**Первые шаги**

**1.1 Установка Python**

Вначале необходимо установить на компьютер интерпретатор **Python**.

Для загрузки дистрибутива нужно скачать последнюю версию для Windows со страницы <http://python.org/download/>.

После установки вам будут доступны два исполняемых файла: **python.exe** и **pythonw.exe**. Файл **python.exe** предназначен для выполнения консольных приложений. Именно эта программа запускается при двойном щелчке на значке файла с расширением **ру**. Файл **pythonw.exe** используется для запуска оконных приложений. В этом случае окно консоли выводиться· не будет. Эта программа запускается при двойном щелчке на значке файла с расширением **pyw**.

Если сделать двойной щелчок на файле **python.exe**, то запустится интерактивная оболочка в окне консоли (рис. 1.1). Символы >>> в этом окне означают приглашение для ввода инструкций на языке **Python**. Если после этих символов ввести, например, 2 + 2 и нажать клавишу <**Enter**>, то на следующей строке сразу будет выведен результат выполнения, а затем опять приглашение для ввода новой инструкции. Таким образом, это окно можно использовать в качестве калькулятора, а также для изучения языка. Открыть такое же окно можно через меню **Пуск**.

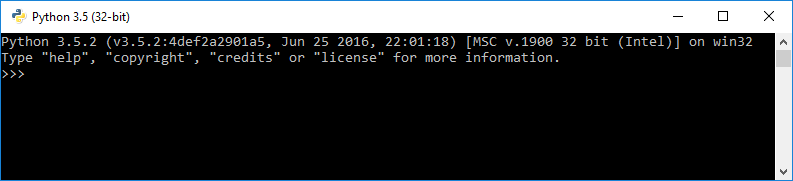


Рис. 1.1 – Интерактивная оболочка

Вместо интерактивной оболочки для изучения языка, а также создания и редактирования файлов с программой, лучше воспользоваться редактором **IDLE**, который входит в состав установленных компонентов. Для запуска редактора нужно использовать меню Пуск с пунктом **IDLE**. В результате откроется окно **Python Shell** (рис. 1.2), которое выполняет все функции интерактивной оболочки, но дополнительно производит подсветку синтаксиса, выводит подсказки и др. Именно этим редактором мы будем пользоваться. Более подробно редактор **IDLE** мы рассмотрим немного позже.

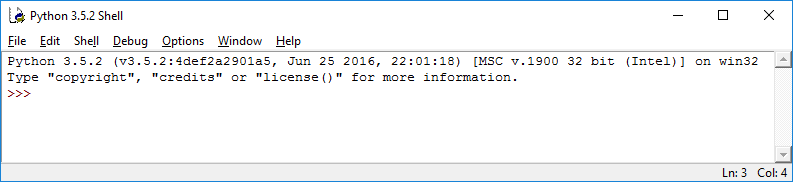


Рис. 1.2 - Окно Python Shell редактора IDLE

**1.2. Первая программа на Python**

Изучение языков программирования принято начинать с программы, выводящей надпись "Привет, мир!". Не будем нарушать традицию и продемонстрируем, как это будет выглядеть на Python (листинг 1.1).

Листинг 1.1 - Первая программа на Python



Запустите из меню **Пуск** **IDLE (Python GUI)**. В результате откроется окно **Python Shell**, в котором символы >>> означают приглашение ввести команду. Введите сначала первую строку листинга 1.1, а затем вторую. После ввода строки нажимайте клавишу <**Enter**>. Результат выполнения нашей программы показан на рисунке 1.3.

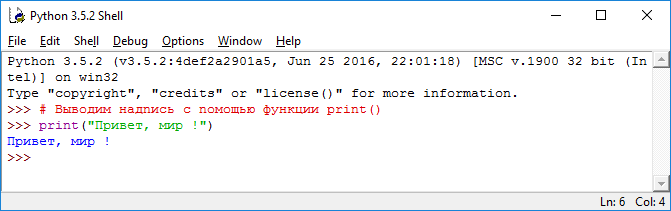


Рис. 1.3 – Результат выполнения программы

Присмотритесь к строке, которую вы ввели: **print( "Привет, мир !" )**. Владея английским хотя бы на базовом уровне, можно догадаться, что этот код значит . Таков по своей сути весь **Python**. Функция **print()** выводит на экран текст, который пользователь, заключив в кавычки, помещает внутрь скобок. Если ничем не заполнить скобки, будет напечатана пустая строка.

**ЛОВУШКА**

**Python** чувствителен к регистру. Существует договоренность, в силу которой названия функций пишутся строчными буквами . Поэтому заnись **print( "Привет, мир !" )** сработает, а **Print( "Привет, мир !" )** или **PRINT( "Привет, мир !" )**- нет.

Для создания файла с программой в меню **File** нужно выбрать пункт **New File**. В открывшемся окне нужно набрать код из листинга 1.1, а затем сохранить файл под именем **test.py**. При этом редактор сохранит файл в кодировке **UTF-8 без ВОМ** *(Byte Order Mark,* метка порядка байтов). Именно кодировка **UTF-8** является кодировкой по умолчанию в **Python 3**.

Запустить программу на выполнение можно, выбрав пункт меню **Run -> Run Module** или нажав клавишу <**F5**>. Результат выполнения программы будет отображен в окне **Python Shell**.

Запустить программу можно также с помощью двойного щелчка мыши на значке файла. В этом случае результат выполнения будет отображен в консоли **Windows**. Следует учитывать, что после вывода результата окно консоли сразу закрывается. Чтобы предотвратить закрытие окна, необходимо добавить вызов функции **input()**, которая будет ожидать нажатия клавиши <**Enter**> и не позволит окну сразу закрыться. Если до функции **input()** возникнет ошибка, то сообщение о ней будет выведено в консоль, но сама консоль после этого сразу будет закрыта, и вы не сможете прочитать сообщение об ошибке . Если подобная ситуация возникла, то запустите программу из командной строки или с помощью редактора IDLE и вы сможете прочитать сообщение об ошибке.

**1.3. Структура программы**

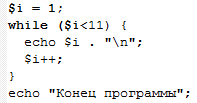
Как вы уже знаете, программа на языке **Python** представляет собой обычный текстовый файл с инструкциями. Каждая инструкция располагается на отдельной строке. Если инструкция не является вложенной, то она должна начинаться с начала строки, иначе будет выведено сообщение об ошибке.

Во многих языках программирования (например, в **РНР**, **Perl** и др.) каждая инструкция должна завершаться точкой с запятой. В языке **Python** в конце инструкции также можно поставить точку с запятой, но это не обязательно. В отличие от языка **JavaScript**, где рекомендуется завершать инструкции точкой с запятой, в языке **Python** точку с запятой ставить **не** рекомендуется. Концом инструкции является конец строки. Тем не менее, если необходимо разместить несколько инструкций на одной строке, то точку с запятой следует указать.

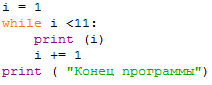
Листинг 1.2 - Несколько инструкций на одной строке



Еще одной отличительной особенностью языка **Python** является отсутствие ограничительных символов для выделения инструкций внутри блока. Например, в языке РНР инструкции внутри цикла **while** выделяются фигурными скобками:



В языке **Python** тот же код будет выглядеть по-другому

**

Обратите внимание, что перед всеми инструкциями внутри блока расположено одинаковое количество пробелов. Таким образом, в языке **Python** выделяются блоки. Инструкции, перед которыми расположено одинаковое количество пробелов, являются телом блока. В нашем примере две инструкции выполняются десять раз. Концом блока является инструкция, перед которой расположено меньшее количество пробелов. В нашем случае это функция **print()**, которая выводит строку "Конец программы". Если количество пробелов внутри блока будет разным, то интерпретатор выведет сообщение о фатальной ошибке и программа будет остановлена. Так язык **Python** приучает программистов писать красивый и понятный код.

**ПРИМЕЧАНИЕ**

В языке **Python** принято использовать **четыре** пробела для выделения инструкций внутри блока.

Если блок состоит из одной инструкции, то допустимо разместить ее на одной строке с основной инструкцией. Например, код



можно записать так:



Если инструкция является слишком длинной, то ее можно перенести на следующую строку нижеприведёнными способами:

* в конце строки разместить символ \. После этого символа должен следовать символ перевода строки. Другие символы (в том числе и комментарии) недопустимы. Пример:



* поместить выражение внутри круглых скобок. Это лучший способ, т. к. любое выражение можно разместить внутри круглых скобок. Пример:



* определение списка и словаря можно разместить на нескольких строках, т. к. используются квадратные и фигурные скобки соответственно. Пример определения списка:



Пример определения словаря:



**1.4. Комментарии**

Комментарии предназначены для вставки пояснений в текст скрипта, и интерпретатор полностью их игнорирует. Внутри комментария может располагаться любой текст, включая инструкции, которые выполнять не следует. Помните, комментарии нужны программисту, а не интерпретатору **Python**. Вставка комментариев в код позволит через некоторое время быстро вспомнить предназначение фрагмента кода.

В языке **Python** присутствует только однострочный комментарий. Он начинается с символа #:

**# Это комментарий**

Однострочный комментарий может начинаться не только с начала строки, но и располагаться после инструкции. Например, после инструкции вывести надпись **"Привет, мир!"**:

**print ("Привет, мир!") # Выводим надпись с помощью функции print ()**

Если символ комментария разместить перед инструкцией, то она не будет выполнена:

**# рrint("Привет, мир!") Эта инструкция выполнена не будет**

Если символ **#** расположен внутри кавычек или апострофов, то он не является символом комментария:

**print("# Это НЕ комментарий")**

Так как в языке **Python** нет многострочного комментария, то часто комментируемый фрагмент размещают внутри утроенных кавычек (или утроенных аnострофов):

**"""**

**Эта инструкция выполнена не будет**

**print ("Привет, мир!")**

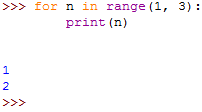
**"""**

Следует заметить, что этот фрагмент кода не игнорируется интерпретатором, т. к. он не является комментарием. В результате выполнения фрагмента будет создан объект строкового типа. Тем не менее, инструкции внутри утроенных кавычек выполнены не будут, т. к. все инструкции будут считаться простым текстом. Такие строки являются строками документирования, а не комментариями.

**1.5. Скрытые возможности IDLE**

Как вы уже знаете, в окне **Python Shell** символы >>>означают приглашение ввести команду. После ввода команды нажимаем клавишу <**Enter**>. На следующей строке сразу отобразится результат (при условии, что инструкция возвращает значение), а далее - приглашение для ввода новой команды. При вводе многострочной команды после нажатия клавиши <**Enter**> редактор автоматически вставит отступ и будет ожидать дальнейшего ввода. Чтобы сообщить редактору о конце ввода команды, необходимо **дважды** нажать клавишу <**Enter**>.

Пример:



В предыдущем разделе мы выводили строку **"Привет, мир!"** с помощью функции **print()**. В окне **Python Shell** это делать не обязательно. Например, можно просто ввести строку и нажать клавишу <**Enter**> для получения результата:



Обратите внимание на то, что строки выводятся в апострофах. Этого не произойдет, если выводить строку с помощью функции **print()**:



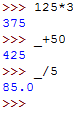
Учитывая возможность получить результат сразу после ввода команды, **окно Python Shell** можно использовать для изучения команд, а также в качестве многофункционального калькулятора.

Пример:



Результат вычисления последней инструкции сохраняется в переменной \_(одно подчеркивание). Это позволяет производить дальнейшие расчеты без ввода предыдущего результата. Вместо него достаточно ввести символ подчеркивания.

Пример:



При вводе команды можно воспользоваться комбинацией клавиш **<Сtrl>+<Пробел>**. В результате будет отображен список, из которого можно выбрать нужный идентификатор. Если при открытом списке вводить буквы, то показываться будут идентификаторы, начинающиеся с этих букв. Выбирать идентификатор необходимо с помощью клавиш <↑> и <↓>. После выбора **не следует** нажимать клавишу <**Enter**>, иначе это приведет к выполнению инструкции. Просто вводите инструкцию дальше, а список закроется. Такой же список будет автоматически появляться (с некоторой задержкой) при обращении к атрибутам объекта или модуля после ввода точки. Для автоматического завершения идентификатора после ввода первых букв можно воспользоваться комбинацией клавиш <Alt>+</>. При каждом последующем нажатии этой комбинации будет вставляться следующий идентификатор. Эти две комбинации клавиш очень удобны, если вы забыли, как пишется слово, или хотите, чтобы редактор закончил его за вас.

При необходимости повторно выполнить ранее введенную инструкцию ее приходится набирать заново. Можно использовать комбинации клавиш быстрого доступа - **<Ctrl>+<C>** (копировать) и **<Ctrl>+<V>** (вставить). Комбинации стандартны для **Windows**, и вы наверняка их уже использовали ранее. Прежде чем скопировать инструкцию, ее предварительно необходимо выделить. Редактор **IDLE** предоставляет комбинацию клавиш **<Alt>+<N>** для вставки первой введенной инструкции, а также комбинацию **<Alt>+<P>** для вставки последней инструкции. Каждое последующее нажатие этих клавиш будет вставлять следующую (или предыдущую) инструкцию. Для еще более быстрого повторного ввода инструкции следует предварительно ввести ее первые буквы. В этом случае перебираться будут только инструкции, начинающиеся с этих букв.

Вы, вероятно, заметили, что слова на экране отображаются разными цветами. Такое цветовое кодирование называется подсветкой синтаксиса и упрощает понимание того, что именно вы печатаете. Каждое слово окрашивается в один из цветов, и отнюдь не бессистемно. Встроенные идентификаторы в языке **Python**, - такие как **print** – отображаются фиолетовым цветом. Строки имеют зеленый цвет. Результат работы программы интерпретатор выводит на экран шрифтом голубого цвета.

**1.6. Вывод результатов работы программы**

Вывести результаты работы программы можно с помощью функции **print()**. Функция имеет следующий формат:



Функция **print()** преобразует объект в строку и посылает ее в стандартный вывод **stdout**. С помощью параметра **file** можно перенаправить вывод в другое место, например в файл. После вывода строки автоматически добавляется символ перевода строки.

Если необходимо вывести результат на той же строке, то в функции **print()** данные указываются через запятую в первом параметре:



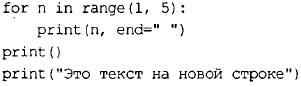
Как видно из примера, между выводимыми строками автоматически вставляется пробел. С помощью параметра **sep** можно указать другой символ. Выведем строки без пробела между ними:



После вывода объектов в конце добавляется символ перевода строки. Если необходимо произвести дальнейший вывод на той же строке, то в параметре **end** следует указать другой символ:



Если наоборот необходимо вставить символ перевода строки, то функция **print()** указывается без параметров. Пример:



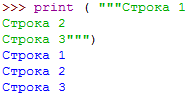
Результат выполнения:



В этом примере использован цикл **for**, который позволяет последовательно перебирать элементы. На каждой итерации цикла переменной **n** присваивается новое число, которое выводится с помощью функции **print()**, расположенной на следующей строке. Обратите внимание, что перед функцией добавлено четыре пробела. Таким образом, в языке **Python** выделяются блоки. Инструкции, перед которыми расположено одинаковое количество пробелов, являются телом цикла. Все эти инструкции выполняются определенное количество раз. Концом блока является инструкция, перед которой расположено меньшее количество пробелов. В нашем случае это функция **print()** без параметров, которая вставляет символ перевода строки.

Если необходимо вывести большой блок текста, то его следует разместить между утроенными кавычками или утроенными апострофами. В этом случае текст сохраняет свое форматирование.

Пример:



**1.7. Ввод данных**

Для ввода данных в **Python** **3** предназначена функция **input()**, которая получает данные со стандартного ввода **stdin**. Функция имеет следующий формат:

**[<Значение> = ] input ( [ <Сообщение> ] )**

Для примера переделаем нашу первую программу так, чтобы она здоровалась не со всем миром, а только с нами (листинг 1.3).

Листинг 1.3 - Пример использования функции **input()**



Введите код и сохраните файл под именем **test.py**, а затем запустите программу на выполнение с помощью двойного щелчка на значке файла. Откроется черное окно, в котором будет надпись "Введите ваше имя: ". Введите свое имя, например "Николай", и нажмите клавишу <**Enter**>. В результате будет выведено приветствие "Привет, Николай". Чтобы окно сразу· не закрылось, повторно вызывается функция **input()**. В этом случае окно не закроется, пока не будет нажата клавиша <**Enter**>.

