**СТРОКИ: String**

|  |  |  |
| --- | --- | --- |
| **Экранирование строк**  **(https://pythonworld.ru/tipy-dannyx-v-python/stroki-literaly-strok.html)** | | |
| **text="Privet, i am \"progrmist\""** | | Внутренние кавычки должны экрнироватся |
| **silly\_string = '''"Тут что-то не так,не будь я д'Артаньян", — подумал он.'''** | | Три одинарные кавычки позволяют использовать любой формт строки в том числе переносы через ентер. Многострочные блоки |
| **text1="Privet, \ni am \"progrmist\""** | | Перенос строки \n |
| **adress="c:\Programs\vscode"** | | Не правильный пример. Необходимо экранировать \ |
| **adress1="c:\\Programs\\vscode"** | | Правильный пример но не удобный |
| **adress2=r"c:\Programs\vscode"** | | Правильный и удобный пример экранирования адресов |
| **Индексирование строк. СРЕЗЫ** | | |
| **len(text)** | | Подсчет символов в строке |
| **text[0:5]** | | Выбор символов от 0 до 5 |
| **text[7:len(text)]** | | Выбор символов от 7 до конца. Символы можно вычетать |
| **text[::2]** | | Выбор символов с шагом |
| **text[::-1]** | | Выбор символов в обратном порядке |
| **Форматирование строк. Заполнители. (https://pythonworld.ru/osnovy/formatirovanie-strok-metod-format.html)** | | |
| **name="Dmitriy"**  **vozrast= 31**  **email= "dmitriyx@bk.ru"** | | | |
| **text = "Privet, menya zovut %s, mne %s let. Moy E-mail dlya svyazi:%s" %(name,vozrast,email)** | | | |
| **text1 = "Privet, menya zovut {1}, mne {2} let. Moy E-mail dlya svyazi:{0}".format(name,vozrast,email)** | | | |
| **text2 = "Privet, menya zovut {n}, mne {age:15.5f} let. Moy E-mail dlya svyazi:{mail}".format(n=name,age=vozrast,mail=email)** | | | |
| **text3 = f"Privet, menya zovut {name}, mne {vozrast} let. Moy E-mail dlya svyazi:{email}"** | | | |
| **“%(n)d %(x)s” % {“n”:1, “x”:”spam”} -**В данном случае (n) и (x) в строке формата ссылаются на ключи в словаре в пра­вой части выражения и служат для извлечения соответствующих им значе­ний. Этот прием часто используется в программах, создающих код разметки HTML или XML | | | |
| **Print(name % vozrast)**  **Print(name % email)** | | | |
| **reply = “Greetings...Hello %(name)s! Your age squared is %(age)s»** *# Шаблон с замещаемыми спецификаторами формата*  **values = {‘name’: ‘Bob’, ‘age’: 40}**  *# Подготовка фактических значений*  **>>> print reply % values**  *# Подстановка значений* | | | |
| **Таблица 7.4. Коды типов в форматировании строк**  **Код Описание**  s Строка (или строка str (X) любого объекта)  г То же, что и s, но применяется герг, а не str  с Символ (int или str)  d Десятичное число (целое число с основанием 10)  i Целое число  и То же, что и d (устарел: больше не является беззнаковым)  о Восьмеричное целое число (с основанием 8)  х Шестнадцатеричное целое число (с основанием 16)  х То же, что и х, но в верхнем регистре  е Число с плавающей точкой со степенью, в нижнем регистре  Е То же, что и е, но в верхнем регистре  f Десятичное число с плавающей точкой  F То же, что и f, но в верхнем регистре  g Число с плавающей точкой е или f  G Число с плавающей точкой Е или F  % Литерал % (записывается как %%) | | | |
| **СТРОКОВЫЕ ФУНКЦИИ** | | |
| **print(text.count("i"))** | Подсчет количества нужных элементов | |
| **print(text.capitalize())** | Возвращает 1-ую заглавную остальные маленькие буквы | |
| **print(text.upper())**  **text.lower()** | Перевод в верхний/нижний регистр | |
| **print(text.isupper())**  **text.islower()** | Проверка на все ли в верхнем/нижнем регистре | |
| **print(text.find("Dmitriy",5,30))** | Ищет нужный текст в строке и выводит номер символа | |
| **print("123qazwx".isalnum())** | Проверка Буквы + цифры = ТРУ | |
| **print("qazwsx".isalpha())** | Проверка Все буквы = ТРУ | |
| **print("123".isnumeric())**  **.isdecimal()** | Проверка Все цифры = ТРУ | |
| **print("  ".isspace())**  **"".isspace()** | Все символы пробелы =ТРУ | |
| **.istitle()** | Проврка Первый символ каждого слова заглавный а остальные строчные= ТРУ | |
|  |  | |
| **print("  1".strip(" "))**  **text.strip(" ")**  **“Оченьдлиннаястрока\_ьнечО”.strip(«ьнечО”) вернет «длиннаястрока\_»**  **text.lstrip()**  **text.rstrip()** | Удаляет сначала и конца пробелы и другие выбранные символы. **Порядок удаляемых символов не существенен**  **По умолчанию удаляет пробельные символы. Табуляция, новой строки и т.п.** | |
| **print(text.startswith("Privet"))**  **text.endswith("bk.ru")** | Проверка начало или конец строки = выбранный текст = ТРУ | |
| **print(text.split(" "))** | Возвращает строку типа LIST/Список и делит ее по РАЗДЕЛИТЕЛЮ | |
| **«, “.join([“slovo1”, “slovo2”])** | Вставляет нужный разделитель между элементами списка и возвращает строку | |
| **text.replace(‘pa’, ‘XYZ’)**  **‘SXYZm’**  **S.replace(‘SPAM’, ‘EGGS’, 1)**  **‘xxxxEGGSxxxxSPAMxxxx** | Замена одной подстроки другой  Исх.(Spam)  ***# Заменить одну подстроку*** | |
|  |  | |
| ВЫРАВНИВАНИЕ ТЕКСТА | | |
| .ljust()  .rjust()  .center()  «hello».rjust(20, “\*”)  “\*\*\*\*\*\*\*\*\*\*\*hello”  «hello».center(20, “=”)  “=======hello========” | Выравнивание текста, до длинны указанной в скобках. Добавляет пробелы или заполнитель с разных сторон | |
|  |  | |
| c= "ABC" "DEF"  c= "ABC" + "DEF" | **Конкатенация строк** | |
| **myjob = “hacker”**  **for c in myjob: print(c, end=’ ‘)**  h a c k e r | ***Обход элементов строки в цикле*** | |
| **‘spam’ in ‘abcspamdef’**  True | *Поиск подстроки, позиция не возвращается* | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |

*Таблица 7.2. Экранированные последовательности*

|  |  |
| --- | --- |
| **Последовательность** | **Назначение** |
| \*newline* | Игнорируется (продолжение на новой строке) |
| \\ | Сам символ обратного слеша (остается один символ \) |
| \’ | Апостроф (остается один символ ‘) |
| \” | Кавычка (остается один символ “) |
| \a | Звонок |
| \b | Забой |
| \f | Перевод формата |
| \n | Новая строка (перевод строки) |
| \r | Возврат каретки |
| \t | Горизонтальная табуляция |
| \v | Вертикальная табуляция |
| \*xhh* | Символ с шестнадцатеричным кодом *hh* (не более 2 цифр) |
| \*ooo* | Символ с восьмеричным кодом *ooo* (не более 3 цифр) |
| \0 | Символ Null (не признак конца строки) |
| \N{*id*} | Идентификатор ID базы данных Юникода |
| \*uhhhh* | 16-битный символ Юникода в шестнадцатеричном представлении |
| \U*hhhhhhhh* | 32-битный символ Юникода в шестнадцатеричном представленииa |
| \*другое* | Не является экранированной последовательностью (символ обратного слеша сохраняется) |

Таблица 7.3. Вызовы строковых методов в Python 3.3

S.capitalize()

S.1just(width[,fill])

S.casefold()

S.lower()

S.center(width[,fill])

S.lstrip([chars])

S.count(sub[,start[,end]])

S.maketrans (x [,y[,z] ] )

S.encode([encoding[,errors]])

S.partition(sep)

S.endswith(suffix[rstart[, end]])

S.replace(old,new[,count])

S.expandtabs([tabsize])

S.rfind(sub[,start[,end]])

S.find(sub[,start[,end]])

S.rindex(sub[,start[,end]])

S.rjust(width[,fill])

S.format(fmtstr,\*args,kwargs)

S.index(sub[,start[,end]])

S.rpartition(sep)

S.isalnum()

S.rsplit([sep[,maxsplit]])

S.rstrip([chars])

S.isalpha()

S.isdecimal()

S.split([sep[,maxsplit]])

S.splitlines([keepends])

S.isdigit()

S.isidentifier()

S.startswith(prefix [,start [,end]])

S.islower()

S.strip([chars])

S.swapcase()

S.isnumeric()

S.isprintable()

S.title()

S.translate(map)

S.isspace()

S.upper()

S.istitle()

S.zfill(width)

S.isupper()

S.join(iterable)