[**Питон зачет**](#_b67ul2qvwaqa)[**2**](#_b67ul2qvwaqa)

[**Описание зачета**](#_oqk5tx5ut3mb)[**2**](#_oqk5tx5ut3mb)

[**Гит**](#_7cbsznubbygz)[**3**](#_7cbsznubbygz)

[**Задача 1**](#_vstdlq7cz6lj)[**6**](#_vstdlq7cz6lj)

[Задача 1.1](#_q37a4js1km58) [6](#_q37a4js1km58)

[Шаблон.](#_6zzn17ngky3) [6](#_6zzn17ngky3)

[Пример:](#_vsnd0a9v3lys) [6](#_vsnd0a9v3lys)

[Задача 1.2](#_9qazucnajeux) [7](#_9qazucnajeux)

[Разобранный пример.](#_700pqqasjsb9) [7](#_700pqqasjsb9)

[Задача 1.3](#_xgyjmcb8ik3v) [8](#_xgyjmcb8ik3v)

[Разобранный пример 1.](#_7ys0lawj0ici) [8](#_7ys0lawj0ici)

[Разобранный пример 2.](#_bcymxs9n9zcg) [9](#_bcymxs9n9zcg)

[Задача 1.4](#_oz9rdnot19bu) [10](#_oz9rdnot19bu)

[Разобранный пример.](#_nnsl4d7jybhy) [10](#_nnsl4d7jybhy)

[Задача 1.5](#_f3ee19gugvhn) [11](#_f3ee19gugvhn)

[Шаблон.](#_zch09b20cio3) [11](#_zch09b20cio3)

[Пример 1.](#_ckmitofnh0ny) [11](#_ckmitofnh0ny)

[Пример 2.](#_7bn0fqd6ngjm) [12](#_7bn0fqd6ngjm)

[**Задача 2**](#_luaehqoctyg8)[**13**](#_luaehqoctyg8)

[Задача 2.6](#_woc32ymj20yw) [13](#_woc32ymj20yw)

[Разобранный пример.](#_lhetpvvnvd5h) [13](#_lhetpvvnvd5h)

[Задача 2.7](#_vb660ovbghm) [15](#_vb660ovbghm)

[Вариант с двоичной маской.](#_8gc0jmqrqmxy) [15](#_8gc0jmqrqmxy)

[Вариант с 16-ой маской.](#_ps6l1tuxx0je) [17](#_ps6l1tuxx0je)

[Задача 2.8](#_dpmn09pt6hzs) [18](#_dpmn09pt6hzs)

[Разбор.](#_4tmlinh92l9o) [18](#_4tmlinh92l9o)

[Пример 1.](#_uhuisr2i90q) [18](#_uhuisr2i90q)

[Пример 2.](#_6yd3lrmd5xis) [19](#_6yd3lrmd5xis)

[Пример 3.](#_l25jnu3lv47l) [20](#_l25jnu3lv47l)

[Задача 2.9](#_bk56txew74b2) [22](#_bk56txew74b2)

[Разбор.](#_xqgg5sya4v4i) [22](#_xqgg5sya4v4i)

[Пример.](#_cudhuiswgjnp) [23](#_cudhuiswgjnp)

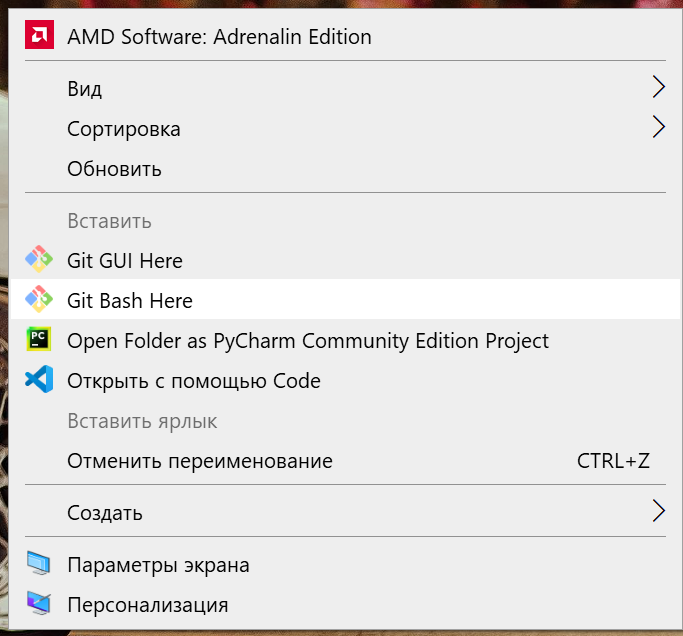
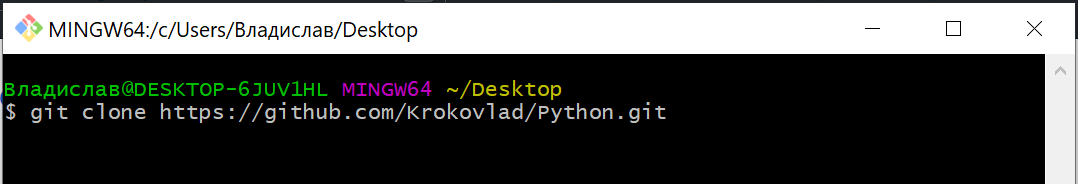
# Питон зачет

## Описание зачета

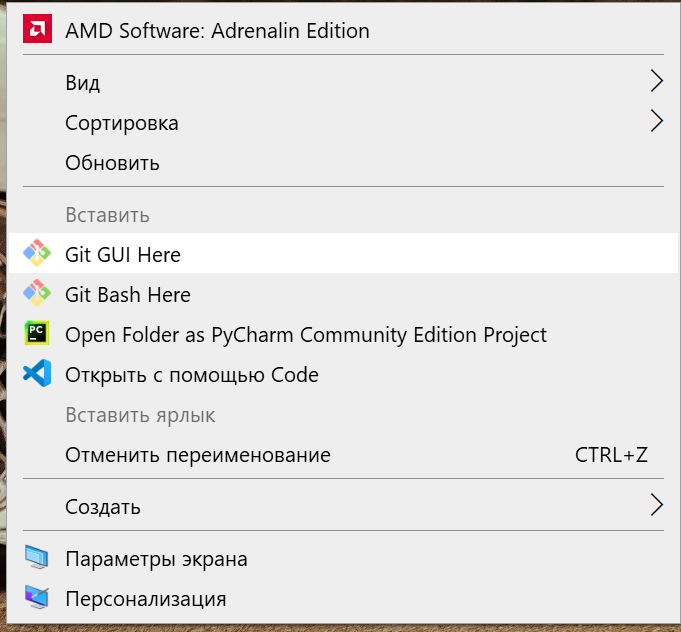
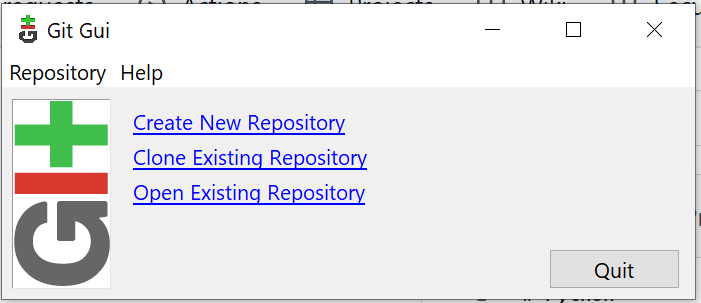
2 задачи на 50 минут (каждая 25 минут) первые задачи: Задача 1.1-1.5, вторые: 2.6-2.9. На гите находятся шаблоны.

## Гит

Шаблоны Чтобы подтянуть шаблоны через bash:

1. Зайти в папку, куда вы хотите сохранить файлы
2. Нажать пкм
3. Выбрать пункт "Git Bash here" 
4. Прописать команду git clone <https://github.com/Krokovlad/Python.git> 
5. Наслаждаться скачанными шаблонами

Альтернативный способ:

1. Зайти в папку, куда вы хотите сохранить файлы
2. Нажать пкм
3. Выбрать пункт "Git GUI here" 
4. Выбрать пункт "Clone existing repository" 
5. В поле source location прописать <https://github.com/Krokovlad/Python.git>
6. В поле Target destination прописать папку, куда вы хотите сохранить файлы
7. Жмакнуть "Clone"

## 

## Задача 1

### Задача 1.1

#### Шаблон.

Реализовать функцию:

**Просто сделать заданную формулу.**

from math import \*

def main(x):

return #Формула

Всё!

Нужные методы из math:

**log(значение, основание):** если основание не указано, то это натуральный

**log2** - по основанию 2

**log10** - по основанию 10

**fabs, sqrt** - модуль и корень

**ceil** - округление вверх с любым остатком (**символ ⌈ ⌉** - это округление в большую сторону, для него используется **ceil**)

**floor** - округление вниз (**символ ⌊ ⌋** - это округление в меньшую сторону, для него используется **floor**)

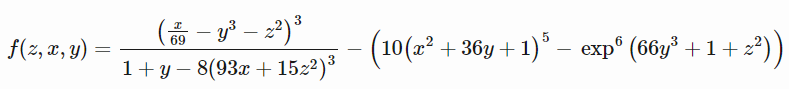
**sin, cos, asin, acos, tan, atan**

**exp(x)** - е в степени x

**pow(основание, степень)** - возведение основания в степень. **Лучше использовать две звездочки (основание\*\*степень). Например 2\*\*8 это 2 в 8 степени**

#### Пример:

**ИВБО-07-20 Вариант 13**



import math

def main(z, x, y):

f = ((x/69-y\*\*3-z\*\*2)\*\*3)/(1+y-8\*(93\*x+15\*z\*\*2)\*\*3)

f -= 10\*(x\*\*2+36\*y+1)\*\*5-math.exp(66\*y\*\*3+1+z\*\*2)\*\*6

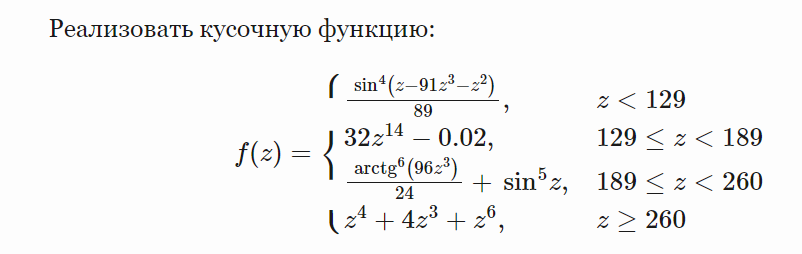
return f

### 

### Задача 1.2

#### Разобранный пример.

Реализовать кусочную функцию. При конкретном значении переменной нужно написать конкретную функцию.



Прописываем n условий (в данном случае 4), они находятся в правой части (в данном случае для z). Для каждого условия возвращаем то, что в левой части.

from math import \*

def main(z):

if (z<129):

return (sin(z-91\*z\*\*2-z\*\*2)\*\*4)/89

elif (z<189):

return (32\*z\*\*14)-0.02

elif (z<260):

return ((atan(96\*z\*\*3)\*\*6)/24)+sin(z)\*\*5

else:

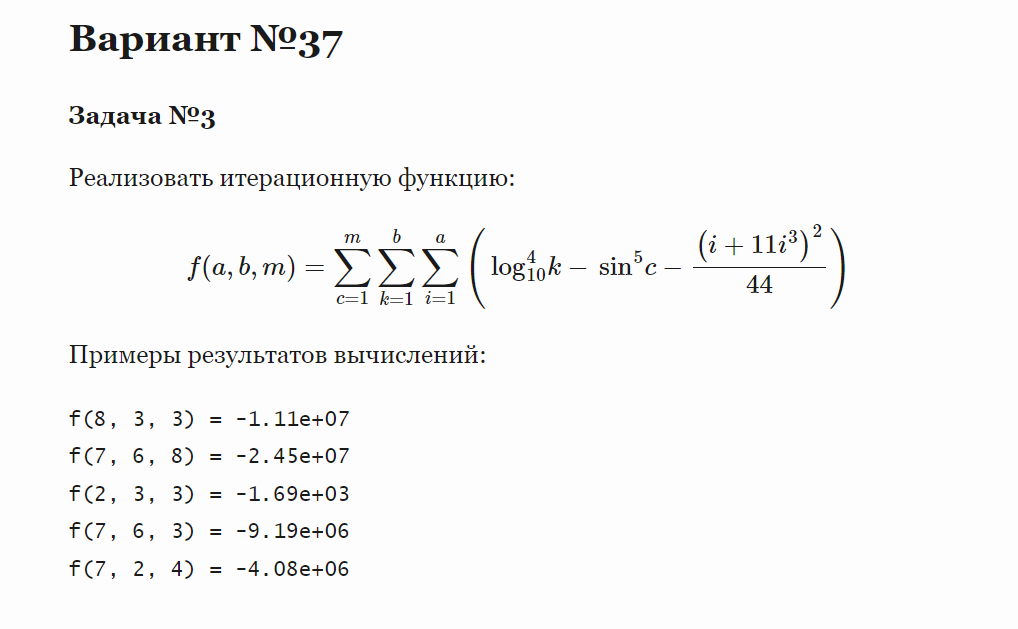
return (z\*\*4+4\*z\*\*3+z\*\*6)

### 

### Задача 1.3

#### Разобранный пример 1.

Знак сигма (Σ‎) - это сложение через цикл (+=), знак пи (П) - это произведение через цикл (\*=). Подряд идущие знаки пи и сигма - вложенные циклы.



from math import \*

def main(a, b, m):

f = 0

for c in range(1, m+1):

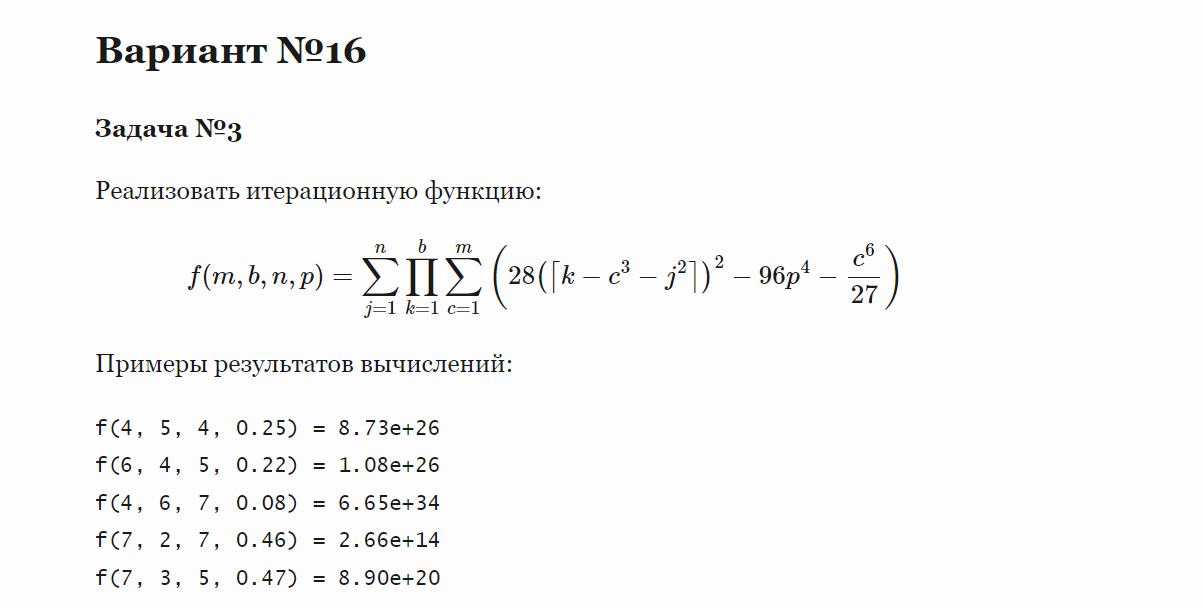
for k in range(1, b+1):

for i in range(1, a+1):

f += (log10(k)\*\*4) - sin(c)\*\*5 - (i + 11\*i\*\*3)\*\*2)/44

return f

#### Разобранный пример 2.



from math import \*

def main(m, b, n, p):

f = 0

for j in range(1, n+1):

f1 = 1

for k in range(1, b+1):

f2 = 0

for c in range(1, m+1):

f2 += 28\*(ceil(k-c\*\*3-j\*\*2))\*\*2-96\*p\*\*4-(c\*\*6)/27

f1 \*= f2

f += f1

return f

### 

### Задача 1.4

#### Разобранный пример.

Сделать формулы и потом рекурсивно вызываем main. Аналогично предыдущим задачам, только с рекурсией.

### 

def main(n):

if n == 0:

return -0.73

elif n == 1:

return -0.36

else:

return main(n-2)\*\*3-main(n-1)\*\*6

# элементарно

### 

### 

### Задача 1.5

#### Шаблон.

**Реализовать функцию, оперирующую векторами длины n:**

Импортируем math, в основном, только ceil, остальное по необходимости.

Берем длину какого-то вектора (массива) и записываем в n.

**ВАЖНО! у каждого индекса массива (вектора) нужно отнять -1**

def main(x, …):

n = len(x)

f = 0

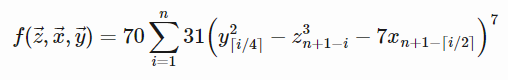
for i in range(1, n+1):

f += x[ceil(i/4) - 1] …

return f (Возможно умножение если перед суммой есть число)

#### Пример 1.

**ИВБО-07-20 Вариант 13**



from math import ceil

def main(z, x, y):

f = 0

n = len(z)

for i in range(1, n+1):

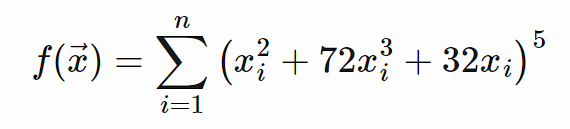
f += 31\*((y[ceil(i/4)-1])\*\*2-(z[n+1-i-1])\*\*3-7\*x[n+ 1-ceil(i/2)-1])\*\*7

return 70\*f

#### 

#### Пример 2.

**Вариант 5**



from math import \*

def main(n):

result = 0

for i in n:

result += pow((i \*\* 2 + 72 \* (i \*\* 3) + 32 \* i), 5)

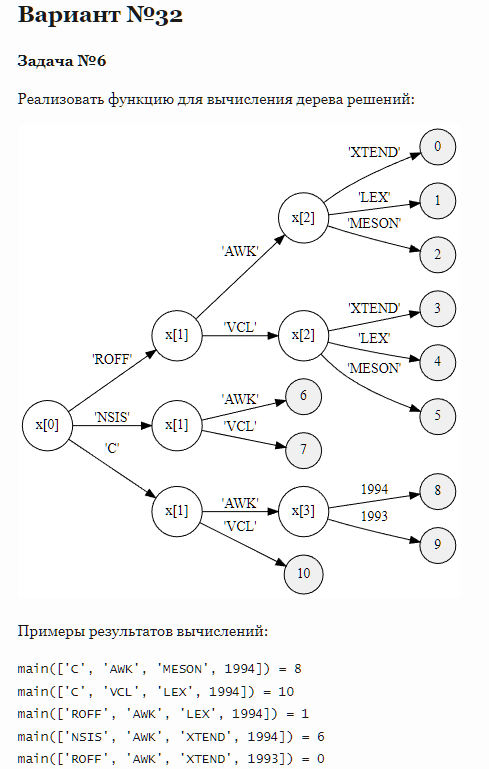
return result

## 

## Задача 2

### Задача 2.6

#### Разобранный пример.

  
Сначала мы проверяем нулевой элемент массива, потом первый и тд. Для каждой конкретной комбинации возвращаем конкретное значение. Рассмотрим 2 ветви дерева:

**Ветвь 1:**



Сначала проверяем нулевой элемент массива - он равен ‘C’, поэтому идем по нижней ветви дерева. Проверяем первый элемент - он равен ‘AWK’, проходим по верхней ветви. Таким образом доходим до возвращаемой восьмерки.

**Ветвь 2:**



После проверки 0 и 1 элементов, получаем возвращаемую 6, дальнейшие значения нас не интересуют.

def main(x):

match x[0]:

case "ROFF":

match x[1]:

case "AWK":

match x[2]:

case "XTEND": return 0

case "LEX": return 1

case "MESON": return 2

case "VCL":

match x[2]:

case "XTEND": return 3

case "LEX": return 4

case "MESON": return 5

case "NSIS":

match x[1]:

case "AWK": return 6

case "VCL": return 7

case "C":

match x[1]:

case "AWK":

match x[3]:

case 1994: return 8

case 1993: return 9

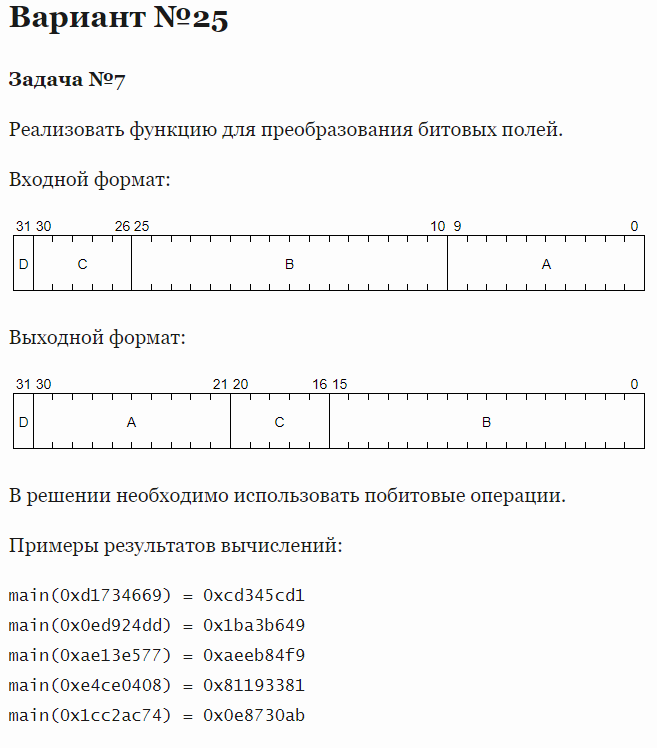
case "VCL": return 10

### 

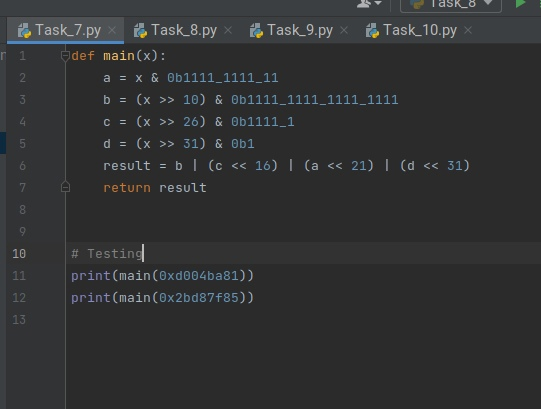
### Задача 2.7

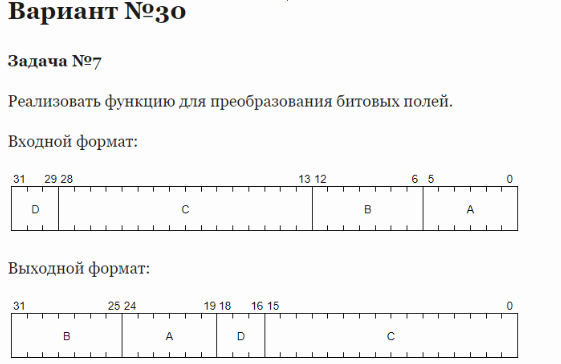
#### Вариант с двоичной маской.

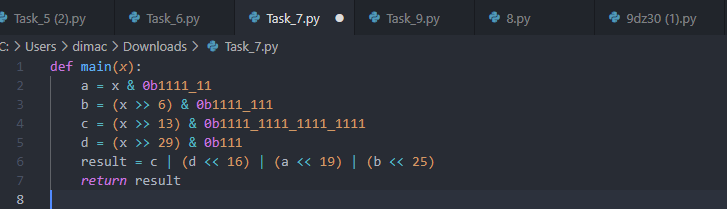
**ЦАП ИКБО-10-20**



Создаем 4 переменных (в данном случае 4): a, b, c, d. Затем смотрим на номер бита, с которого начинается блок во входном формате. Например, B начинается с 10-го бита, пишем это число в (x >> число). Потом применяем битовую маску, которая соответствует размеру поля. Например, поле D занимает 1 бит, поэтому маска 0b1, а поле A занимает 10 битов, поэтому маска 0b1111\_1111\_11. Как написать маску? Берем размер поля и пишем столько единиц, сколько составляет размер, пишем в начале 0b и после каждой четвертой единицы ставим \_. Затем смотрим на выходной формат и записываем в result в обратном порядке этот выходной формат. Опять смотрим на бит, с которого начинается блок в выходном формате и пишем в (переменная битового поля << число). Между сдвинутыми полями ставим | и возвращаем result.

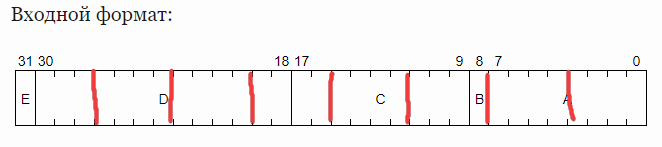






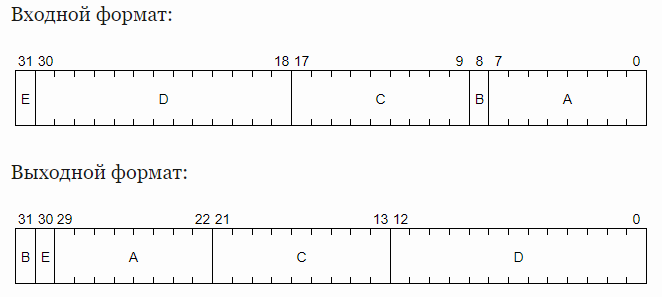
#### Вариант с 16-ой маской.

Делим верхнюю “линейку” по 4 и смотрим какую часть из 4 занимает определенная буква. Например в 13 варианте Е будет занимать 1000 из первой четверке, что равно 8 в двоичной, D занимает в 1 четверке остальные 0111, что равно 7 и так далее. Потом идет сдвиг.



**Красное это деление по 4 бита. Нужно для закомментированных строк кода ниже.**

**ИВБО-07-20 Вариант 13**



def main(x):

e = x & 0x80000000 #0b1000\_0000\_0000\_0000\_0000\_0000\_0000\_0000

d = x & 0x7ffc0000 #0b0111\_1111\_1111\_1100\_0000\_0000\_0000\_0000

c = x & 0x0003fe00 #0b0000\_0000\_0000\_0011\_1111\_1110\_0000\_0000

b = x & 0x00000100 #0b0000\_0000\_0000\_0000\_0000\_0001\_0000\_0000

a = x & 0x000000ff #0b0000\_0000\_0000\_0000\_0000\_0000\_1111\_1111

e = e >> 1

b = b << 23

d = d >> 18

c = c << 4

a = a << 22

return b+e+a+c+d

### 

### Задача 2.8

#### Разбор.

**.replace(‘заменяемый символ’, ‘символ на который надо заменить’)** - заменяет 1 символ на другой

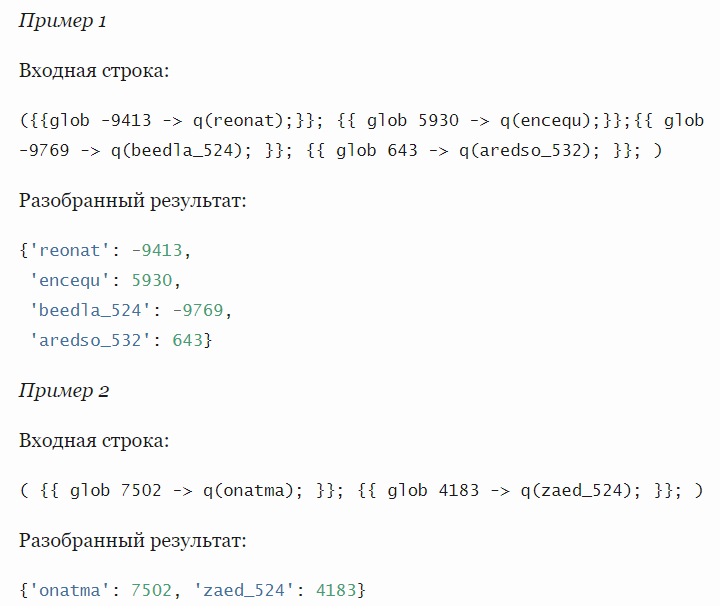
**.split(‘разделяющий символ’)** - создает массив и разделяет строку по определенным символам

**.find(‘символ(ы) который надо найти’)** - находит первое похожее сочетание

Создаем 2 массива, потом, в основном, заменяем \n на пустоту и разделяем строчку по определенным символам. Далее проходим по всем новым строкам. Ищим похожие слова или символы, которые есть в каждом примере и находим их позиции. Потом добавляем слова в два ранее созданных массива и пакуем их с помощью **dict(zip(то, что до двоеточия , то, что после двоеточия))**. Если нужно число, то, в данных случаях, в sl добавляем часть строки через метод **int().**

#### Пример 1.

**ИВБО-07-20 Вариант 3**



def main(x):

al = []

sl = []

a = x.replace(' ', '').replace('\n', '').split(';}};')

a = a[:-1]

for i in range(0, len(a)):

string = a[i]

pos1 = string.find('glob')+4

pos2 = string.find('q(')

sl.append(int(string[pos1:pos2-2]))

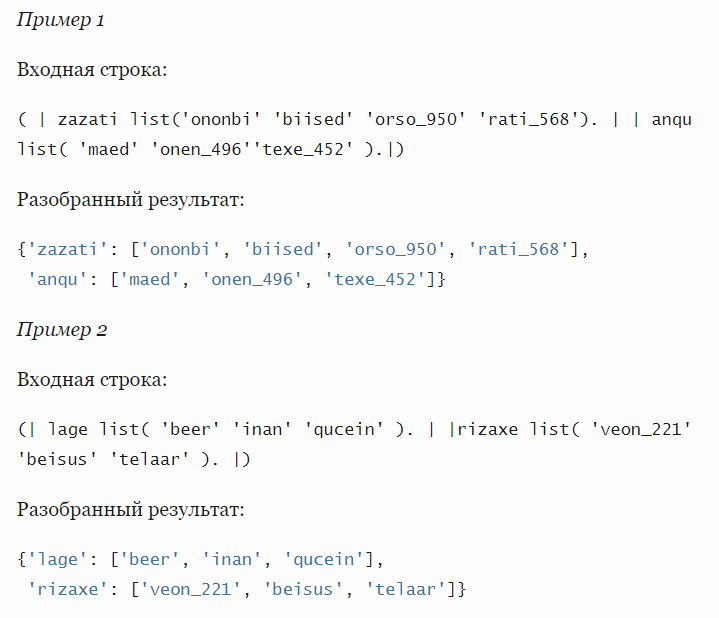
al.append(string[pos2+2:-1])

di = dict(zip(al, sl))

return di

#### Пример 2.

**ИВБО-07-20 Вариант 13**



def main(x):

al = []

sl = []

a = x.split('|')

a = a[1::2]

for i in range(0, len(a)):

string = a[i]

string = string.replace(" ", "")

string = string.replace('\n', '')

pos1 = string.find('list')

pos2 = string.find(').')

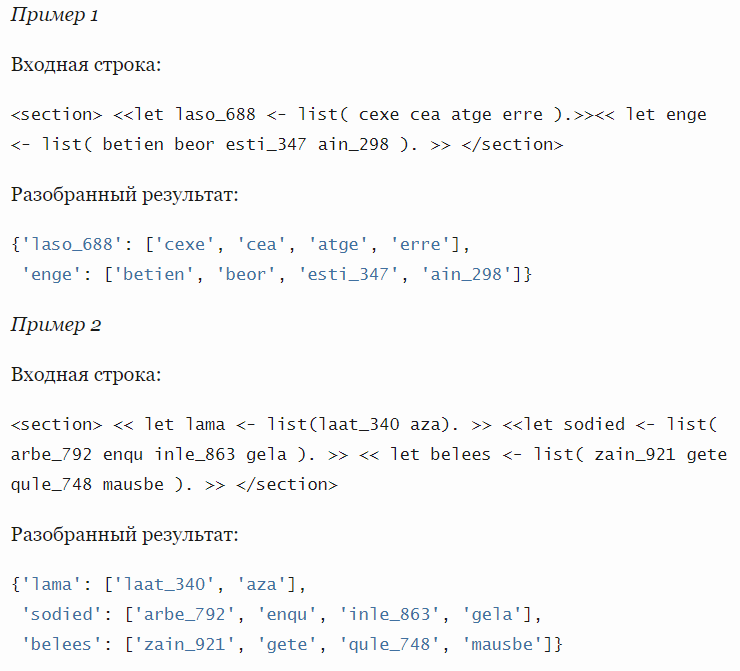
al.append(string[0:pos1])

sl.append(string[pos1+5:pos2].split("'")[1::2])

di = dict(zip(al, sl))

return di

#### Пример 3.



def main(x):

al = []

sl = []

a = x.split('>>')

a = a[:-1]

for i in range(0, len(a)):

string = a[i]

string = string.replace('(',' ').replace(')',' ')

string = string.replace('\n', ' ')

pos2 = string.find('list')

pos3 = string.find('.')

sl.append(string[pos2+4:pos3].split(" ")[1:-1])

string = string.replace(' ', '')

pos1 = string.find('let')+3

pos2 = string.find('list')

al.append(string[pos1:pos2-2])

sl = de(sl)

di = dict(zip(al, sl))

return di

def de(x):

for i in range(0, len(x)):

x1 = []

for j in range(0, len(x[i])):

if x[i][j] != '':

x1.append(x[i][j])

x[i] = x1

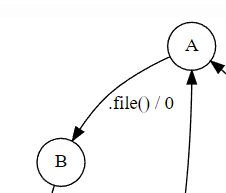
return x

### 

### Задача 2.9

#### Разбор.

Необходимо реализовать конечный автомат через класс. Для этого необходимо создать класс, имеющий некоторое количество различных методов. В данном примере только 2 уникальных метода, отвечающих за переход в другое состояние (mass и file). Записываем в переменную state первое состояние (‘A’). Далее расписываем все уникальные методы. Рассмотрим ниже:

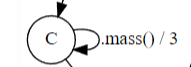


Рассмотри участок для 1 перехода: .file() это название метода, ‘A’ - это начальное состояние, ‘B’ - это конечное состояние, /0 - возвращаемое значение.

if self.state == 'A':

self.state = 'B'

return 0



Если состояние автомата не меняется, то просто возвращаем значение.

if self.state == 'C':

return 3

Если метод используется неоднократно, используем оператор elif.

Для каждого уникального метода в конце пишем это:

else:

raise(KeyError)

В конце пишем

def main():

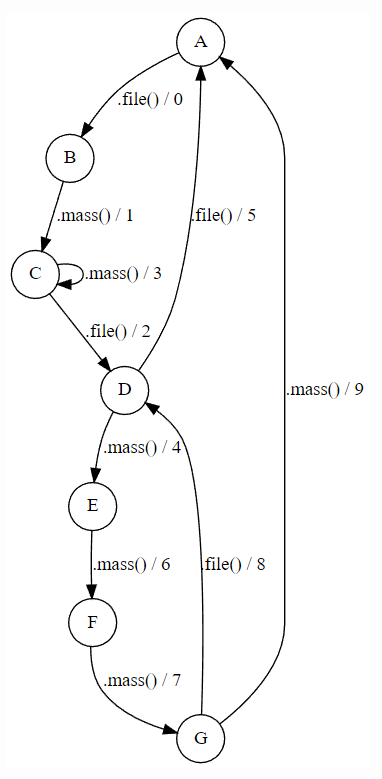
return StateMachine()

где StateMachine() это конструктор класса, описанного ранее

class StateMachine:

#### Пример.

**ЦАП ИВБО-08-20 Вариант 31**



class StateMachine:

state = 'A'

def file(self):

if self.state == 'A':

self.state = 'B'

return 0

elif self.state == 'D':

self.state = 'A'

return 5

elif self.state == 'G':

self.state = 'D'

return 8

elif self.state == 'C':

self.state = 'D'

return 2

else:

raise(KeyError)

def mass(self):

if self.state == 'B':

self.state = 'C'

return 1

elif self.state == 'C':

return 3

elif self.state == 'G':

self.state = 'A'

return 9

elif self.state == 'F':

self.state = 'G'

return 7

elif self.state == 'E':

self.state = 'F'

return 6

elif self.state == 'D':

self.state = 'E'

return 4

else:

raise(KeyError)

def main():

return StateMachine()