI_row))
    return infl
```