Программирование на Python

Владимир Юрьевич Полищук

Инженерная школа
информационных технологий и
робототехники,
Томский политехнический университет

pvy3@tpu.ru

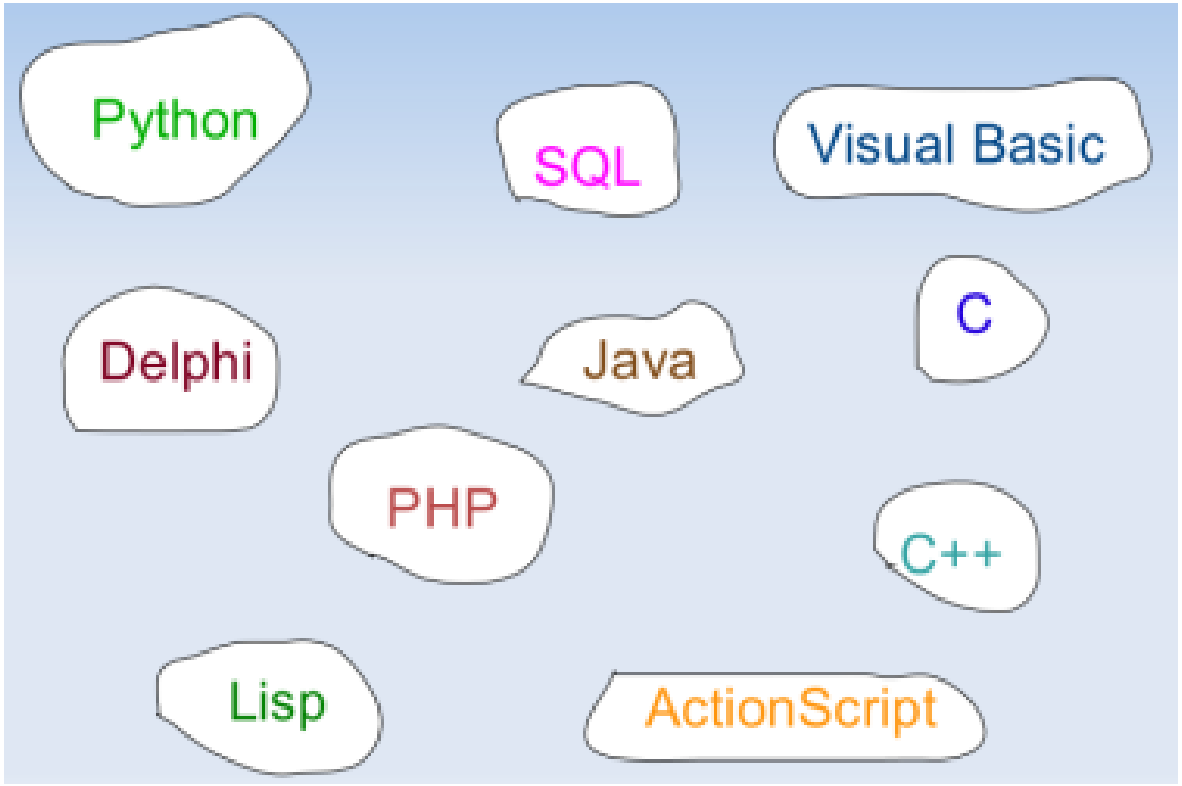
Содержание

- Краткая история. Что такое программа, языки программирования.
- Основные этапы развития языков программирования. Разнообразие языков программирования, трансляция.
- Знакомство с python. Краткая история, особенности языка.
- Типы данных. Переменные
- Ввод и вывод данных

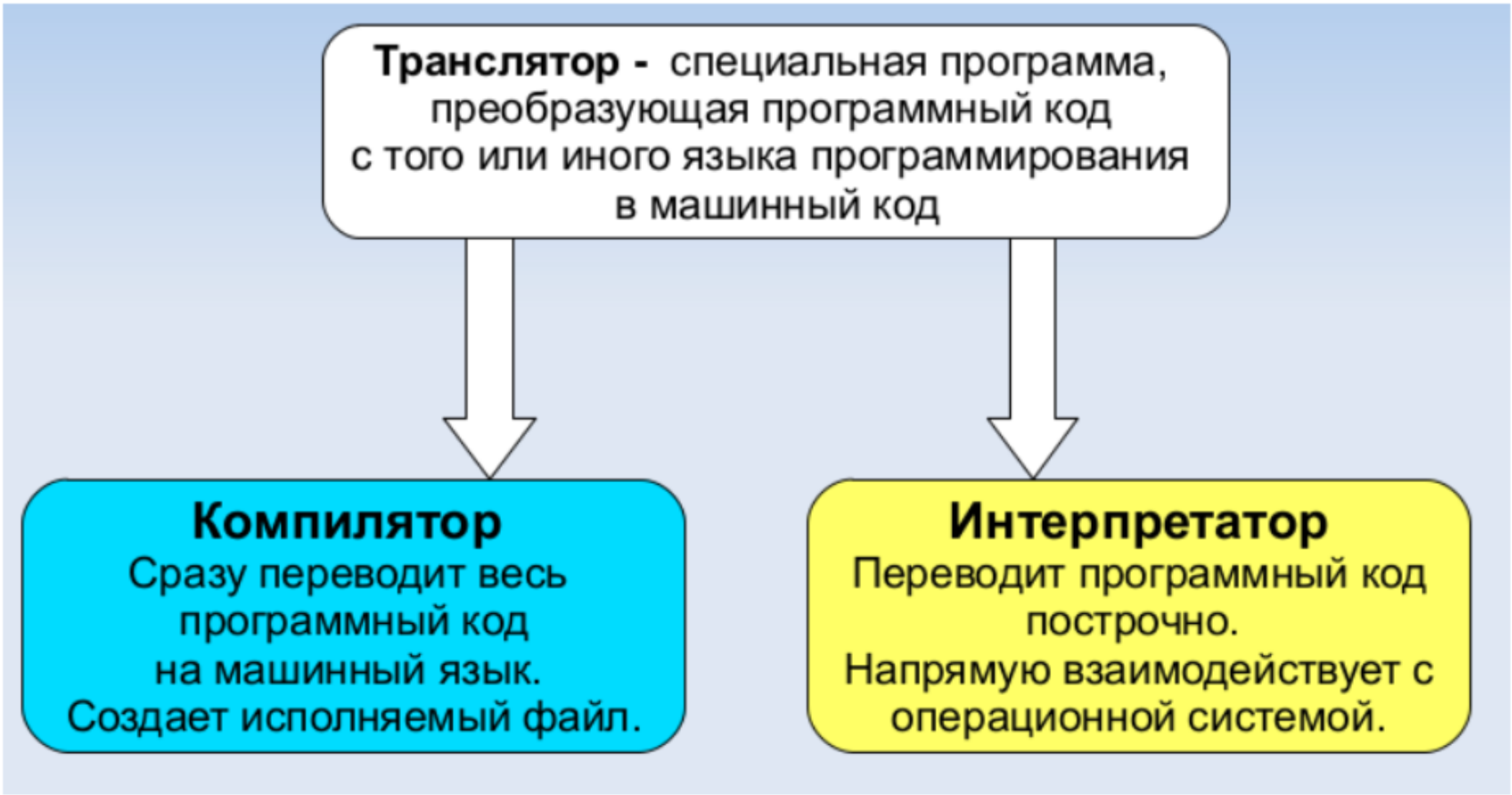
Программа, языки программирования

Основные этапы исторического развития языков программирования

Разнообразие языков программирования



Трансляция


Транслятор - специальная программа, преобразующая программный код с того или иного языка программирования в машинный код
Компилятор
Сразу переводит весь программный код на машинный язык.
Создает исполняемый файл.
Интерпретатор
Переводит программный код построчно.
Напрямую взаимодействует с операционной системой.

Как описать язык

- Семантика
- Синтаксис
- Прагматика

История языка Python

Язык программирования Python был создан к 1991 году голландцем Гвидо ван Россумом.

Свое имя - Пайтон (или Питон) - получил от названия популярного в 70-х годах британского телесериала «Летающий цирк Монти Пайтона», а не пресмыкающегося.

Официальный сайт поддержки языка - http://python.org

Основные особенности языка

- Python - интерпретируемый язык программирования.
- Python характеризуется ясным синтаксисом.
- Python используется в различных сферах.
- Интерпретаторы Python распространяются свободно на основании лицензии подобной GNU General Public License

Сильные стороны Python

- **Объектно-ориентированный**
- **Мощный**
  - *Динамическая типизация*
  - *Автоматическое управление памятью*
  - *Модульное программирование*
  - *Встроенные типы объектов*
  - *Встроенные инструменты*
  - *Библиотеки утилит*
  - *Утилиты сторонних разработчиков*
- **Прост в изучении**

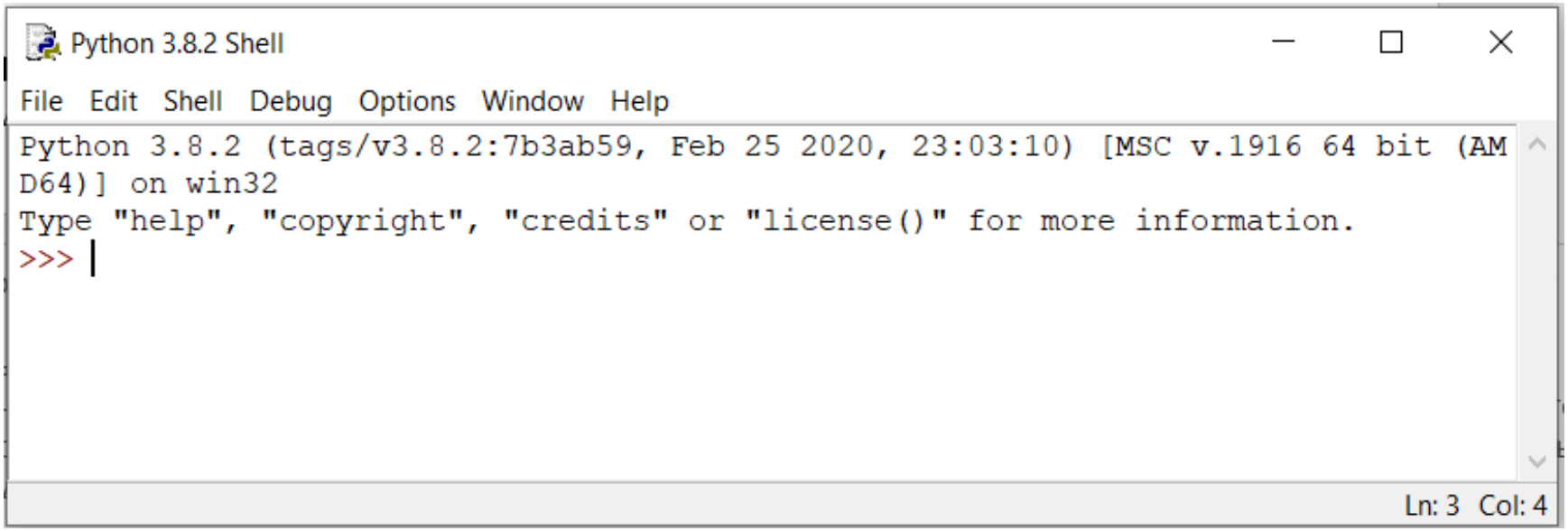
Дзен Python

import this

Явное лучше неявного.
Простое лучше сложного.
Сложное лучше усложнённого.
Плоское лучше вложенного.
Разрежённое лучше плотного.
Удобочитаемость важна.
Частные случаи не настолько существенны, чтобы нарушать правила.
Однако практичность важнее чистоты.
Ошибки никогда не должны замалчиваться.
За исключением замалчивания, которое задано явно.
В случае неоднозначности сопротивляйтесь искушению угадать.
Должен существовать один - и, желательно, только один - очевидный способ сделать это.
Хотя он может быть с первого взгляда не очевиден, если ты не голландец.
Сейчас лучше, чем никогда.
Однако, никогда чаще лучше, чем прямо сейчас.
Если реализацию сложно объяснить – это плохая идея.
Если реализацию легко объяснить - это может быть хорошая идея.
Пространства имён - прекрасная идея, давайте делать их больше!

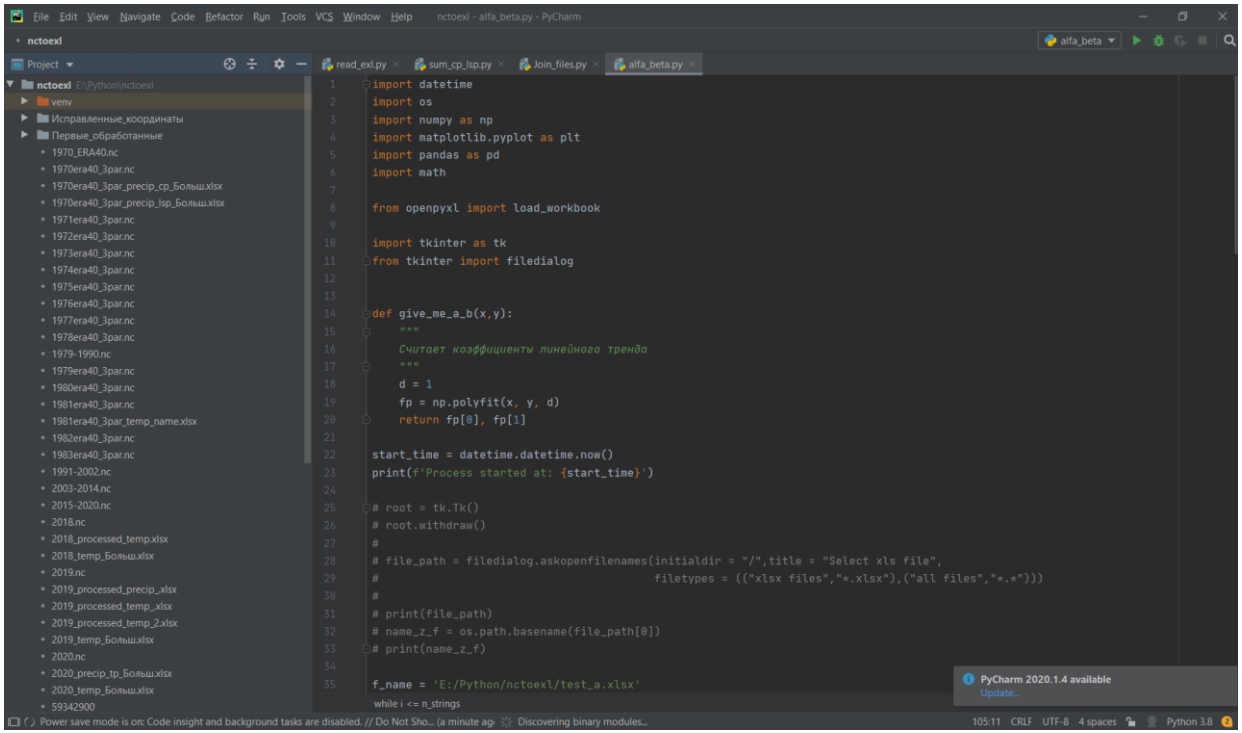
Как писать программы на Python
Интерактивный режим

Для экспериментов и тестирования.



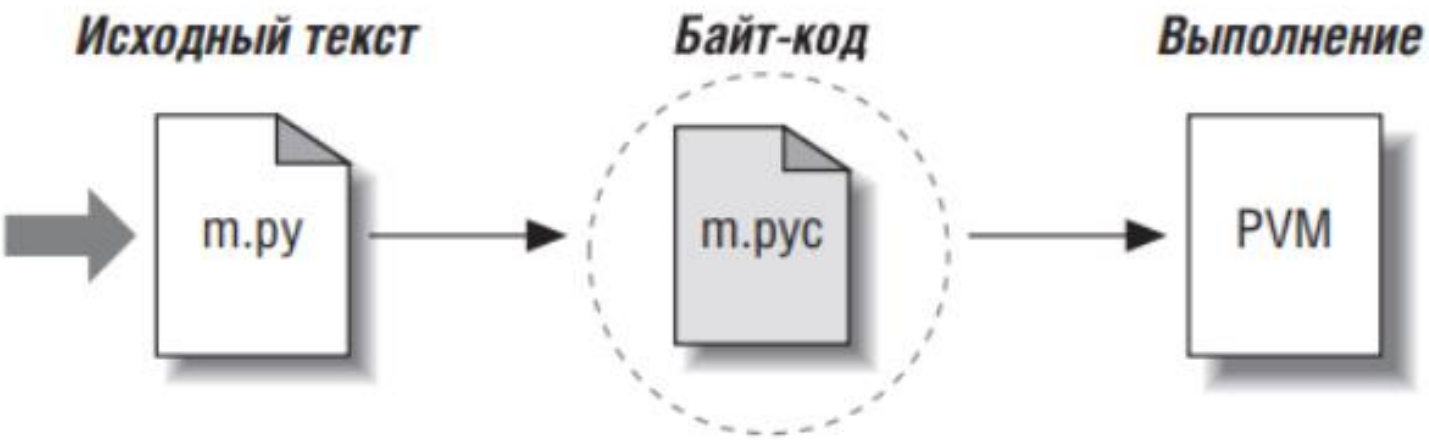
В среде IDLE (в Windows) для этого используются сочетания клавиш (скорее всего **Alt+N и Alt+P,** в MAC OS **Ctrl-P и Ctrl-N)**

Как писать программы на Python
Создание скриптов



Файлы с кодом на Python обычно имеют расширение *.py

Выполнение программы с точки зрения python

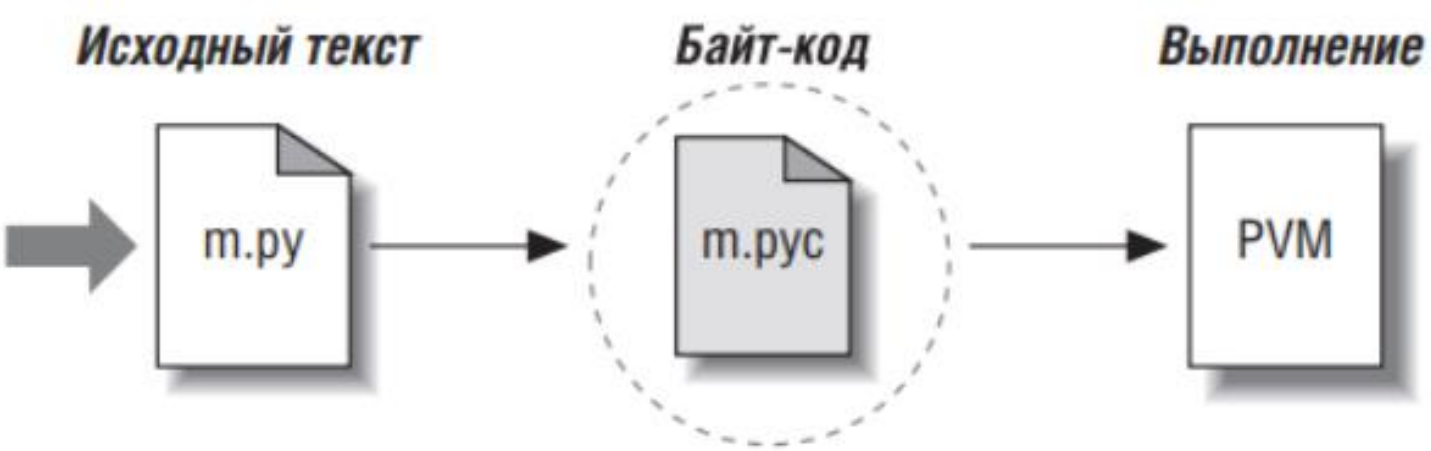


Выполнение программы с точки зрения python
Виртуальная машина Python (PVM)

Python Virtual Machine (PVM)

Фактически **PVM** – это просто большой цикл, который выполняет перебор инструкций в байт-коде, одну за одной, и выполняет соответствующие им операции.

Выполнение программы с точки зрения python



Фиксированные двоичные файлы

Фиксированные двоичные файлы **(frozen binaries).**

Три инструмента создания фиксированных двоичных файлов:

- ***Py2exe***
- ***PyInstaller***
- ***freeze***

Введение в типы объектов языка Python

Объекты – это области памяти со значениями и ассоциированными с ними наборами операций.

Программы на языке Python можно разложить на:

1. Программы делятся на модули.
2. Модули содержат инструкции.
3. Инструкции состоят из выражений.
4. *Выражения создают и обрабатывают объекты.*

Базовые типы данных в языке Python

Тип объекта	Пример литерала/создания
Числа	`1234, 3.1415, 3+4j, Decimal, Fraction`
Строки	`'spam', "guido's", b'a\x01c'`
Списки	`[1, [2, 'three'], 4]`
Словари	`{'food': 'spam', 'taste': 'yum'}`
Кортежи	`(1,'spam', 4, 'U')`
Файлы	`myfile = open('eggs', 'r')`
Множества	`set('abc'), {'a', 'b', 'c'}`
Прочие базовые типы	Сами типы, `None`, логические значения

Операции в программировании


```
>>> 10.25 + 98.36
108.61
>>> 'Hello' + 'World'
'HelloWorld'
```

Знак плюса неприменим, если операндами являются, с одной стороны, число, а с другой - строка

```
>>> 1 + 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Математические операторы

Оператор	Описание	Пример	Результат
+	Сложение	7 + 3	10
-	Вычитание	7 - 3	4
*	Умножение	7 * 3	21
/	Деление (истинное)	7 / 3	2.3333333333333335
**	Возведение в степень	7**3	343
//	Целочисленное деление	7 // 3	2
%	Остаток от деления	7 % 3	1

Изменение типов данных

```
>>> str(1) + 'a'
'1a'
>>> int('3') + 4
7
>>> float('3.2') + int('2')
5.2
>>> str(4) + str(1.2)
'41.2'
```

Однако надо понимать, что преобразовать можно не все:

```
>>> int('hi')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'hi'
```

```
>>> int('101', 2)
5
>>> int('F', 16)
15
```

Переменные

Имена переменных могут быть любыми. Однако есть несколько общих правил их написания:

1. Желательно давать переменным осмысленные имена, говорящие о назначении данных, на которые они ссылаются.
2. Имя переменной не должно совпадать с командами языка (зарезервированными ключевыми словами).
3. Имя переменной должно начинаться с буквы или символа подчеркивания (_), но не с цифры.
4. Имя переменной не должно содержать пробелы.

Переменные

Чтобы узнать значение, на которое ссылается переменная, находясь в режиме интерпретатора, достаточно ее вызвать, т. е. написать имя и нажать Enter.

```
>>> sq = 4
>>> sq
4
```

Вот более сложный пример работы с переменными в интерактивном режиме:

```
>>> apples = 100
>>> eat_day = 5
>>> day = 7
>>> apples = apples - eat_day * day
>>> apples
65
```

Ввод и вывод данных

- осуществляется с помощью встроенных функций

Ввод: input (параметры)

Вывод: print (параметры)

Вывод данных

```
>>> print(1032)
1032
>>> print(2.34)
2.34
>>> print("Hello")
Hello
```

В скобках могут быть любые типы данных. Кроме того, количество данных может быть различным:

```
>>> print("a:", 1)
a: 1
>>> one = 1
>>> two = 2
>>> three = 3
>>> print(one, two, three)
1 2 3
```

Вывод данных

Если в скобках стоит выражение, то сначала оно выполняется, после чего print() уже выводит результат данного выражения:

```
>>> print("hello" + " " + "world")
hello world

>>> print(10 - 2.5/2)
8.75
```

В print() предусмотрены дополнительные параметры. Например, через параметр **sep** можно указать отличный от пробела разделитель строк:

```
>>> print("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun", sep="-")
Mon-Tue-Wed-Thu-Fri-Sat-Sun
>>> print(1, 2, 3, sep="//")
1//2//3
```

Вывод данных

Параметр **end** позволяет указывать, что делать, после вывода строки. По умолчанию происходит переход на новую строку. Однако это действие можно отменить, указав любой другой символ или строку:

```
>>> print(10, end="")
10>>>

>>> print(10, end='\n')
10
>>>

>>> print(10, end='\n\n')
10

>>>
```

Вывод данных
использование форматирования строк

```
>>> pupil = "Ben"
>>> old = 16
>>> grade = 9.2
>>> print("It's %s, %d. Level: %f" % (pupil, old, grade))
It's Ben, 16. Level: 9.200000
```

Здесь вместо трех комбинаций символов %s, %d, %f подставляются значения переменных pupil, old, grade. Буквы s, d, f обозначают типы данных - строку, целое число, вещественное число. Если бы требовалось подставить три строки, то во всех случаях использовалось бы сочетание %s.

```
>>> print("It's %s, %d. Level: %.1f" % (pupil, old, grade))
It's Ben, 16. Level: 9.2
```

Вывод данных
использование форматирования строк

```
>>> print("This is a {0}. It's {1}.".format("ball", "red"))
This is a ball. It's red.
>>> print("This is a {0}. It's {1}.".format("cat", "white"))
This is a cat. It's white.
>>> print("This is a {0}. It's {1} {2}.".format(1, "a", "number"))
This is a 1. It's a number.
```

В строке в фигурных скобках указаны номера данных, которые будут сюда подставлены. Далее к строке применяется метод format(). В его скобках указываются сами данные (можно использовать переменные). На нулевое место подставится первый аргумент метода format(), на место с номером 1 - второй и т. д.

F-строки Python 3
улучшенный синтаксис форматирования строк

Также называемые «**formatted string literals**» («отформатированные строковые литералы»)

```
>>> name = "Eric"
>>> age = 74
>>> f"Hello, {name}. You are {age}."
Hello, Eric. You are 74.
```

F-строки Python 3
улучшенный синтаксис форматирования строк

```
>>> f"{2 * 37}"
74

>>> def to_lowercase(input):
        return input.lower()
>>> name = "Eric Idle"
>>> f"{to_lowercase(name)} is funny."
eric idle is funny.
```

Многострочные f-строки

```
>>> name = "Eric"
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> message = (
        f"Hi {name}. "
        f"You are a {profession}. "
        f"You were in {affiliation}."
    )
>>> message
Hi Eric. You are a comedian. You were in Monty Python.
```

Многострочные f-строки

```
>>> name = "Eric"
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> message = f"Hi {name}. " \
      f"You are a {profession}. " \
      f"You were in {affiliation}."
>>> message
Hi Eric. You are a comedian. You were in Monty Python.
```

Python f-строки: особенности использования

Кавычки

```
>>> f"{'Eric Idle'}"
Eric Idle

>>> f'{"Eric Idle"}'
Eric Idle

>>> f""""Eric Idle""""
Eric Idle

>>> f'''Eric Idle'''
Eric Idle

>>> f"The \"comedian\" is {name}, aged {age}."
The "comedian" is Eric Idle, aged 74.
```

Ввод данных

```
>>> input()
Yes!
'Yes!'
```

```
>>> answer = input()
No, it is not.
```

```
>>> answer
'No, it is not.'
```

```
>>> print(answer)
No, it is not.
```

Ввод данных

test.py

```
1  nameUser = input()
2  cityUser = input()
3  print("Вас зовут {0}. Ваш город {1}.".format(nameUser, cityUser))
4
```

Ввод данных

test.py

```
1  nameUser = input("Ваше имя: ")
2  cityUser = input("Ваш город: ")
3  print("Вас зовут {0}. Ваш город {1}.".format(nameUser, cityUser))
4
```

Статус
Компилятор
Сообщения
Заметки
Терминал

```
pl@pl-desk:~$ python3 test.py
Ваше имя: Сикама
Ваш город: где-то в Африке
Вас зовут Сикама. Ваш город где-то в Африке.
pl@pl-desk:~$
```

Ввод данных

test.py

```
1  qtyOranges = input("Сколько апельсинов? ")
2  priceOrange = input("Цена одного апельсина? ")
3
4  qtyOranges = int(qtyOranges)
5  priceOrange = float(priceOrange)
6
7  sumOranges = qtyOranges * priceOrange
8
9  print("Заплатите", sumOranges, "руб.")
10
```

Статус
Компилятор
Сообщения
Заметки
Терминал

```
pl@pl-desk:~$ python3 test.py
Сколько апельсинов? 5
Цена одного апельсина? 21.50
Заплатите 107.5 руб.
pl@pl-desk:~$
```

Ввод данных

Программный код можно сократить, если преобразование типов выполнить в тех же строках кода, где вызывается функция input():

```
qtyOranges = int(input("Сколько апельсинов? "))
priceOrange = float(input("Цена одного апельсина? "))

sumOranges = qtyOranges * priceOrange

print("Заплатите", sumOranges, "руб.")
```