Язык программирования Питон (Python)

Типы и структуры данных

Строки

Строки – последовательности символов, заключённые в кавычки (одиночные или двойные).

- элементы строки нумеруются начиная с нуля;
- одиночный символ в Python – строка из одного элемента;
- длина строки ограничена только доступным объёмом памяти;
- в тройных кавычках можно указывать “многострочные” строки;
- расположенные рядом две строковые константы автоматически объединяются в одну

Встроенные функции для работы со строками:

- преобразование числа в строку `str();`
- преобразование строки-последовательности цифр в число `int(), float();`
- узнать код символа `ord();`
- получить символ по его коду `chr().`

Примеры:

```
str(123)       ->   '123';          ord('s')     ->   115;
int('123')     ->   123;            chr(100)     ->   'd';
float('12.34') ->   12.34
```

Использование метода **format** для ***формирования строки*** на основе данных

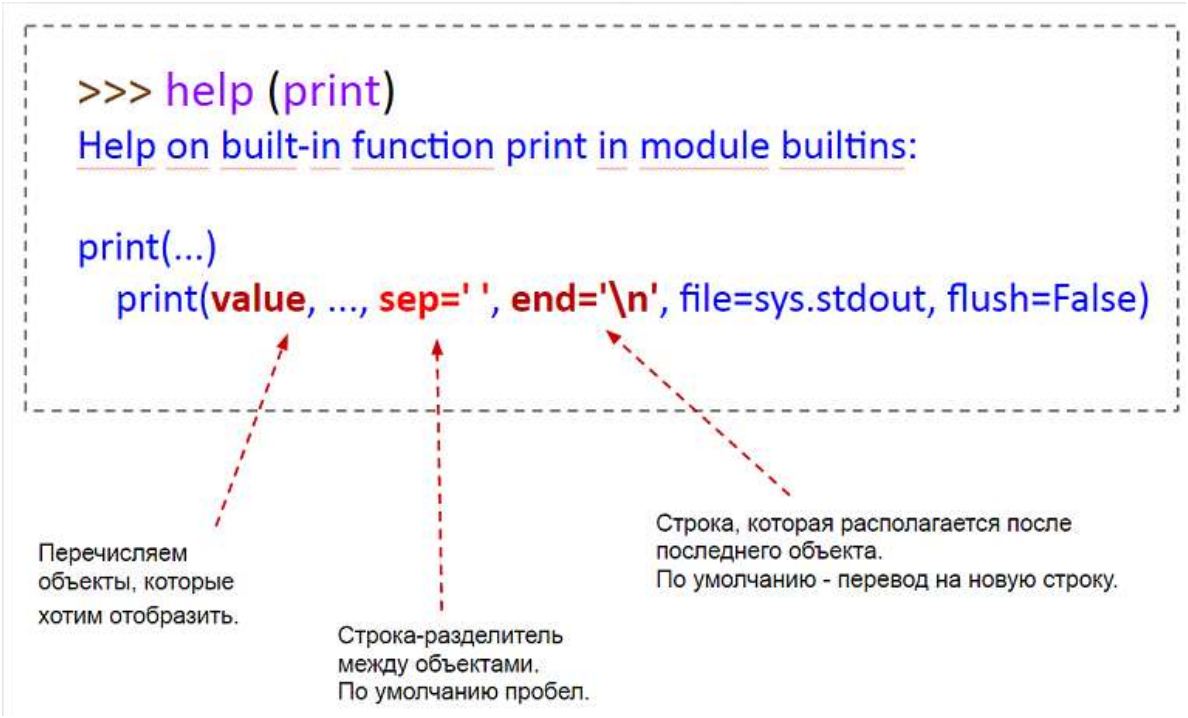
```
age = 26
name = 'Ivan'
rost = 1.76

print('')
print('Возраст {0} лет -- Имя {1} -- Рост {2} м.' .format(age, name, rost))
# Иначе
print('Возраст ' + str(age) + ' лет -- Имя ' + name +
      ' -- Рост ' + str(rost) + ' м.')
```

Определение ***“многострочной” строки***

```
s = ''' \n\n Это многострочная строка.
        Это её вторая строка. Можно использовать “кавычки”  '''
print(s)
```

Аргументы функции `print()`, *задаваемые по умолчанию*



Примеры:

```
print(1, 6, 7, 8, 9)            ->  1 6 7 8 9

print(1, 6, 7, 8, 9, sep=':')   ->  1:6:7:8:9
```

Основные операции со строками

Функция или операция	Описание
`len(s)`	Вычисляется длина строки `s` как число символов
`s1 + s2`	Конкатенация. К концу строки `s1` присоединяется строка `s2` *Пример:* `'вы' + 'года' -> 'выгода'`
`s*n (или n*s)`	`n`-кратное повторение строки `s` *Пример:* `'Ла' * 3 -> 'ЛаЛаЛа'`
`s[ i ]`	Выбор из s элемента с номером i, нумерация начинается с 0. Результатом является символ. Если i < 0, отсчёт идёт с конца (первый символ строки имеет номер 0, последний имеет номер −1). *Пример*: `s= 'дерево';  s[2] -> 'р';  s [-2] -> 'в'`
`s[i : j : k]`	***Срез*** – подстрока, содержащая символы строки `s` с номерами от `i` до `j` с шагом `k` – элемент с номером `i` входит в итоговую подстроку, а с номером `j` – не входит; – если `k` не указан (использован вариант `s[i:j]`), то символы идут подряд *Пример:* `s= 'дерево';  s[3:5] -> 'ев';  s[1:5:2] -> 'ее'`

Основные операции со строками *(продолжение)*

Функция или операция	Описание
`min(s)`	Определяет и выводит (возвращает) символ с наименьшим значением (кодом – номером в кодовой таблице) *Пример:* `s= 'derevo';  min(s) -> 'd'`
`max(s)`	Возвращает символ с наибольшим значением (кодом) *Пример:* `s= 'derevo';  max(s) -> 'v'`

Служебные символы:

`\n` - переход на новую строку
`\t` - знак табуляции
`\\` - наклонная черта влево
`\'` - символ одиночной кавычки
`\"` - символ двойной кавычки

Основные методы строк

Метод	Описание

`s.center(n)`	Возвращается строка `s`, дополненная пробелами справа и слева до ширины в `n` символов. Исходная строка не изменяется. Если **`n ≤ len(s)`**, пробелы не добавляются.
	Пример: `s='Zoom-Zoom';`
	`s.center(15)` -> `'␣␣␣Zoom-Zoom␣␣␣'`
	`str.center('Zoom-Zoom',15)` -> `'␣␣␣Zoom-Zoom␣␣␣'`

`s.ljust(n)`	Строка `s` выравнивается по левому краю (дополняется пробелами справа) в пространстве шириной `n` символов. Если **`n < len(s)`**, пробелы не добавляются. *Пример:* `s='Zoom-Zoom';` `s.ljust(15)` -> `'Zoom-Zoom␣␣␣␣␣␣␣'`
`s.rjust(n)`	Строка `s` выравнивается по правому краю (дополняется пробелами слева) в пространстве шириной `n` символов. Если **`n < len(s)`**, пробелы не добавляются. *Пример:* `s='Zoom-Zoom';` `s.rjust(15)` -> `'␣␣␣␣␣␣␣Zoom-Zoom'`

Основные методы строк (продолжение)

Метод	Описание

`s1.count(s[,i,j])`
 	
Квадратные скобки обозначают необязательные параметры
	Определяется количество вхождений подстроки `s` в строку `s1`. Результатом является число.
	Можно указать позицию начала поиска `i` и окончания поиска `j` (по тем же правилам, что и начало и конец среза).
	Пример: `s1 = 'abrakadabra'`
	`s1.count('ab') -> 2;` `s1.count('ab' ,1) -> 1`
	`s1.count('ab',1,-3) -> 0,`
	потому что `s1[1:-3] -> 'brakada'`

`s1.find(s[,i,j])`	Определяется позиция первого (считая слева) вхождения подстроки `s` в строку `s1`. Результатом является число. Необязательные аргументы `i` и `j` определяют начало и конец области поиска. Возвращает «`-1`» если подстрока не найдена. *Пример:* `s1 = 'abrakadabra';` `s1.find('br') -> 1`
`s1.rfind(s[,i,j])`	Определяется позиция последнего (считая слева) вхождения подстроки s в строку s1. Результатом является число. Необязательные аргументы i и j определяют начало и конец области поиска *Пример:* `s1 = 'abrakadabra';` `s1.rfind('br') -> 8`

Основные методы строк (продолжение)

Метод	Описание
`s.strip()`	Создаётся копия строки, в которой удалены пробелы в начале и в конце (если они есть или образовались в результате каких-то операций). *Пример:* `s='␣breKeKeKeKeKs␣'` `s1 = s.strip();  s1 -> 'breKeKeKeKeKs'`
`s.lstrip()`	Создаётся копия строки, в которой удалены пробелы в начале (если они есть или образовались в результате каких-то операций). *Пример:* `s='␣breKeKeKeKeKs␣';` `s.lstrip() -> 'breKeKeKeKeKs␣'`
`s.rstrip()`	Создаётся копия строки, в которой удалены пробелы в конце (если они есть или образовались в результате каких-то операций). *Пример:* `s='␣breKeKeKeKeKs␣'; s.rstrip() -> '␣breKeKeKeKeKs'`
`s.replace(s1, s2[,n])`	Создаётся новая строка, в которой фрагмент (подстрока) s1 исходной строки заменяется на фрагмент s2. Необязательный аргумент n указывает количество замен (если требуется заменить не все фрагменты). *Пример:* `s='breKeKeKeKeKs';` `s.replace('Ke','XoXo',2) -> 'breXoXoXoXoKeKs'`

Основные методы строк *(продолжение)*

Метод	Описание
`s.capitalize()`	Создаётся новая строка, в которой первая буква исходной строки становится заглавной (прописной), а все остальные становятся маленькими (строчными). *Пример:* `s='breKeKeKeKs';` `s.capitalize() -> 'Brekekekeks'`
`s.swapcase()`	Создаётся новая строка, в которой прописные буквы исходной строки заменяются на строчные и наоборот. *Пример:* `s='breKeKeKeKs';` `s.swapcase() -> 'BREkEkEkEkS'`
`s.upper()`	Создаётся новая строка, в которой все буквы исходной строки становятся заглавными (прописными). *Пример:* `s='breKeKeKeKs';` `s.upper() -> 'BREKEKEKEKS'`
`s.lower()`	Создаётся новая строка, в которой все буквы исходной строки становятся маленькими (строчными). *Пример:* `s='breKeKeKeKs';` `s.lower() -> 'brekekekeks'`

Основные методы строк *(продолжение)*

Метод	Описание
`s.startswith(s1)`	Проверяет, начинается ли строка `s` с подстроки `s1`. *Пример:* `s='breKeKeKeKs';` `s.startswith('br') -> True`

Примеры:

```
s='breKeKeKeKs'
if 're' in s:
    print('Да, строка s содержит строку "re"')
```

(см.3_3_списки, стр.9)

```
'ПРИВЕТ'.swapcase().endswith('т') -> True
```

аналогично `s.startswith(s1)`

Метод `format()`

– вместо **{0}** и **{1}** подставятся аргументы методы **format()**

```
'{0} и {1}'.format('труд', 'май')   -> 'труд и май'
'{1} и {0}'.format('труд', 'май')   ->  'май и труд'
```

– вывод числа **n** в двоичной системе

```
n = 10
'{:b}'.format(n)          ->  '1010'
```

– вывод в формате **Unicode**

```
'{:c}'.format(10)         ->  '\n'
```

– вывод числа в десятичной системе

```
'{:d}'.format(0xA)        ->  '10'
'{:d}'.format(0b0101)     ->  '5'
```

– вывод числа **n** в шестнадцатеричной системе

```
'{:x}'.format(255)        ->  'ff'
```