

Лабораторная работа 1 Введение в язык программирования Python

Цель работы: познакомиться со средой разработки Python. Изучить основные типы данных, команды ввода и вывода данных.

Краткая теория

Python— это объектно-ориентированный, интерпретируемый, переносимый язык сверхвысокого уровня. Программирование на Python позволяет получать быстро и качественно необходимые программные модули.

В комплекте вместе с интерпретатором Python идет IDLE (интегрированная среда разработки). По своей сути она подобна интерпретатору, запущенному в интерактивном режиме с расширенным набором возможностей (подсветка синтаксиса, просмотр объектов, отладка и т.п.).

Для запуска IDLE в Windows необходимо перейти в папку Python в меню “Пуск” и найти там ярлык с именем “IDLE (Python 3.X XX-bit)”.

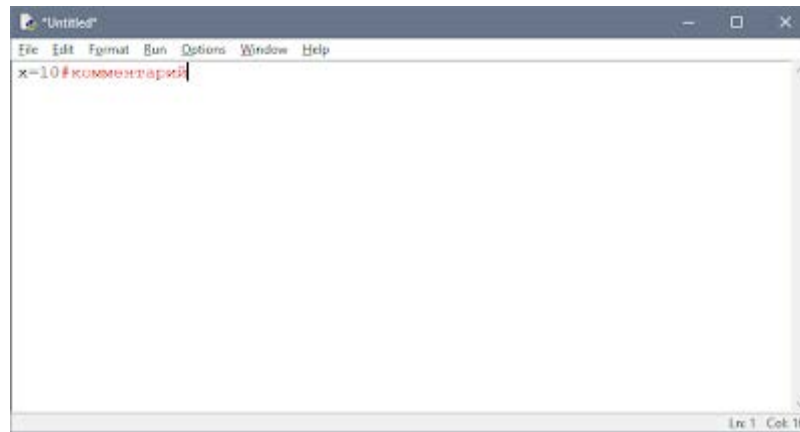
Для запуска редактора программы (кода) следует выполнить команду File->New File или сочетание клавиш Ctrl+N.

Любая Python-программа состоит из последовательности допустимых символов, записанных в определенном порядке и по определенным правилам.

Программа включает в себя:

- комментарии;
- команды;
- знаки пунктуации;
- идентификаторы;
- ключевые слова.

Комментарии в Python обозначаются предваряющим их символом # и продолжаются до конца строки (т.е. в Python все комментарии являются однострочными), при этом не допускается использование перед символом # кавычек:



Знаки пунктуации

В алфавит Python входит достаточное количество знаков пунктуации, которые используются для различных целей. Например, знаки "+" или "*" могут использоваться для сложения и умножения, а знак запятой "," - для разделения параметров функций.

Идентификаторы

Идентификаторы в Python это имена используемые для обозначения переменной, функции, класса, модуля или другого объекта.

Ключевые слова

Некоторые слова имеют в Python специальное назначение и представляют собой управляющие конструкции языка.

Ключевые слова в Python:

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Типы данных

1. None (неопределенное значение переменной)
2. Логические переменные (Boolean Type)
3. Числа (Numeric Type)
 1. int – целое число
 2. float – число с плавающей точкой
 3. complex – комплексное число
4. Списки (Sequence Type)
 1. list – список
 2. tuple – кортеж
 3. range – диапазон

5. Строки (Text Sequence *Type*)

1. str

Ввод и вывод данных

Ввод данных осуществляется при помощи команды **input**(список ввода):

```
a = input()
```

```
print(a)
```

В скобках функции можно указать сообщение - комментарий к вводимым данным:

```
a = input ("Введите количество: ")
```

Команда `input()` по умолчанию воспринимает входные данные как строку символов. Поэтому, чтобы ввести целочисленное значение, следует указать тип данных `int()`:

```
a = int (input())
```

Для ввода вещественных чисел применяется команда

```
a=float(input())
```

Вывод данных осуществляется при помощи команды **print**(список вывода):

```
a = 1
```

```
b = 2
```

```
print(a)
```

```
print(a + b)
```

```
print('сумма = ', a + b)
```

Существует возможность записи команд в одну строку, разделяя их через `;`. Однако не следует часто использовать такой способ, это снижает удобочитаемость:

```
a = 1; b = 2; print(a)
```

```
print (a + b)
```

```
print ('сумма = ', a + b)
```

Для команды **print** может задаваться так называемый сепаратор — разделитель между элементами вывода:

```
x=2
```

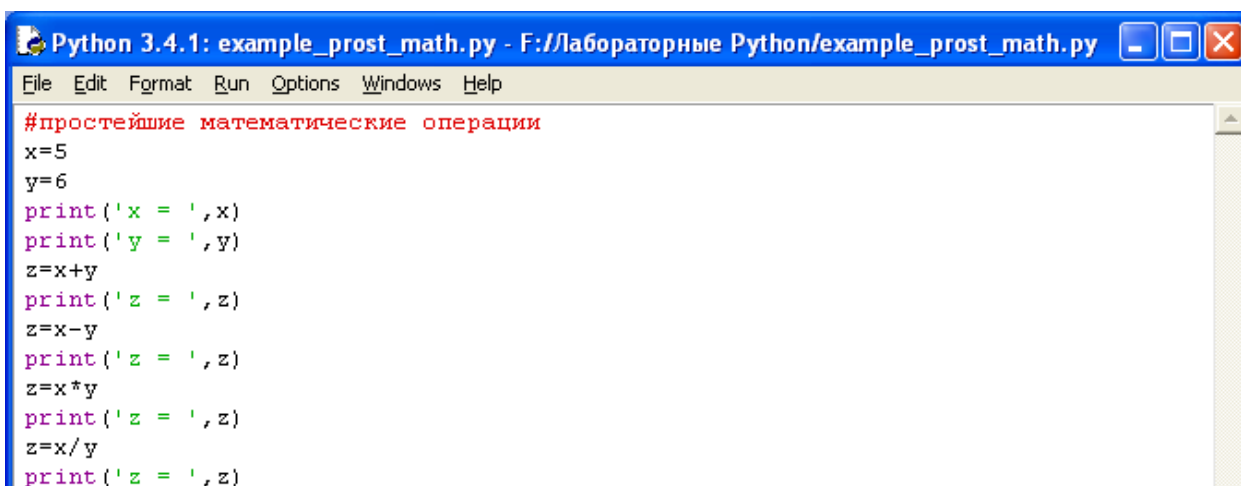
```
y=5
```

```
print ( x, "+", y, "=", x+y, sep = " " )
```

Результат отобразится с пробелами между элементами: $2 + 5 = 7$

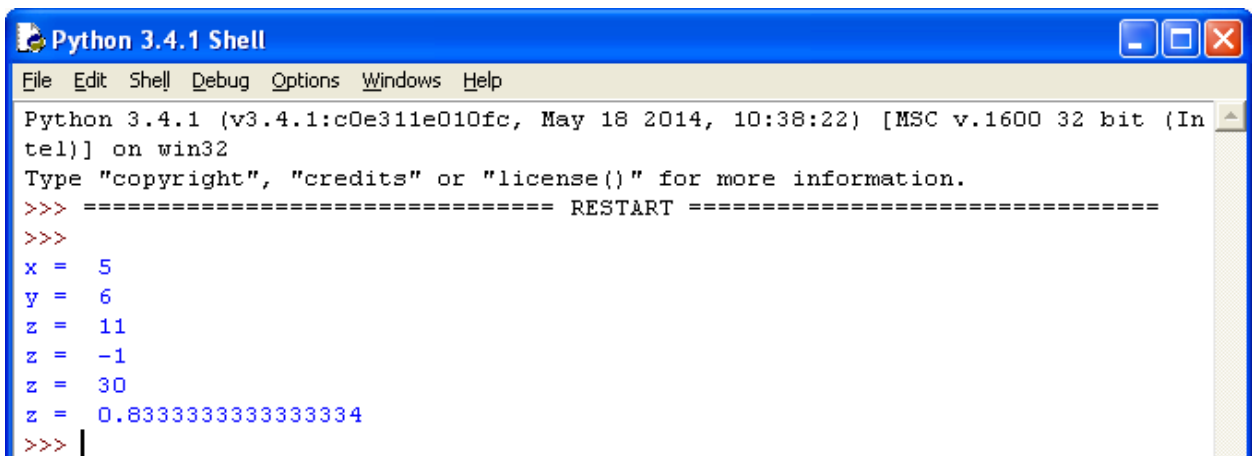
Простые арифметические операции над числами

$x + y$	Сложение
$x - y$	Вычитание
$x * y$	Умножение
x / y	Деление



```
Python 3.4.1: example_prost_math.py - F://Лабораторные Python/example_prost_math.py
File Edit Format Run Options Windows Help
#простейшие математические операции
x=5
y=6
print('x = ', x)
print('y = ', y)
z=x+y
print('z = ', z)
z=x-y
print('z = ', z)
z=x*y
print('z = ', z)
z=x/y
print('z = ', z)
```

Пример программы на Python



```
Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
x = 5
y = 6
z = 11
z = -1
z = 30
z = 0.8333333333333334
>>> |
```

Результат выполнения программы с применением простых арифметических операций

Для форматированного вывода используется **format**:

Строковый метод `format()` возвращает отформатированную версию строки, заменяя идентификаторы в фигурных скобках `{}`. Идентификаторы могут быть позиционными, числовыми индексами, ключами словарей, именами переменных.

Синтаксис команды **format**:

поле замены := `"{" [имя поля] ["!" преобразование] [":" спецификация] "}"`

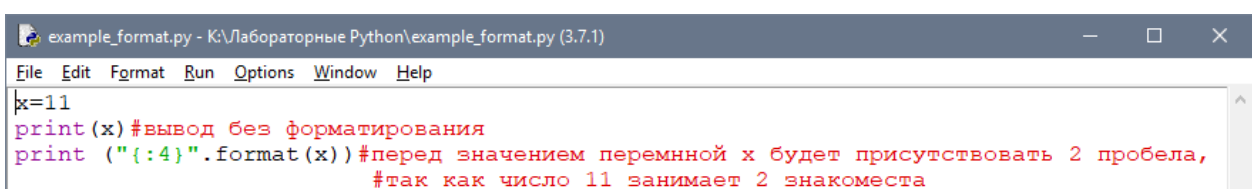
имя поля := `arg_name ("." имя атрибута | "[" индекс "]")*`

преобразование := `"r"` (внутреннее представление) | `"s"` (человеческое представление)

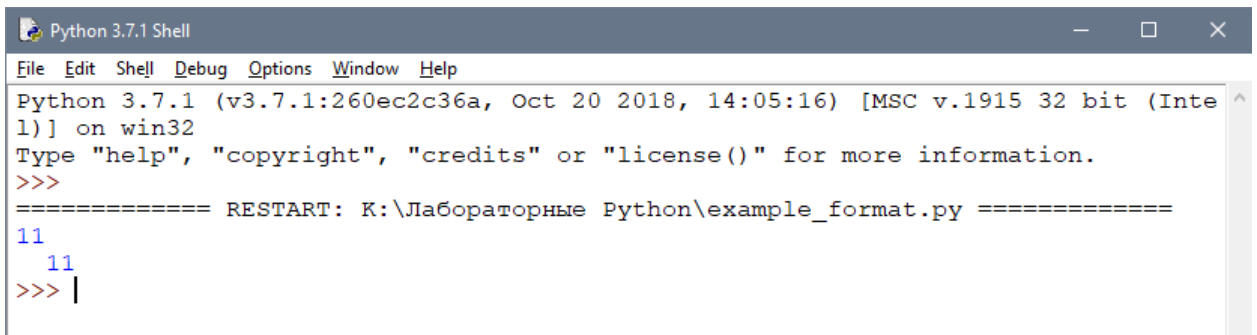
спецификация := см. ниже

Аргументов в `format()` может быть больше, чем идентификаторов в строке. В таком случае оставшиеся игнорируются.

Идентификаторы могут быть либо индексами аргументов, либо ключами:



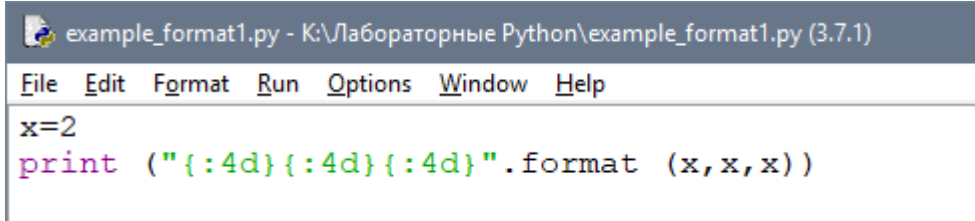
```
example_format.py - K:\Лабораторные Python\example_format.py (3.7.1)
File Edit Format Run Options Window Help
x=11
print(x) #вывод без форматирования
print ("{:4}".format(x)) #перед значением переменной x будет присутствовать 2 пробела,
                        #так как число 11 занимает 2 знакоместа
```



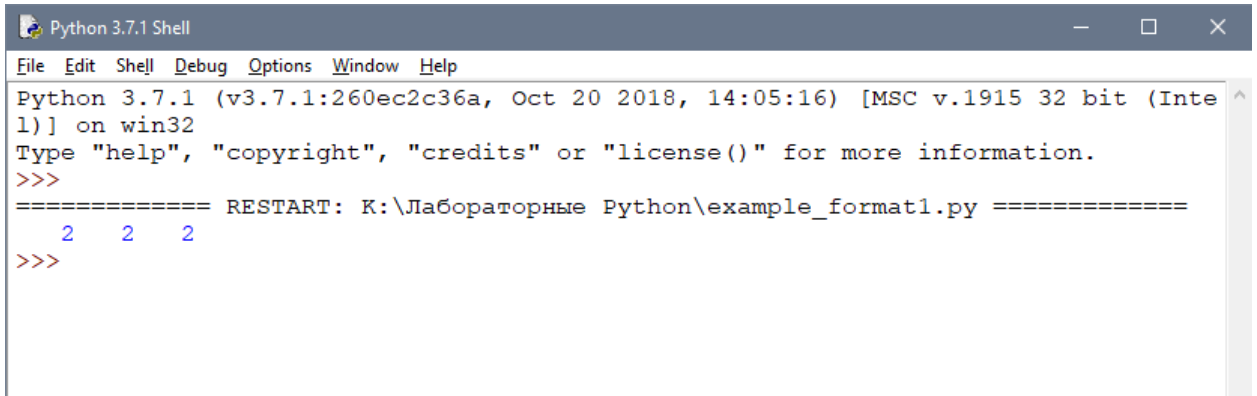
```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: К:\Лабораторные Python\example_format.py =====
11
11
>>> |
```

В результате выведется число 11, а перед ним два пробела, так как указано использовать для вывода четыре знакоместа.

Или с несколькими аргументами:



```
example_format1.py - К:\Лабораторные Python\example_format1.py (3.7.1)
File Edit Format Run Options Window Help
x=2
print ("{:4d}{:4d}{:4d}".format (x,x,x))
```



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: К:\Лабораторные Python\example_format1.py =====
2 2 2
>>>
```

В итоге каждое из значений выводится из расчета 4 знакоместа.

Спецификация формата:

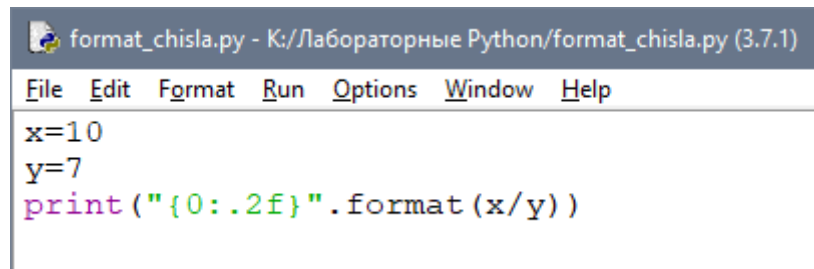
спецификация	:= [[fill]align][sign][#][0][width][,][.precision][type]
заполнитель	:= символ кроме '{' или '}'
выравнивание	:= "<" ">" "=" "^"
знак	:= "+" "-" " "
ширина	:= integer

точность	<code>:= integer</code>
тип	<code>:= "b" "c" "d" "e" "E" "f" "F" "g" "G" "n" "o" "s" "x" "X" "%"</code>

Тип	Значение
'd', 'i', 'u'	Десятичное число.
'o'	Число в восьмеричной системе счисления.
'x'	Число в шестнадцатеричной системе счисления (буквы в нижнем регистре).
'X'	Число в шестнадцатеричной системе счисления (буквы в верхнем регистре).
'e'	Число с плавающей точкой с экспонентой (экспонента в нижнем регистре).
'E'	Число с плавающей точкой с экспонентой (экспонента в верхнем регистре).
'f', 'F'	Число с плавающей точкой (обычный формат).
'g'	Число с плавающей точкой. с экспонентой (экспонента в нижнем регистре), если она меньше, чем -4 или точности, иначе обычный формат.
'G'	Число с плавающей точкой. с экспонентой (экспонента в верхнем регистре), если она меньше, чем -4 или точности, иначе обычный формат.
'c'	Символ (строка из одного символа или число - код символа).
's'	Строка.
'%'	Число умножается на 100, отображается число с плавающей точкой, а за ним знак %.

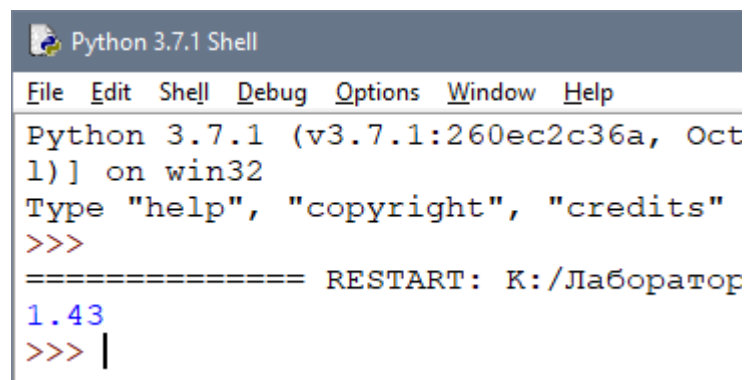
Для форматирования вещественных чисел с плавающей точкой используется следующая команда:

```
print('{0:.2f}'.format(вещественное число))
```



```
format_chisla.py - K:/Лабораторные Python/format_chisla.py (3.7.1)
File Edit Format Run Options Window Help
x=10
y=7
print("{0:.2f}".format(x/y))
```

В результате выведется число с двумя знаками после запятой.



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 1) on win32
Type "help", "copyright", "credits"
>>>
===== RESTART: K:/Лаборатор
1.43
>>> |
```

Пример

Напишите программу, которая запрашивала бы у пользователя:

- ФИО ("Ваши фамилия, имя, отчество?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где вы живете?")

После этого выводила бы три строки:

"Ваше имя"

"Ваш возраст"

"Вы живете в"

Решение


```
a=input('Введите ваши фамилию, имя, отчество ')\nb=input('Сколько вам лет? ')\nc=input('Где вы живёте? ')\nprint('Ваше имя ',a)\nprint('Ваш возраст ',b)\nprint('Вы живете в ',c)|
```

```
Введите ваши фамилию, имя, отчество Иванов Иван Иванович  
Сколько вам лет? 15  
Где вы живёте? Уссурийск  
Ваше имя  Иванов Иван Иванович  
Ваш возраст  15  
Вы живете в  Уссурийск
```

Задания для самостоятельной работы

1) Установите Python <https://www.python.org/ftp/python/3.8.5/python-3.8.5.exe>

2) Напишите программу, которая запрашивала бы у пользователя:

Имя, Фамилия, Возраст, Место жительства

- фамилия, имя ("Ваши фамилия, имя?")

- возраст ("Сколько Вам лет?")

- место жительства ("Где вы живете?")

После этого выводила бы три строки:

"Ваши фамилия, имя"

"Ваш возраст"

"Вы живете в"