ЛАБОРАТОРНАЯ РАБОТА

Тема: ***Знакомство со средой программирования***

Цель работы: Получение общего представления о программировании на Питоне версии 3.x и режимах выполнения программ.

Содержание

Интерактивный и сценарный режимы выполнения программ

Для выполнения программного кода на Питоне применяются два режима: *интерактивный* и *сценарный*.

Интерактивный режим предполагает диалог пользователя и системы, который поддерживается интерпретатором Питона. После запуска интерпретатора на экране появляется *приглашение* ввести инструкцию (команду), которое имеет вид >>>.

Когда пользователь завершил ввод инструкции (`Enter`), интерпретатор считывает её, выполняет необходимые вычисления, выводит на экран результат (если такой имеется) и переходит в режим ожидания следующей команды.

Эта схема работы, в которой команды вводятся и выполняются *по одной*, носит название `REPL` — "**`R`**`ead, `**`E`**`valuate,`**`P`**`rint `**`L`**`oop`". Она представляет собой циклический процесс (loop), в котором раз за разом повторяются три шага: `"Прочесть введенную команду — Выполнить вычисления — Вывести результат"`.

Интерактивный режим удобен, когда нужно, например, поэкспериментировать с несколькими вариантами некоторой программной конструкции, или решать несложные учебные задачи при изучении основ программирования.

Хотя выполненные в интерактивном режиме команды запоминаются на время текущего сеанса работы, и их можно по одной извлекать для повторного выполнения, после завершения сеанса содержание всего диалога теряется.

Но в тех случаях, когда программа достаточно большая по объему и предполагается её многократное использование, код нужно запомнить, и интерактивный метод работы для этого не подходит.

Вместо него применяется *сценарный режим*, в котором программист

а) в текстовом редакторе (plain text) готовит текст программы (т.е. *сценарий* вычислений),

б) сохраняет исходный код в файле с расширением `.py`,

в) загружает файл в среду Питона, где программа без вмешательства пользователя (если не предполагается ввод данных) автоматически выполняется от начала до конца.

Термин "сценарий" (программисты также говорят "скрипт", от англ. *script*) обычно применяется для обозначения текстов программ, предназначенных для интерпретации.

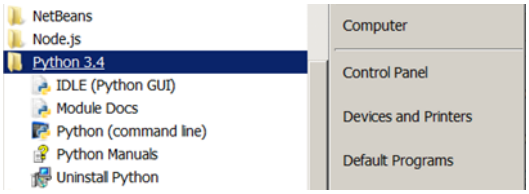
Хотя два режима выполнения программ имеют принципиальные отличия, на практике они дополняют друг друга и оба одинаково полезны программисту. Начало разработки может проходить в интерактивном режиме, когда в экспериментальном порядке тестируются черновые

наброски кода, а окончательный вариант реализуется как сценарий, который затем распространяется в виде файла.

Поэтому среды разработки программ на Питоне позволяют использовать оба режима параллельно.

Интерпретатор Питона

После установки Питона в меню кнопки Пуск (Start) "`Все программы`" появляется папка с именем версии Питона.



Находящиеся в этой папке программы `Python Manuals` и `Module Docs` обеспечивают доступ к различной документации и справочной информации.

Программы IDLE и Python (command line) позволяют запускать интерпретатор Питона и выполнять с его помощью программный код.

Работа в окне текстовой консоли

Текстовая консоль — это окно, используемое для взаимодействия пользователя с компьютером, т.е. для ввода *текста* команд и получения *текста* с результатами. Элементы графического интерфейса не применяются, а для вывода информации используются простейшие системные шрифты.

Интерпретатор Питона запускается при выполнении программы `python.exe` (без параметров!)

Для запуска программы требуется командная строка операционной системы, доступ к которой возможен несколькими способами.

а) Командная строка всегда активна в нижней части окна файлового менеджера FAR. Панели с файлами можно скрыть/отобразить клавишами `Ctrl+O`.

б) Окно операционной системы `Run` со строкой для ввода команды открывается при нажатии клавиш `Win+R`.

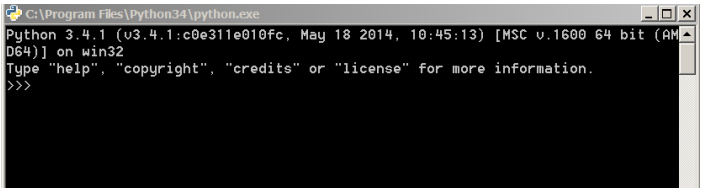
в) Окно с командной строкой открывается при запуске программы `Command Prompt` из списка программ кнопки "Пуск" `Все программы | Стандартные`,

Так как при установке Питона путь к программе прописывается в системной переменной, то вызвать интерпретатор можно из любой текущей папки без указания пути:

```
c:\...\> python
```

В ответ появится так называемый терминал `Python`. Это "черное" окно с сообщением об используемой версии Python и строка с перечислением утилит (вспомогательных программ), предоставляющих пользователю справочную информацию о системе: `"help", "copyright", "credits", "license"`.

Свидетельством того, что интерпретатор запущен, является *приглашение* (primary prompt) для ввода команд в виде трех знаков "больше": `>>>`.



Далее работа следует схеме `REPL`. После приглашения вводится некоторая программная инструкции языка `Python`. После нажатия `Enter` она выполняется интерпретатором и, если у инструкции есть результат, он выводится на экран. После чего появляется новое приглашение и

интерпретатор переходит в режим ожидания следующей инструкции. Этим способом, в частности, могут выполняться утилиты, перечисленные при запуске интерпретатора.

Использование утилиты help

Утилита `help()` предоставляет справки по любой из базовых конструкций языка `Python`. Она имеет два режима работы.

- Можно получить сведения по конкретной команде Питона. Для этого имя команды нужно поместить в скобки после `help` и взять его в кавычки. Например, для получения справки об использовании команды `pass` (не выполняет никаких действий, "заглушка" для команды), нужно выполнить инструкцию:

```
>>> help("pass")
```

- Если ввести команду `help` с пустыми скобками, то утилита запустит свой собственный специализированный интерпретатор и изменит вид приглашения:

```
help>
```

В этом специальном режиме получить справку по конкретной команде проще. Достаточно ввести после приглашения её имя:

```
help> pass
```

- Чтобы завершить работу утилиты `help` и вернуться в обычный режим интерпретации, нужно вести после приглашения символ `q` (от *quit* — выйти) и нажать клавишу `Enter`.

Три другие утилиты `copyright()`, `credits()` и `license()` сообщают информацию, относящуюся к авторским правам и лицензии на программный продукт.

Повторное выполнение команд

Набранные команды запоминаются интерпретатором (т.е. ведется история команд) и их можно в обратном порядке извлекать из памяти с помощью клавиши "стрелка вверх", и затем перемещаться вперед к более поздним командам с помощью клавиши "стрелка вниз".

После выбора команды, её можно отредактировать или просто выполнить, нажав `Enter`.

Выход из интерпретатора

Для выхода из интерпретатора можно

- либо выполнить команду `exit()` (или аналогичную ей `quit()`),
- либо использовать сочетание клавиш `Ctrl+Z` и нажать `Enter`.

Замечание

Интерпретатор также прекращает работу, когда закрывается окно текстовой консоли.

ЗАДАНИЕ 1 (*Простейшие действия с интерпретатором*)

1. Запустить интерпретатор в режиме текстовой консоли.
2. Получить справку по команде `print`, используя утилиту `help` с параметром.

Замечание

При выполнении заданий учесть, что в Питоне пробелы *перед* инструкцией (отступ) имеют принципиальное значение. Даже один введенный после приглашения, но непредусмотренный синтаксисом пробел, вызовет ошибку.

3. Выполнить команду `help` без параметров и скобок. В чем смысл полученного сообщения?
4. Выполнить команду `help` с пустыми скобками. Получить справку по команде `print`. Выйти из интерпретатора `help`.
5. Выполнить команду `print("Python")`
6. Выполнить команду `print("2+3=",2+3)`
7. Вернуться (с помощью стрелки) к команде `print` из пункта 5, добавить к тексту восклицательный знак и выполнить отредактированную команду.
8. Выйти из интерпретатора.

Выполнение сценария из командной строки

В простейшем случае программа `python` (т.е. `python.exe`) вызывается из командной строки операционной системы без параметров и запускает интерпретатор.

Однако программе могут быть также переданы параметры, изменяющие её поведение.

Наиболее важным является случай, когда параметром является файл с программой на Питоне. В этом случае `python.exe` рассматривает и выполняет этот файл как сценарий (не открывая интерпретатора). Результаты выполнения программы в окне консоли появятся только в том случае, если в программе предусмотрены операции вывода информации на стандартное устройство (текстовую консоль).

Замечание

Обычно при инсталляции Питона устанавливается ассоциация расширения .py с программой python.exe. Поэтому для выполнения программы может оказаться достаточным двойной щелчок на имени файла программы в Проводнике. Однако в этом случае окно программы закроется сразу, после завершения программы, и посмотреть результаты не удастся.

Чтобы задержать закрытие окна, в конец программы можно добавить команду `input()`, которая заставит окно ждать нажатие `Enter`.

ЗАДАНИЕ 2 (*Выполнение программ из командной строкой*)

1. Чтобы получить справку по использованию программы `python.exe`, в командной строке операционной системы нужно вызвать её с параметром `-h` (или `/?`), т.е.

```
C:\...\python -h
```

В выведенной строке, описывающей вызов программы,

```
usage:
    C:\Program Files\Python34\python.EXE [option] [-c cmd | -m mod | file | -] [arg]
```

необязательные параметры взяты в квадратные скобки, при перечислении вариантов параметров вертикальная черта означает "или".

Проанализировать полученный текст и выяснить, что делает программа, когда в качестве параметра ей передается файл.

2. В любом текстовом редакторе (`Блокнот`, `FAR`) создать файл `my_fisrt.py`, который содержит следующий программный код

```
s="Hello, Python"
print(s)
```

3. В командной строке вызвать программу `python` с файлом `my_fisrt.py` в качестве параметра.

Интегрированные среды разработки

Процесс разработки программ включает несколько этапов: подготовка файла с исходным текстом программы, трансляция программы, сборка исполняемого модуля программы, поиск логических ошибок в работающей программе и т.д. На каждом этапе программист использует соответствующие инструменты, т.е. каждую специальную программу программист должен запустить, передав ей соответствующие данные. В "ручном" режиме это требует от программиста значительных усилий и внимания. Причем это направлено не на решение задач, а на выполнение вспомогательных операций.

Чтобы упростить и автоматизировать такие действия, создаются *интегрированные среды разработки* (`IDE` — **I**ntegrated **D**evelopment **E**nvironment).

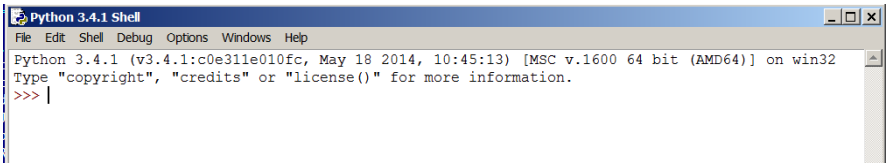
Это комплексы инструментов, необходимых для разработки и отладки программ, которые объединяются в общий пакет (оболочку). У программиста создается иллюзия, что он работает с интерфейсом одной большой программы. Хотя реально из меню им запускаются разные программы, которым оболочкой автоматически передаются необходимые данные.

Обычно в IDE входят текстовый редактор, компилятор и/или интерпретатор, отладчик (`debugger`) и другие полезные программы.

Интегрированная среда IDLE

В базовый инсталляционный комплект Python входит интегрированная среда разработки IDLE (**I**ntegrated **D**eve**L**opment **E**nvironment), которая обеспечивает минимальные потребности профессиональной разработки программ.

Программа IDLE (на панели задач показывается как Python Shell) запускается из папки Python, которая находится в списке "Все программы" кнопки Пуск.



Фактически IDLE — это оболочка, в которую обернут тот же самый интерпретатор Python, что рассматривался ранее, только с дополнительными возможностями, упрощающими процесс разработки. Поэтому запуск IDLE начинается с вывода той же самой информации, что и в консольном окне, и приглашения.

Исключение составляет утилита help(), отсутствие которой объясняется возможностью открывать через пункт меню Help | Python Docs (горячая клавиша F1) программу Python Docs, с обширной документацией по Питону.

Замечание

В меню кнопки Пуск эта программа названа Python Manuals.

Кроме этого, в меню Help предлагается справочная информация по самой среде IDLE.

С сравнении с текстовой консолью IDLE обладает рядом существенных преимуществ. Программа имеет меню File и Edit с содержанием, похожим на меню текстовых редакторов, типа Блокнот. Это позволяет выполнять с текстом в окне стандартные операции — выделение, копирование, поиск, замена, вырезка или вставка текста.

Но, помимо этого, IDLE предлагает важные полезные функции, необходимые для анализа и отладки программ.

Подсветка синтаксиса

Программа строится из различных конструкций, образующих некоторую структуру. Программисту нужно хорошо ориентироваться в ней, иначе из-за неправильного представления могут возникать ошибки. Поэтому наглядное отображение кода программы может повысить надежность разработки.

Современные текстовые редакторы, в которых программисты готовят текст программ, отображают различные по смыслу элементы программ разными цветами. Это называется подсветкой синтаксиса.

Подсветка делает структуру кода более рельефной и обозримой, что полезно при анализе конструкции программы и удобно при поиске ошибок.

В IDLE есть цветовая схема, применяемая для подсветки по умолчанию (после установки Питона). Ключевые слова языка отображаются оранжевым цветом, символьные строки в тексте программы — зеленым, выводимые интерпретатором результаты — синим, а сообщения об ошибках — красным цветом.

Пример подсветки в окне интерактивного режима:

```
>>> for i in range(3):
        print("i=",i)

i= 0
i= 1
i= 2
```

При необходимости программист может изменить отдельные цвета или всю цветовую схему. Для этого нужно в меню `Options` выбрать пункт `Configure IDLE`, а в открывшемся окне вкладку `Highlighting`.

Некоторые приемы работы с интерпретатором

Вывод результатов вычислений

Обычно, чтобы показать результат вычисления *выражения*, в программах нужно использовать команду типа `print`.

Python в режиме интерпретации этого не требует, так как интерпретатор автоматически показывает результат любого вычисленного им выражения. Например,

```
>>> 1+2
3
```

Так как переменная может рассматриваться, как один из простейших типов выражений, то для получения текущего значения переменной вместо инструкции `print` достаточно указать интерпретатору имя переменной.

Повторное выполнение команд

Так как клавиши стрелок в окне редактора используются для перемещения по тексту, то извлечь ранее выполненную команду из списка с помощью "стрелка вверх", как это было при использовании текстовой консоли, в `IDLE` нельзя.

Но вместо них для перемещения по истории команд можно пользоваться сочетаниями клавиш:

`Alt+p` — получить предыдущую команду (от `previous` — предыдущий)

`Alt+n` — получить следующую команду (от `next` — следующий).

Так как окно интерпретатора программы `IDLE` поддерживает функции текстового редактора, то есть возможность выделить и скопировать ранее выполненную команду, а затем вставить её после приглашения как новую команду. До нажатия `Enter` инструкция может быть отредактирована.

Перезапуск интерпретатора

В IDLE работает тот же интерпретатор, что и в текстовой консоли, но предлагаются дополнительные возможности. В частности это касается перезапуска интерпретатора.

Работа с интерпретатором напоминает пошаговое выполнение программы. Например, если при выполнении инструкции переменная получила некоторое значение, то это значение запоминается и может использоваться в следующих командах.

Однако, если программист закончил работу с одной такой программой и хочет выполнять другую, сохраненные значения переменных могут мешать работе новой программе.

Чтобы избежать этого, требуется перезапустить интерпретатор. Для этого нужно в меню `Shell` выбрать пункт `Restart Shell` или воспользоваться клавишами быстрого доступа `Ctrl+F6`.

Замечание

При работе с текстовой консолью перезапуск интерпретатора нужно выполнить буквально: выйти из программы и заново её запустить.

ЗАДАНИЕ 3 (*Повторное выполнение команд, перезапуск интерпретатора*)

1. Запустить программу IDLE.
2. Выполнить инструкцию, в которой переменной `x` присваивается значение `1`:

```
>>> x=1
```

3. Убедиться в том, что интерпретатор запомнил значение переменной:

```
>>> print(x)
```

или просто

```
>>> x
```

4. Скопировать команду из пункта 2, вставить её после приглашения и изменить значение переменной на `5`. Проверить результат выполнения инструкции.

5. Перезапустить интерпретатор. Проверить, сохранилось ли в нем значение переменной `x`? Какое сообщение выдает интерпретатор?

Работа в IDLE в сценарном режиме

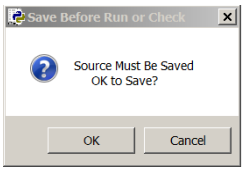
Программа IDLE обеспечивает работу и в интерактивном, и в сценарном режимах.

а) Создание файла сценария.

В сценарном режиме можно не только выполнить программу из файла, подготовленного во внешнем редакторе, но и создать программу во встроенном редакторе IDLE.

Для этого, после запуска программы нужно открыть меню `File` и выбрать пункт `New Window` (быстрый доступ — `Crtl+N`). Откроется новое окно, которое предназначено для работы с текстом программы (подсказки интерпретатора нет).

После завершения ввода текста программы, файл с кодом нужно сохранить с расширением `.py`. При попытке выполнить несохраненную программу, появится окно с предупреждением



б) Выполнение сценария из окна редактора.

Чтобы выполнить сценарий, его нужно поместить в окно редактора IDLE. Программа может быть либо создана в редакторе, либо загружена в него через меню `File | Open`...

Чтобы запустить программу, нужно выполнить команду `Run Module` из меню `Run` ("горячая" клавиша `F5`).

Результаты выполнения кода будут отображены в окне *интерпретатора*.

ЗАДАНИЕ 4 (*Выполнение вычисление по программе*)

Решается задача вычисления площади круга радиуса `5`. Задание выполнить в интерактивном и сценарном режимах.

В Питоне математические константы, элементарные и тригонометрические функции собраны вместе в модуле `math`. Поэтому, чтобы получить значение числа π, нужно использовать обращение `math.pi`.

Решение задачи в интерактивном режиме

1. Запустить интерпретатор в оболочке IDLE.
2. Набрать и выполнить следующие команды

```
>>> r=5
>>> area=math.pi*r**2
```

Сообщение об ошибке связано с тем, что модуль `math` является внешним по отношению к транслятору Питона (т.е. он об этом модуле ничего не знает). Поэтому модуль нужно специальным образом подключить в программе (импортировать).

3. Выполнить инструкцию

```
>>> import math
```

4. Скопировать инструкцию с вычислением площади круга `area` и заново выполнить её.
5. Выполнить инструкцию

```
>>>  print("Площадь круга равна ", area)
```

Решение задачи в сценарном режиме

1. Открыть в IDLE окно текстового редактора, набрать текст программы, в которой вычисляется и выводится площадь круга и сохранить программу в файле `area.py`.

Программа содержит все действия, которые выполнялись в интерактивном режиме.

Инструкция `import` должна появиться в программе раньше использования модуля `math`, т.е. до вычисления площади. Её можно, например, указать после инструкции с заданием величины `r`. Однако, по многим соображениям команды `import` принято помещать в самое начало программы.

2. Запустить программу и убедиться в её работоспособности.
3. Добавить перед инструкцией `r=5` один или несколько пробелов. Сохранить и выполнить файл. Какое сообщение об ошибке получено?
4. Восстановить работоспособность программы. Добавить в программу строки, позволяющие вычислить длину окружности. Для этого скопировать и добавить в конец программы строки с вычисление площади и выводом результат. Отредактировать инструкции:

- переменную для хранения длины окружности назвать `circumference` (англ. дина окружности),
- отредактировать формулу, заменив вычисление площади вычислением длины окружности,
- изменить текст "`Площадь круга`" на "`Длина окружности`" и заменить переменную `area` в инструкции `print`.

5. Проверить работу программы.

Вопросы для самоконтроля

1. Что понимается под интерактивным режимом работы пользователя? Когда он применяется?
2. В чем суть работы по схеме `REPL`?

3. Что понимается под сценарием?

4. В чем состоит суть сценарного режима работы?

5. Какими способами можно запустить интерпретатор Питона?

6. Как запустить интерпретатор Питона в режиме текстовой консоли? Как в этом режимы можно повторно выполнить команду? Как выйти из интерпретатора.

7. Каким образом можно пользоваться утилитой `help`?

8. Как выполнить программу в сценарном режиме из командной строки операционной системы?

9. Для чего нужны интегрированные среды разработки? Какие основные функции они реализуют?

10. Что понимается под подсветкой синтаксиса? В чем польза подсветки?

11. Вывод каких данных выполняется интерпретатором автоматически?

12. Как повторно выполнить команду в `IDLE`?

13. Для чего может потребоваться перезапуск интерпретатора? Как перезапустить интерпретатор?

14. Каким образом в `IDLE` происходит выполнение программ в сценарном режиме?

15. Какого типа сообщение появляется в том случае, если в строке перед инструкцией добавлены лишние пробелы?

Литература для изучения языка Питон

Есть много книг, статей и интернет-источников по языку Питон. Но относиться к ним нужно внимательно, так как большая часть относится к второй версии языка.

По характерным особенностям программного кода вторую версию определить несложно. Например,

- `print` применяется без скобок, а в третьей версии они обязательны,
- упоминается тип целых чисел `long`, который в третьей версии не поддерживается,
- для ввода информации с клавиатуры используется конструкция **`raw_`**`input`, которая из третьей версии исключена.

Книги

1. Лутц М. Изучаем Python. – СПб.: Символ-Плюс, 2011.

2. Доусон М. Программируем на Python – СПб.: Питер, 2014.

3. Сузи Р. А. Python. – СПб.: БХВ-Петербург, 2002.

4. Бизли Д. Подробный справочник. – СПб.: Символ-Плюс, 2010.

5. Саммерфилд М. Программирование на Python 3. СПб.: Символ-Плюс, 2009

Интернет-источники

1. Python 3.4.3 documentation // Python
https://docs.python.org/3/

2. Python/Учебник Python 3.1 // Викиучебник
https://ru.wikibooks.org/wiki/Python/Учебник_Python_3.1

3. Язык программирования Python //Интуит
http://www.intuit.ru/studies/courses/49/49/info#

4. Основы программирования // Интуит
http://www.intuit.ru/studies/courses/2193/67/info

5. Учебник по языку программирования Python (хабраиндекс) // Хабрахабр
http://habrahabr.ru/post/61905/