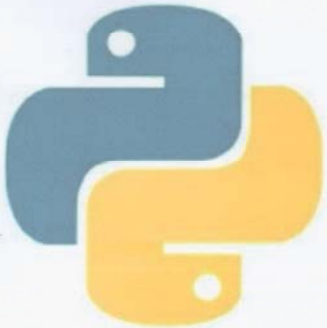
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»

Python. Множества

Методические указания к лабораторной работе

18



Рязань 2021

УДК 004.432

Python. Множества / Рязан. гос. радиотехн. универ.; сост.: А.В. Климухина, А.Н. Пылькин, Ю.С. Соколова, Е.С. Щенёв, М.Г. Щетинин. – Рязань, 2022 / Рязань: ИП Коняхин А.В. (Book Jet), 2022. – 22 с.

Рассмотрен один из разновидностей коллекций в Python тип множества (set), который выделяется в отдельную группу благодаря особенностям организации и обработки таких структур данных, относящихся к изменяемым типам. Рассмотрены различные аспекты математической теории множеств и особенности множеств в Python. Приведено описание основных средств работы с множественным типом данных, а также примеры, позволяющие продемонстрировать подходы решения практических задач по обработке больших хранилищ информации.

В качестве заданий для самостоятельного закрепления материала предлагается составить алгоритм и программу обработки данных в простейших массивах.

Предназначено для студентов направлений 09.03.03 – Прикладная информатика и 09.03.04 – Программная инженерия по дисциплине «Алгоритмические языки и программирование», а также для студентов очной и заочной формы обучения всех направлений подготовки и специальностей.

Табл. 1, Ил. 2.

Печатается по решению Научно-методического совета Рязанского государственного радиотехнического университета имени В.Ф. Уткина.

Рецензент: кафедра информатики, информационных технологий и защиты информации ФГБОУ ВО «Липецкий государственный педагогический университет им. П.П. Семенова-Тян-Шанского» (зав. каф., к.т.н., доц. Скуднев Д.М.).

Python. Множества

Составители: К л и м у х и н а Анастасия Витальевна

П ы л ь к и н Александр Николаевич

С о к о л о в а Юлия Сергеевна

Щ е н ё в Евгений Сергеевич

Щ е т и н и н Максим Геннадьевич

18. Организация данных в множестве

Определение множества в математической теории множеств

Множества являются основополагающими структурами в математике и программировании и нередко используются при проектировании алгоритмов и программ с целью повышения их эффективности.

В математике **множество** – это любой набор объектов, который понимается как единое целое. Объекты, принадлежащие множеству, называются *членами* или *элементами множества*. Множество определяется только своими элементами, элементы множеств не упорядочены, и два множества равны тогда и только тогда, когда они содержат одни и те же элементы. Например, следующие множества одинаковы: {1, 3, 5}, {5, 3, 1} и т.д.

Если все элементы одного множества являются также элементами другого, но не наоборот, то первое множество называется *подмножеством* второго, например множество {1, 3} являются подмножеством {1, 3, 5}. Множество всех подмножеств данного множества называется его *множеством-степенью*. Множество-степень {1, 3, 5} есть {{1, 3, 5}, {1, 3}, {1, 5}, {3, 5}, {1}, {3}, {5}, {}}. Так как каждый элемент исходного множества {1, 3, 5} или присутствует, или отсутствует в каждом из подмножеств, общее число подмножеств есть $2^3=8$.

Над множествами определены три бинарные операции, похожие на операции сложения, умножения и вычитания для целых чисел.

Объединение AUB (сумма) двух множеств A и B – это множество, элементы которого являются элементами либо A, либо B. Например, объединением множеств {1} и {3, 5} являются множество {1, 3, 5}.

Пересечение A∩B (произведение) двух множеств A и B – это множество, элементы которого являются элементами как A, так и B. Например, пересечением множеств {1, 3, 5} и {3, 5, 7} являются множество {3, 5}.

Дополнение B\A *(разность)* A и B – это множество, элементы которого являются элементами множества B, но не являются элементами множества A. Например, множество {1, 3} являются дополнением множества {5} до {1, 3, 5}.

Использование множеств в Python

В языке Python множество определяется как неупорядоченная коллекция значений без дублирующихся (повторяющихся) элементов. Множество может включать различные элементы, т.е. элементом множества может быть любой неизменяемый тип данных: числа, строки или кортежи. В тоже время изменяемые типы данных (список, словарь, другое множество) не могут быть элементами множества.

Объекты типа множество (set) обеспечивают выполнение классических математических операций с множествами: объединение, пересечение, разность и симметрическая разность.

Задание множества реализуется созданием его путем присвоения переменной последовательности значений в фигурных скобках, например, с помощью инструкции

m = {1, 2, 0, 1, 3, 2}.

При использовании метода

print(m)

будут введены значения

{0, 1, 2, 3}

Полученная последовательность будет уникальной, то есть без повторений, что подтверждает определение множества в виде не повторимой коллекции элементов.

Другой способ определения множества базируется на применении метода set. В качестве аргумента такой функции используется набор некоторых данных или строка текста. Строка программного кода

m = set ('РГРТУ')

приведет к заданию множества

{'Р', 'Г', 'Т', 'У'}.

Необходимо помнить, что пустое множество создается только с использованием set ():

m = set ()

Применение функции set() позволяет создать и передазать итерируемый объект (список, строку, кортеж) в качестве аргумента:

```
l = ['3', '4', '5'] # объявление списка l
s=set (l) # представление списка в множество s
print (type(s)) # выведется <class 'set'>
```

Основными способами использования множеств являются проверка на вхождение и устранение дублирующихся элементов. Чтобы проверить, входит некоторое значение в множество, используется операция in, например,

a = {2, 3, 4, 5}

print (5 in a)

приводит к выводу значения True.

С другой стороны, можно использовать операцию not in:

a={2, 3, 4, 5}

print (5 not in a)

В результате получается значение False.

Перебор всех элементов можно организовать с помощью оператора for, например, вывести все элементы множества можно с помощью цикла:

for a in {0, 1, 2, 3}:

print (a)

Задание множества можно с помощью генерации значений множества а с помощью цикла for для нескольких чисел:

a = {i for i in [1, 2, 0, 1, 3, 2]}

В результате сформируется множество

{0, 1, 2, 3} # повторяющиеся элементы убираются.

Изменение множеств

Число элементов во множестве может изменяться (путем добавления и удаления отдельных) с помощью специальных методов, определенных для типа set. Метод len позволяет определить точное число элементов, входящих в состав множества, например, строки

a={0, 1, 2, 3, 4}

print (len(a))

позволяют вывести значение числа элементов во множестве а, которое равно 5.

Приведем программный код, демонстрирующий возможные изменения множеств:

```
m = {'ноль','один','два','три'}
m.add('четыре') #добавляется элемент 'четыре'
print(m)
print('число элементов множества =', len(m))
#выведется {'четыре','один','два','ноль','три'}
#выведется "число элементов во вмножестве = 5"

m.remove('два') #удаляется елемент 'два'
print(m) #выведется
{'четыре','один','ноль','три'}
#при попытке удалить несуществующий элемент -
```

```
ОШИБКА!

m.discard('двадцать') # попытка удаления
несуществующего элемента
print(m) #выведется
{'четыре','один','ноль','три'}

m.discard('один') #удаление элемента 'один'
print(m) #выведется {'четыре','три','ноль'}

m.pop() #удаление первого элемента
print(m) #выведется {'три','ноль'}

m.clear() #полная очистка массива
print(m) # выведется set()
```

Программа состоит из шести частей, разделенных пустой строкой. В первой части с помощью метода add () вводит во множество еще элемент. Аргументом функции будет добавляемый аргумент ‘четыре’. Чтобы узнать точное число элементов множества используется метод len().

В последующих частях программы демонстрируются методы удаления элементов. Можно использовать три метода:

remove – удаление элемента с генерацией исключения в случае, если такого элемента нет;

discard – удаление элемента без генерации исключения, если элемент отсутствует;

pop – удаление первого элемента (при попытке удаления из пустого множества генерируется исключение).

Для удаления всех элементов (очистка множества) используется метод clear.

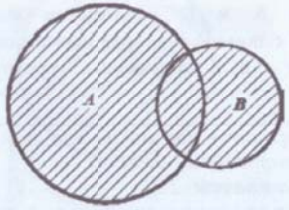
Операции над множествами

К переменным и константам множественного типа в Python применимы операции, которые обеспечивают одной строкой кода реализовывать сложные преобразования. В первую очередь к таким преобразованиям относятся типовые операции, определенные в математической теории множеств:

- объединение (логическое ИЛИ);
- пересечение (логическое И);
- разность;
- симмерическая разность.

Отметим тот факт, что реализация операцией над множествами (в том числе математические теоретико-множественные операции) в Python может быть осуществлена символьным способом или с помощью использования методов.

Объединением (суммой) двух множеств называется множество, состоящее из элементов, входящих хотя бы в одно из исходных множеств (на рисунке это заштрихованная область):

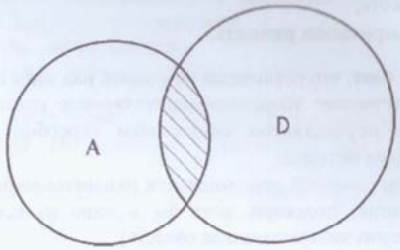



```
#математические операции над множествами
#(объединение, пересечение, разность,
симметричная разность,
#подмножество и надмножество)

A = {1, 2}
B = {3, 4}
C = A | B          #объединение множеств A и B
                   #символьный способ
print(C)           #выведется {1, 2, 3 ,4}
D = {2, 5, 6}
E = D.union(C)     #объединение множеств D и C
                   #использование метода
print(E)     # выведется {1, 2, 3 ,4, 5, 6}
```

В приведенном фрагменте программного кода реализуется объединение двух исходных множеств A = {1,2} и B = {3, 4} с помощью операции “|”. В итоге получается результирующее множество C={1, 2, 3, 4}. Далее операция объединения выполнена с помощью метода union (объединяются множества D и C).

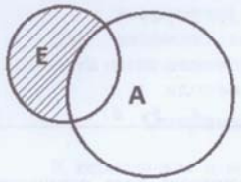
Пересечением (умножением) двух множеств называется множество элементов, общих для всех исходных множеств:



Пересечение множеств A и D символьным способом реализуется операцией "&" или с помощью метода intersection. Оба способа продемонстрированы в следующем продолжении программного кода:

```
F = A & D               # пересечение множеств A и D
                        # символьный способ
print(F)                # выведется {2}
F = A.intersection      # пересечение множеств A и D
                        # использование метода
print(F)                # выведется {2}
```

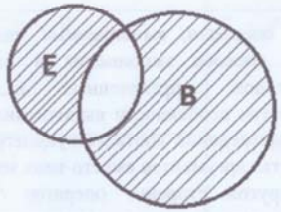
Разностью двух множеств называется третье множество, каждый элемент которого принадлежит уменьшаемому множеству и не принадлежит вычитаемому множеству:



В продолжении программного кода вычисляется разность двух множеств E и A двумя способами:


```
G = E - A               # разность множеств E и A
                        # символьный способ
G = E.difference(A)     # разность множеств E и A
                        # использование метода
print(G)    # выведется {2, 4, 5, 6}
```

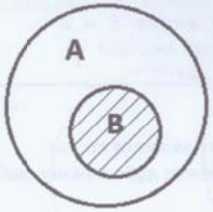
Симметрическая разность двух множеств является третье множество, каждый элемент которого принадлежит либо одному, либо другому множеству, но не их пересечению:



В продолжении программного кода демонстрируется выполнение симметрической разности множеств E и B двумя способами (символьным и с помощью метода):

```
S = E ^ B               # симметрическая разность
множеств E и B
                        # символьный способ
S = E.symmetric_difference(B)
                        # симметрическая разность
множеств E и B
                        # использование метода
print(S)    # выведется {1, 2, 5, 6}
```

В процессе обработки множественных объектов в ряде случаев возникает необходимость определить является одно множество частью (подмножеством) другого или на оборот – содержит одно множество другое в виде подмножество (если B = A, то B является подмножеством A):



B – подмножество множества A
A – надмножество по отношению к множеству B

В табл. 18.1 приведены все операции, выполняемые над множествами с учетом символьного способа выполнения и их реализации с применением методов, определенных над множественным типом. Существуют другие особенности выполнения операции с множествами. Например, применение соответствующего метода допускает использование в качестве аргументов вместо типа set других итерируемых объектов. С другой стороны, оператор ^ симметрической разности допускает использование нескольких наборов, а соответствующий метод symmetric_difference() не допускает подобного. Однако авторы считают, что подобные тонкости программист освоит при дальнейшим изучении и практическом использовании Python. Отметим лишь еще два факта. Возможно преобразовании строки во множество, а также списка во множество. Вторая особенность заключается в использовании структуры данных Frozenset, которая является множеством, но с неуказанными данными.

Примеры использования множеств в программах

Правила использования и работы со множествами рассмотрим на классических примерах, в которых применение типа set обеспечивает нужную обработку данных.

Пример. Проверить, является ли вводимая строка символов идентификатором.

Первой проблемой, которую требуется определить является формулирование привил написания идентификаторов, так как в разных реализациях языка программирования эти правила отличаются. Будем считать, что идентификатор состоит из строчных (маленьких) и

прописных (заглавных) латинских букв, арабских цифр и символа подчеркивания '_', при этом первым символом идентификатора не может быть цифра.

На рисунке 18.1 показана схема алгоритма проверки строки, является она идентификатором.

Таблица 18.1. Операции над множествами

Символьный способ выполнения операции	Выполнение операции с использованием метода	Результаты выполнения операции
C = A | B	`C = A.union(B)`	Возвращает множество C, которое является объединением множеств A и B
A |= B	`A.update(B)`	Добавляет в множество A все элементы из множества B
C = A & B	`C = A.intersection(B)`	Возвращает множество C, которое является пересечением множеств A и B
A &= B	`A.intersection_update(B)`	Оставляет в множества A только те элементы, которые есть в множестве B
C = A – B	`C = A.difference(B)`	Возвращает множество C, которое является разностью множеств A и B, т.е. элементы, входящие в A, но не входящие в B
A -= B	`A.difference_update(B)`	Удаляет из множества A все элементы, входящие в B

C = A Δ B	C = A.summetric_difference(B)	Возвращает множество C, которое является симметрической разностью множеств A и B, т.е. элементы, входящие в A и B, но не оба в них одновременно
A Δ= B	A.summetric_difference_update(B)	Записать в A симметрическую разность множеств A и B

Символьный способ выполнения операции	Выполнение операции с использованием метода	Результаты выполнения операции
d = A <= B	d=A.issubset(B)	Возвращает d = True, если множество A является подмножеством B. В противном случае d = False
d = A >= B	d=A.issuperset(B)	Возвращает d = True ,если A является надмножеством B. В противном случае d = False
d = A < B d = A > B		Аналогично действию A <= B и A >= B, но с учетом равенства множеств A и B

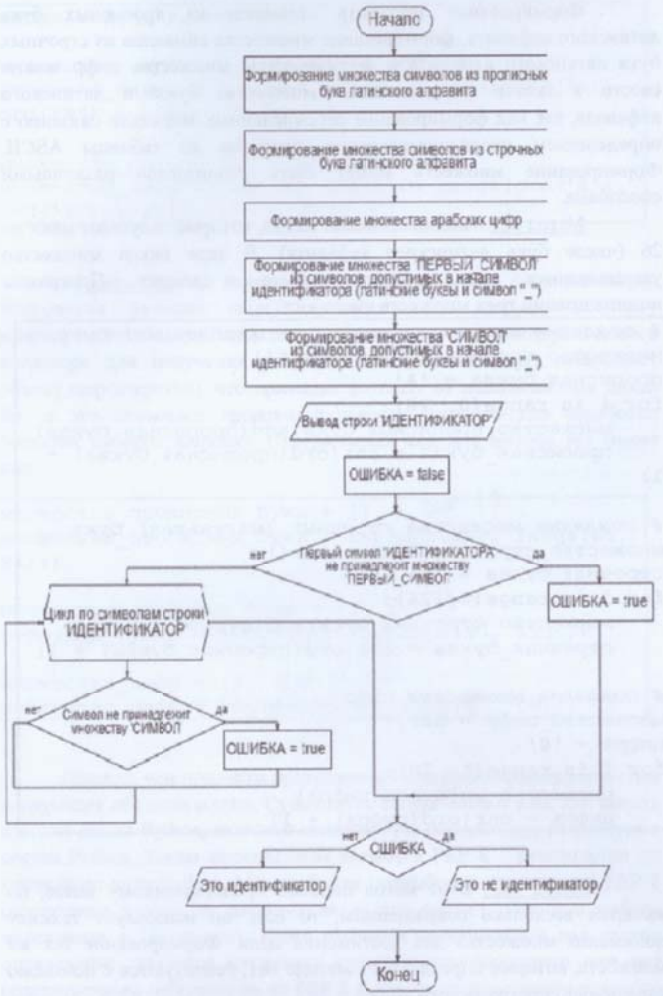


Рис. 18.1. Алгоритм определения, является строка идентификатором

Формирование множеств символов из прописных букв латинского алфавита, формирование множества символов из строчных букв латинского алфавита и формирование множества цифр можно свести к задачи инициализации множества буквами латинского алфавита, так как формирование перечисленных множеств связанно с определением последовательности символов из таблицы ASCII. Формирование множеств может быть реализовано различными способами.

Метод №1. Наиболее общий метод, который запускает цикл до 26 (число букв латинского алфавита). В теле цикла множество увеличиваются при добавлении букв в массив. Программа инициализации трех множеств имеет вид:

```
# создание множества прописных (заглавных) букв
множество_прописных_букв = set()
прописная_буква = 'A'
for i in range(0, 26):
    множество_прописных_букв.add(прописная_буква)
    прописная_буква = chr(ord(прописная_буква) +
1)

# создание множества строчных (маленьких) букв
множество_строчных_букв = set()
строчная_буква = 'a'
for i in range(0, 26):
    множество_строчных_букв.add(строчная_буква)
    строчная_буква = chr(ord(строчная_буква) + 1)

# создание множества цифр
множество_цифр = set()
цифра = '0'
for i in range(0, 10):
    множество_цифр.add(цифра)
    цифра = chr(ord(цифра) + 1)
```

Метод №2. Этот метод подобен представленному выше, но является несколько сокращенным, то есть он использует технику понимания множества для достижения цели. Формирование тех же множеств, которые определены в методе №1, реализуются с помощью следующего программного кода:

```
множество_прописных_букв = {}
множество_прописных_букв = {chr(x) for x in
range(ord('A'),ord('Z') + 1)}

множество_строчных_букв = {}
строчная_буква = {chr(x) for x in range(ord('a'),
ord('z') + 1)}

множество_цифр = {}
множество_цифр = {chr(x) for x in range(ord('0'),
ord('9') + 1)}
```


Метод №3. В данном методе используется функция map(). Встроенная функция map() применяется к каждому элементу итерируемого объекта (множества, списка и другое) и возврата нового итератора для получения результатов. Функция map() возвращает объект map(итератор), что приводит к отказу от использования цикла for, а это повышает производительность (полезно для обработки больших наборов данных). Программный код для метода №3 имеет вид:

```
множество_прописных_букв = {}
множество_прописных_букв = set(map(chr, range(65,
91)))

множество_строчных_букв = {}
строчная_буква = set(map(chr, range(97, 123)))

множество_цифр = {}
множество_цифр = set(map(chr, range(47, 58)))
```

Прежде, чем привести программный код всего примера, отметим следующее обстоятельство. Существует соглашение о том, как писать код для языка Python, включая стандартную библиотеку, входящую в состав Python. Таким соглашением являются PEP 8 – руководство по написанию кода Python. Мы не будем подробно рассматривать PEP 8 (достаточно просто эту информацию можно найти с помощью любого поисковика в Интернете). Ниже приведен программный код задачи определение, является введенная строка идентификатором или нет (единственным отступление от PEP 8 является использование русского алфавита при определении идентификаторов, вместо латинского):

```
# создание множества прописных (заглавных) букв
множество_прописных_букв = set()
прописная_буква = 'A'
for i in range(0, 26):
    множество_прописных_букв.add(прописная_буква)
    прописная_буква = chr(ord(прописная_буква) +
1)

# создание множества строчных (маленьких) букв
множество_строчных_букв = set()
строчная_буква = 'a'
for i in range(0, 26):
    множество_строчных_букв.add(строчная_буква)
    строчная_буква = chr(ord(строчная_буква) + 1)

# создание множества цифр
множество_цифр = set()
цифра = '0'
for i in range(0, 10):
    множество_цифр.add(цифра)
    цифра = chr(ord(цифра) + 1)
# создание множеств первого и последующих
символов
первый_символ = множество_прописных_букв |
                множество_строчных_букв |  {'_'}
символ = множество_прописных_букв |\
                множество_строчных_букв |  {'_'}
| множество_цифр

# ввод строки символов
идентификатор = str(input('введите строку
символов: '))
print('вы ввели идентификатор: ', идентификатор)

# основная программа
ошибка = False
if идентификатор[0] not in первый_символ:
    ошибка = True
else:
# k = len(идентификатор)
    for i in range(1, len(идентификатор)):
        if идентификатор[i] not in символ:
            ошибка = True
if ошибка:
```



```
    print('это не идентификатор')
else:
    print('это идентификатор')
```

Пример. Определить, из какого количества разных цифр состоит целые положительное числоп.

```
#              определение количества разных цифр
#              в записи целого положительного
числа
n = int(input('введите целое положительное число:
'))
s = n            # запоминание исходного числа
k = 0
M = set()
while n > 0:
    z = n % 10  # выделение последней цифры числа
    if z not in M:
        k += 1
        M.add(z)
    n = n // 10 # удаление последней цифры
print(" число разных цифр в записи числа %s = %i
" % (s, k))
```

Алгоритм решения поставленной задачи приведен на рис. 18.2. В рассматриваемом алгоритме первоначально создается пустое множество M, а затем на каждом шаге цикла с предусловием с помощью операции % определяется очередная младшая цифра числа. Если выделенная цифра не принадлежит множеству M, то она добавляются в это множество путем использования метода add(). Одновременно счетчик k подсчитывает количество таких включений. Затем с помощью операции // в числе отбрасывается его младшая цифра. Цикл продолжается до тех пор, пока анализируемое число содержит значащие цифры, то есть n > 0.

Программа на языке Python имеет довольно тривиальный вид:

Задание 9. В некотором районе находится пять продовольственных магазинов. В каждый их них завезли некоторые продукты из тех, что есть на базе: *хлеб, масло, молоко, мясо, рыбу, соль, сыр, сахар, чай, кофе, творог,* муку. Определите: какие продукты есть во всех магазинах; какие – хотя бы в одном; каких нет нигде.

Задание 10. Дано предложение, состоящее из латинских букв. Выведите все согласные буквы, которые входят хотя бы в одно слово.

Задание 11. Дано предложение, состоящее из латинских букв. Выведите все согласные буквы, которые входят только в одно слово.

Задание 12. Несколько городов одного региона обслуживает одно автотранспортное предприятие, наладившее между ними автобусные сообщения. Каждый автобус за один рейс заходит в некоторые из этих городов. Всего 5 – 10 рейсов. Составьте программу, которая отвечала бы на вопросы: в какие города можно добраться на автобусе за один рейс? Как можно добраться из города А в город В?

Задание 13. Задано предложение, состоящее из латинских букв. Выведите все гласные буквы, которые не входят более чем на одно слово.

Задание 14. Введите символьное множество. Разбейте его на три подмножества: цифры, буквы, специальные символы. Проверьте, есть ли среди них пустое множество.

Задание 15. В группе есть студенты, увлекающиеся спортом, и студенты, увлекающиеся искусством. Определите, множество студентов, которые увлекаются и спортом, и искусством.

Задание 16. В группе есть студенты, увлекающиеся спортом, и студенты, увлекающиеся искусством. Определите, множество студентов, которые не увлекаются ни спортом, ни искусством.

Задание 17. В группе есть студенты, увлекающиеся спортом, и студенты, увлекающиеся искусством. Определите, множество студентов, которые увлекаются либо спортом, либо искусством.

Задание 18. В заданном тексте из букв латинского алфавита определите общее число вхождений в него букв *a, e, c, h.*

Задание 19. Выведите в убывающем порядке все нечетные простые числа из допустимого диапазона.

Задание 20. Напишите программу, формирующую случайным образом множество целых чисел. Определите количество элементов в этом множестве и организуйте вывод элементов множества.

Задание 21. Произвольно заданы множества A, B, C и D. Определите, можно ли из элементов множеств A и C сформировать множество B, а из элементов множеств A, B, C – множество D?

Задание 22. Произвольно задано целое десятичное число. Определите, можно ли из цифр этого числа записать число 2468?

Задание 23. Произвольно задана номенклатура товаров в 4-х магазинах. Маршрут городского транспорта имеет остановки около 3-х магазинов. Определите, сможет ли покупатель приобрести необходимые для него товары, используя маршруты городского транспорта?

Задание 24. Произвольно задано множество товаров. Сформируйте два множества из съедобных и несъедобных товаров.

Задание 25. Определите, можно ли из букв произвольно заданных слов составить заданное слово?

Задание 26. В произвольно заданных двух строках определить количество гласных и согласных букв.

Задание 27. Определите количество десятичных цифр в произвольно заданной строке символов.

Задание 28. В двух произвольно заданных строках символов определите, в которой из них количество согласных букв больше?

Задание 29. В трех произвольно заданных строках символов определите, в которой из них количество гласных букв больше?

Задание 30. В произвольно заданной строке символов определите количество букв и количество арабских цифр.

Оглавление

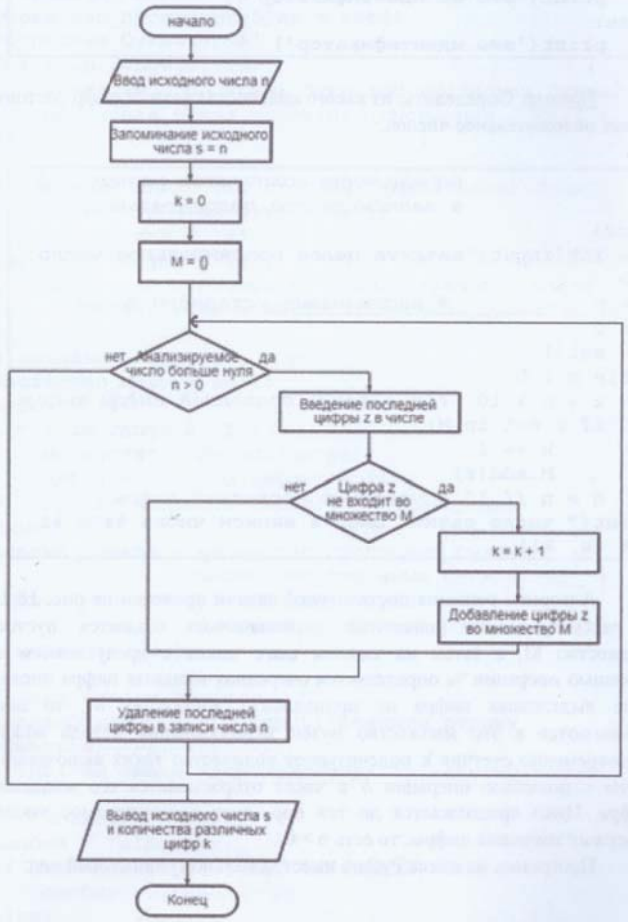


рис 18.2. Алгоритм определения количества разных цифр в записи целого числа

Контрольные вопросы

1. Что такое множество?
2. Каким образом задаются множества в Python?
3. Какое множество называется пустым и каким образом оно задается?
4. Каким образом добавляется элемент в существующее множество?
5. Как удалить элемент из множества?
6. Какие операции над множествами определены в Python?
7. Что такое симметрическая разность двух множеств?
8. Какие методы определенны в Python для выполнения операций над множествами?
9. Какие операции над множествами можно выполнять с использованием символьного способа?
10. Каким образом можно сформировать множество из букв алфавита?

Задания

Задание 1. Вычислите сумму тех элементов матрицы A, номера строк и столбцов которых принадлежат заданным множествам целых чисел S1 и S2.

Задание 2. Дан текст из цифр и латинских букв. Определите, каких букв – гласных *(a, e, i, o, u, y)* или согласных – больше в этом тексте.

Задание 3. Выведите в порядке возрастания все цифры, входящие в десятичную запись некоторого натурального числа *n*.

Задание 4. Выведите в порядке возрастания все цифры, не входящие в десятичную запись некоторого натурального числа *n*.

Задание 5. Дан текст, состоящий из латинских букв. Выведите все буквы, входящие в текст не менее двух раз.

Задание 6. Дан текст, состоящий из латинских букв. Выведите все буквы, входящие в текст ровно один раз.

Задание 7. Задано предложение, состоящее из латинских букв. Выведите все гласные буквы, входящее в это приложение.

Задание 8. Выведите в возрастающем порядке все целые числа из диапазона 1..1000, представимые в виде n^2+m^2, где n, m > 0.