Занятие 2. Команда print()

1. Команда print()

Для вывода на экран используется функция print().
Внутри круглых скобок через запятую указывается то, что необходимо вывести на экран. Если это текст, обрамляем его кавычками - одинарными, или двойными. Главное, чтобы текст начинался и заканчивался кавычками одного типа. Команда **print** записывается только строчными буквами, другое написание недопустимо, так как в Python строчные и заглавные буквы различны. Так же не следует использовать пробелы перед открывающейся скобкой, так как это противоречит стандарту **PEP8**.

In [3]:
```
# Напишем простейшую программу. На экран будет выведена строка Hello, Python
print('Hello, Python!')
```

```
Hello, Python!
```

In [8]:
```
# Немного изменим программу:
print('Hello,', 'Python!')
# В данном случае мы внутри команды указали две строки. Каждую заключили в кавычки и
```

```
Hello, Python!
```

Обратите внимание: после запятой согласно стандарту **PEP8** обязательно нужно добавлять пробел.

2. Именованные аргументы print()

Из прошлого примера видно, что функция print при выводе разделяет аргументы пробелами, а в конце переходит на новую строчку.Но это не всегда удобно. Для тонкой настройки вывода у функции print существуют необязательные именованные аргументы **sep=''**, **end=''**

In [7]:
```
# С помощью команды help можно посмотреть информацию об интересующей функции:
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Таким образом, значения именованных аргументов функции print по умолчанию такие: print(..., sep=' ', end='\n').

sep — разделитель аргументов (по умолчанию пробел) и **end** — то, что выводится после вывода всех аргументов (по умолчанию символ начала новой строки). Например, если end сделать пустой строкой, то print не перейдет на новую строчку, и следующий print продолжит вывод прямо на этой же строке.

```
```

In [14]:
```
print('При', end='')
print('вет!')
```

```
Привет!
```

In [15]:
```
print('Раз', 'два', 'три', sep='--')
```

```
Раз--два--три
```

Обратите внимание: знак = здесь не выполняет никакого присваивания, переменных end и sep не появляется. Согласно стандарту **PEP8** пробелы вокруг знака = не используются

3. Экранирующая последовательность

Если внутри кавычек встречается символ \ — обратная косая черта, обратный слеш, бэкслеш, он вместе с идущим после него символом образует экранирующую последовательность (escape sequence) и воспринимается интерпретатором как единый специальный символ.

В частности, \n — символ начала новой строки. Кроме того, \t — табуляция, \' — кавычка, \ — просто бэкслеш. Использование экранирующих последовательностей вместо специальных символов называется их экранированием.

In [13]:
```
print('Произведение А.С. Пушкина "Евгений Онегин"')
print('Произведение А.С. Пушкина \'Евгений Онегин\'')
```

```
Произведение А.С. Пушкина "Евгений Онегин"
Произведение А.С. Пушкина 'Евгений Онегин'
```

In [20]:
```
print('восход\t04:04\nзакат\t20:53')
```

```
восход  04:04
закат   20:53
```

In [19]:
```
print('print(\'восход\\t04:04\\nзакат\\t20:53\')')
```

```
print('восход\t04:04\nзакат\t20:53')
```

4. "Сырые" (r-строки)

Если приписать букву r перед открывающей строку кавычкой, бэкслеши будут считаться обычными символами.

In [24]:
```
print("Файл на диске c:\\\\")
print(r"Файл на диске c:\\")
```

```
Файл на диске c:\\
Файл на диске c:\\
```

А если открывать и закрывать строку не одной, а тремя кавычками подряд, внутри можно делать обычные переводы строки (внутри одинарных кавычек так делать нельзя).

In [23]:
```
print("""Есть всего два типа языков программирования: те, на которые
люди всё время ругаются, и те, которые никто не использует.

Bjarne Stroustrup
""")
```

```
Есть всего два типа языков программирования: те, на которые
люди всё время ругаются, и те, которые никто не использует.

Bjarne Stroustrup
```