Строки. Цифры и аримфметика

Основные функции для проверки строк и символов

1. Функция isdigit()

Проверяет, состоит ли строка полностью из цифр. Возвращает True, если все символы строки — цифры, и False в противном случае.

```
s = "12345"
print(s.isdigit())
```

Вывод >>> True

```
s = "12A45"
print(s.isdigit())
```

Вывод >>> False

2. Функция isalpha()

Проверяет, состоит ли строка только из букв

```
s = "ABC"
print(s.isalpha())
```

Вывод >>> True

```
s = "ABC123"
print(s.isalpha())
```

Вывод >>> False

3. Проверка на цифры с помощью in

Можно проверить, является ли символ цифрой, используя оператор in с набором символов "0123456789".

```
ch = '5'
if ch in "0123456789":
    print("Это цифра")
```

Работа с предопределенными наборами символов

1. Функция digits

Возвращает строку всех десятичных цифр: "0123456789"

```
from string import *
print(digits)
```

Вывод >>> 0123456789

2. Функция ascii_uppercase

Возвращает строку всех заглавных латинских букв: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

```
from string import *
print(ascii_uppercase)
```

Вывод >>> ABCDEF.....VWXYZ

Пример создания алфавита для работы с системами счисления

Соединив цифры от 0 до 9 и 15 первых заглавных букв латинского алфавита, мы получили алфавит 25-ричной системы счисления.

```
from string import *
alph = digits + ascii_uppercase[:15]
print(alph)
```

Вывод >>> 0123456789ABCDEFGHIJKLMNO

Пример задания

Текстовый файл состоит из символов, обозначающих заглавные буквы латинского алфавита и цифры от 1 до 9 включительно. Определите в прилагаемом файле максимальное количество идущих подряд символов, которые могут представлять запись числа в шестнадцатеричной системе счисления.
Примечание. Цифры, числовое значение которых превышает 9, обозначены латинскими буквами, начиная с буквы A.

Решение

1. Открываем файл и считываем его содержимое в строку s.
2. Определяем алфавит для шестнадцатеричной системы (цифры 0-9 и буквы A-F).
3. Проходимся по символам строки: если текущий и следующий символы принадлежат шестнадцатеричной системе, то их добавляем в буфер.
4. Когда встречается символ, не входящий в этот алфавит, буфер сохраняем как отдельную последовательность.
5. В конце находим и выводим длину самой длинной последовательности символов, подходящей для шестнадцатеричной системы.

Код программы:

```
from string import *
f = open("24_3.txt")
s = f.read()
buff = ""
ans = []
alph = digits + ascii_uppercase[:6]
for i in range (len(s) - 1):
  if s[i] in alph and s[i + 1] in alph:
    buff += s[i]
  else:
    buff += s[i]
    ans.append(buff)
    buff = ""
print(len(max(ans, key = len)))
```

Заметки