Строки

In [1]:

```
s = "Вася иногда учится"
print(s)
```

```
Вася иногда учится
```

In [2]:

```
s = input("What is your name: ") # ввод строки
```

```
What is your name: Name
```

In [4]:

```
print(s[3]) # можно обращаться к определнным элементам
```

```
e
```

Строки нельзя менять

Для изменения отдельных символов нужно создавать новую строку

In [6]:

```
s[1] = "a"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-5a6e7ae7b844> in <module>()
----> 1 s[1] = "a"

TypeError: 'str' object does not support item assignment
```

In [8]:

```
s = s[:1] + "b" + s[2:]
print(s)
```

```
Nbme
```

In [11]:

```
s = "Sakalof"
s1 = ""
for x in s:
    if x == "a":
        x = "o"
    s1 += x
print(s1)
```

```
Sokolof
```

Управляющие коды строковых констант

\n - перевод строки

\r - возврат каретки

\t - горизонтальная табуляция

\v - вертикальная табуляция

\a - звуковой сигнал

\b - возврат на одну позицию

\f - превод страницы

\0 - нулевой символ;не обозначает конец строки

\" - двойные кавычки

\' - апостроф

\\ - обратный слэш

\uhhhh - 16-битный сивол Unicode

\uhhhhhhhh - 32-битный символ юникод

Операции над строками

Индексация в строках начинается с нуля

В качестве индекса иожно указывать отрицательное число. В этом случае индекс отсчитывается от конца строки. Чтобы Получить положительный индекс, значанеие вычитается из длины строки. Последний символ: s[-1] или s[len(s)-1]

s1 + s2 - конкатенация

s*n - n-кратное повторение

min(s) - элемент с минимальным номером по таблице

max(s) - максимальный

Методы строк

s.center(n) - дополнение пробелами справа и слева

s.ljust(n) - выравнивание по левому краю

s.rjust(n) - выравнивание по правому краю

s.count(s1, [i, j]) - количество вхождений построки s1 в строку s

s.find(s1, [i, j]) - номер первого вхождения s1 в s

s.rfind(s1[, i, j]) - номер последнего вхождения

s.strip() - новая строка без пробелов в начале и конце

s.lstrip() - удаление пробелов в начале

s.rstrip() - удаление пробелов в конце

s.replace(s1, s2[, n]) - создается новая строка, в которой фрагмент s1 исходной строки заменяется на s2, n - количество замен

s.split([s1 [, n]]) - разделяет строку на подстроки по указанному разделителю s1 и добавляет эти строки в список, который возвращается в качестве результата. Если первый символ не указан, то в качестве разделителя используется пробел

Преобразование списка в стоку

Иногда возникает необходимость преобразовать список в сроку. для этого усипользуется метод join(). В этом случае в троску добавляется разделитель, указываемый в кавычках перед словом join()

In [18]:

```
mas = ["aaa", "bb", "ccc"]
st = " = ".join(mas)
print(st)
```

```
aaa = bb = ccc
```

Нельзя чтобы в исходном спике были отличные от строк сиволы, например, числа. Обойти это можно с помощью младующей конструкции

In [19]:

```
mas = ["aaaa", "bb", "ccc", 3, 5, 7]
st = " = ".join([str(x) for x in mas])
print(st)
```

```
aaaa = bb = ccc = 3 = 5 = 7
```

In [22]:

```
s = input()
word = 0
min_len = len(s)
for k in s:
    if "a" <= k <= "z":
        word += 1
        # print(word)
    else:
        if word < min_len and word != 0:
            min_len = word
        word = 0
print("Длина самого короткого слова:", min_len)
```

```
qqq
Длина самого короткого слова: 3
```

In [28]:

```
s = input("input string :")
num = int(input("number of the word: "))
l = len(s)
count = 0
k = 0
b = False
while k < l:
    if s[k] != " " and not b:
        count += 1
        b = True
        if count == num:
            break
    if s[k] == " ":
        b = False
    k += 1
print("Первая буква ", num, "-го слова : ", s[k], sep="")
```

```
input string :aaa bb c
number of the word: 3
Первая буква 3-го слова : c
```

In [1]:

```
s = input("strign of the words: ")
n = int(input("number of the words: "))
s = s.split()
print(s[n-1][0])
print(s)
```

```
strign of the words: 11 22 33
number of the words: 2
2
['11', '22', '33']
```

In [15]:

```
# Поиск целых чисел в строке (программа сохраняет их в отдельный список)
s = input("Input words and numbers: ")
s += " "
l = len(s)
print(l)
int_s = []
k = 1
while k < l:
    sp = ""
    while "0" <= s[k-1] <= "9" and k < l:
        sp += s[k-1]
        k += 1
    k += 1
    if sp != "":
```

```
    if sp != "":
        int_s.append(int(sp))
print("numbers")
print(int_s)
```

```
Input words and numbers: 111 222 333
12
numbers
[111, 222, 333]
111 222 333
```

In [20]:

```
# s = "  456  "
s1 = s.rstrip()
print(s1) # выведет строку s без пробелов справа

s = "11qqq2qqq"
print(s) # выведет строку s

s1= s.replace("qqq", "WWWW")
print(s1) #  11WWWW2WWWW

n = s1.rfind("W")
print(n) # 10

s2 = s1.center(30)
print(s2) # выведет строку длиной 30 с s1 в центре
print()

s3 = s2.rstrip()
print(s3, end="\n\n")

s4 = s3.rjust(30, "*")
print(s4)
```

```
11qqq2qqq
11qqq2qqq
11WWWW2WWWW
10
        11WWWW2WWWW

        11WWWW2WWWW

**********      11WWWW2WWWW
```

In [22]:

```
s = input()
#  удаление символов в начале и в конце строки
k = 0
while s[k] == " ": k += 1
s = s[k:]
k = len(s)
while s[k-1] == " ": k -= 1
s = s[:k]

# замена пробелов в строке на "**"
s1 = s[0]
k = 1
while k < len(s):
    if s[k] != " ":
        s1 += s[k]
    elif s[k-1] != " ":
        s1 += "**"
    k += 1
print(s1)
```

```
   aaa bbb ccc ddd aaa lll.
aaa**bbb**ccc**ddd**aaa**lll.
```

In [35]:

```
# пробелы в строке заменяются на *
s = input()
l = s.split()
s1 = ""
for k in l:
```

```
    s1 += k + "*"
s1 = s1[:-1]
print(s1)
```

```
alskdjlask.     laksjdlkajd. d.    alskdj
alskdjlask.*****laksjdlkajd.*d.****alskdj
```

In [36]:

```
a = [0, 2, 7, 9, 0, 1, 4, 0]
s = 0
a.reverse() # [0, 4, 1, 0, 9, 7, 2, 0]
b = a[a.index(0)+1:]
print(b) # [4, 1, 0, 9, 7, 2, 0]
b = b[:b.index(0)]
print(b) # [4, 1]
for i in b:
    print(i)
    s += i
print(s) # 5
```

```
[4, 1, 0, 9, 7, 2, 0]
[4, 1]
4
1
5
```

Кортежи

Кортежи очень похожи на списки, но обладают одной, но очень важной особенностью - изменять кортежи нельзя. Кортеж можно использовать для нахождения максимального или минимального значения и их индексов с помощью цикла. Но использовать их для сортировки нельзя.

Создавать кортежи можно с помозью функции tuple(). Эта функция позваляет преобразовать любую последовательность в кортеж

In [41]:

```
t = tuple() # пустой кортеж
print(t)
t = (1, 2, 3) # создание кортежа
t = (7,) # создание кортежа из одного элемента (запятая обязательна!!!)
print(t)
t = tuple([7, 8, 9]) # преобразование списка в кортеж (7, 8, 9)
print(t)
t = tuple("xyz") # получаем кортеж ("x", "y", "z")
print(t)
```

```
()
(7,)
(7, 8, 9)
('x', 'y', 'z')
```

При работе с кортежами можно применять конкатенацию(+), дублирование(*), получение среза[], in и not in для проверки на схождение

In [44]:

```
x = (7, 8, 9)
y = x + (4, 5, 6)
r = y[1:4]
print(x, y, r, sep="\n")
```

```
(7, 8, 9)
(7, 8, 9, 4, 5, 6)
(8, 9, 4)
```

Можно искользовать комбинированные операторы присваивания. Кортежи можно сравнивать, использую символы сравнения(<, <=, ==, !=, >=)

Также можно искользовать функции min, max, len, а также методы countи index. Напомним, что метож t.index(s[,i[,j]]) ищет индекс первого вхождния s в кортеж t в диапазоне от i до j, а count определяет количество вхождений значения аргумента

К отдельному элементу кортежа можно обращаться, задавая индекс в квадратных скобках после имени кортежа

Все значения кортежа можно присвоить отдельным элементам

In [45]:

```
colors = ("red", "green", "yellow", "blue")
a, b, c, d = colors
print(a, b, c, d)
```

```
red green yellow blue
```

Кортежи допускают вложения с любым уровнем вложенности

In [46]:

```
t = (-13, 3.4, ("Natalia", (6, "vasa"), "Olga:", "Masha"))
print(t[2][1][1][2])
```

```
s
```

In [48]:

```
a = (1, 2, 3)
b = (7, 8)
a += b
print(a) # (1, 2, 3, 7, 8)
c = (4, 5)
b += c
print(b) # (7, 8, 4, 5)
a = a[:1] + a[2:4]
print(a) # (1, 3, 7)
```

```
(1, 2, 3, 7, 8)
(7, 8, 4, 5)
(1, 3, 7)
```

In [53]:

```
mas = ["aaa", "bb", "cccc"]
st = " = ".join(mas)
print(st)

mas = ["aaa", "bb", "cccc", 3, 5, 7]
st = " = ".join((str(x) for x in mas))
print(st)

t = (-13, 3.4, ("Наталия", (6, "Вася"), "Ольга", "Маша"))
print(t[2][1][1][2]) # s
```

```
aaa = bb = cccc
aaa = bb = cccc = 3 = 5 = 7
c
```