**Простейшие программы**

Программы на языке Python чаще всего выполняются *интерпретатором*, который читает очередную команду и сразу её выполняет, не переводя всю программу в машинный код конкретного процессора. Можно работать в двух режимах:

* через командную строку (в *интерактивном* режиме), когда каждая введённая команда сразу выполняется;
* в программном режиме, когда программа сначала записывается в файл (обычно имеющий расширение **.py**), и при запуске выполняется целиком; такая программа на Python называется *скриптом* (от англ. *script = сценарий*).

Мы будем говорить, главным образом, о программном режиме.

Пустая программа – это программа, которая ничего не делает, но удовлетворяет требованиям выбранного языка программирования. Пустая программа на Pythоn (в отличие от многих других языков программирования) – действительно пустая, она не содержит ни одного *оператора* (команды). Можно добавить в программу *комментарий* – пояснение, которое не обрабатывается транслятором:

**# Это пустая программа**

Как вы уже увидели, комментарий начинается знаком #. Если в программе используются русские буквы, в самом начале обычно записывают специальный комментарий, определяющий кодировку:

**# -\*- coding: utf-8 -\*-**

В данном случае указана кодировка UTF-8. В Python 3 она установлена по умолчанию, поэтому эту

строчку можно не писать.

Напишем программу, которая выводит на экран такие строки:

**2+2=?**

**Ответ: 4**

Вот как она выглядит:

**print ("2+2=?")**

**print ("Ответ: 4")**

Команда **print** выводит на экран символы, заключенные в апострофы или в кавычки. После выполнения той команды происходит автоматический переход на новую строчку.

**Переменные**

Напишем программу, которая выполняет сложение двух чисел:

1) запрашивает у пользователя два целых числа;

2) складывает их;

3) выводит результат сложения.

Программу на псевдокоде (смеси русского языка и Python) можно записать так:

**ввести два числа**

**сложить их**

**вывести результат**

Компьютер не может выполнить псевдокод, потому что команд «ввести два числа» и ей подоб-ных нет в его системе команд. Поэтому нам нужно «расшифровать» все такие команды, выразивих через операторы языка программирования.В отличие от предыдущей задачи, данные нужно хранить в памяти. Для этого используют*переменные*.

**Переменная** — это величина, которая имеет имя, тип и значение. Значение переменной можетизменяться во время выполнения программы.

Переменная (как и любая ячейка памяти) может хранить только одно значение. При записи в неё нового значения «старое» стирается и его уже никак не восстановить. В языке Python (в отличие от многих других языков) переменные не нужно предварительно объявлять. Они создаются в памяти при первом использовании, точнее, при первом присваивании им значения. Например, при выполнении оператора присваивания

a=**4**

в памяти создается новая переменная (объект типа «целое число») и она связывается с именем **a**.По этому имени теперь можно будет обращаться к переменной: считывать и изменять её значение. В именах переменных можно использовать латинские буквы (строчные и заглавные буквы различаются), цифры (но имя не может начинаться с цифры, иначе транслятору будет сложно различить, где начинается имя, а где – число) и знак подчеркивания «\_».

Желательно давать переменным «говорящие» имена, чтобы можно было сразу понять, какую роль выполняет та или иная переменная.

Тип переменной в Python определяется автоматически. Программа

**a = 4**

**print (type(a))**

выдаёт на экран тип (англ. *type*) переменной **a**:

**<class 'int'>**

В данном случае переменная **a** целого типа, на это указывает слово **int** (сокращение от англ. *integer*

– целый). Говорят, что переменная **a** относится к классу **int**.

Тип переменной нужен для того, чтобы

* определить область допустимых значений переменной;
* определить допустимые операции с переменной;
* определить, какой объем памяти нужно выделить переменной и в каком формате будут храниться данные (целые и вещественные числа хранятся по разному).

В языке Python используется *динамическая типизация*, это значит, что тип переменной определяется по значению, которое ей присваивается (а не при объявлении переменной, как с языках Паскаль и C). Таким образом, в разных частях программы одна и та же переменная может хранить значения разных типов3.

Вспомним, что нам нужно решить три подзадачи:

* ввести два числа с клавиатуры и записать их в переменные (назовём их **a** и **b**);
* сложить эти два числа и записать результат в третью переменную **c**;
* вывести значение переменной **c** на экран.

Для ввода используется команда4 **input**, результат работы которой можно записать в переменную, например, так:

**a = input()**

При выполнении этой строки система будет ожидать ввода с клавиатуры и, когда пользователь введёт число и нажмёт клавишу *Enter*, запишет это значение в переменную **a**. При вызове функции **input** в скобках можно записать сообщение-подсказку:

**a = input ( "Введите целое число: " )**

Сложить два значения и записать результат в переменную **c** очень просто:

**c = a + b**

Символ «**=**» – это **оператор присваивания**, с его помощью изменяют значение переменной. Он выполняется следующим образом: вычисляется выражение справа от символа «**=**», а затем результат записывается в переменную, записанную слева. Поэтому, например, оператор

**i = i + 1**

увеличивает значение переменной **i** на 1.

Вывести значение переменной c на экран с помощью уже знакомой функции **print**:

**print (c)**

Казалось бы, теперь легко собрать всю программу:

**a = input()**

**b = input()**

**c = a + b**

**print (c)**

однако, после того, как мы запустим её и введём какие-то числа, допустим 21 и 33, мы увидим странный ответ 2133. Вспомните, что при нажатии клавиши на клавиатуре в компьютер поступаетеё код, то есть код соответствующего символа. И входные данные воспринимаются функцией **input** именно как поток символов. Поэтому в той программе, которая приведена выше, пере-менные **a** и **b** – это цепочки символов, при сложении этих цепочек (с помощью оператора «+») программа просто объединяет их – приписывает вторую цепочку в конец первой.

Чтобы исправить эту ошибку, нужно преобразовать символьную строку, которая полученапри вводе, в целое число. Это делается с помощью функции **int** (от англ. *integer* – целый):

**a = int ( input() )**

**b = int ( input() )**

Возможен еще один вариант: оба числа вводятся не в разных строках, а в одной строке через пробел. В этом случае ввод выполняется сложнее:

**a, b = map ( int, input().split() )**

В этой строке собрано сразу несколько достаточно сложных операций:

**input()** возвращает строку, которая введена с клавиатуры;

**input().split()** – эта строка разрезается на части по пробелам; в результате получается набор значений (*список*);

**map()**применяет какую-то операцию ко всем элементам списка; в нашем случае это функция **int()**, которая превращает строку в целое число.

В результате после работы функции **map** мы получаем новый список, состоящий уже из чисел.

Первое введённое число (первый элемент списка) записывается в переменную **a**, а второе – в переменную **b**.

Итак, после того, как мы преобразовали введённые значения в формат целых чисел, программа работает правильно – складывает два числа, введённые с клавиатуры. Однако у неё есть два недостатка:

1) перед вводом данных пользователь не знает, что от него требуется (сколько чисел нужно вводить и каких);

2) результат выдается в виде числа, которое означает неизвестно что.

Хотелось бы, чтобы диалог программы с пользователем выглядел так:

Введите два целых числа:

**2**

**3**

**2+3=5**

Подсказку для ввода вы можете сделать самостоятельно. При выводе результата ситуация несколько усложняется, потому что нужно вывести значения трёх переменных и два символа: «+» и «=». Для этого строится список вывода, элементы в котором разделены запятыми:

**print(a,"+", b,"=",c )**

Мы почти получили нужный результат, но почему-то знаки «+» и «=» отделены лишними пробелами, который мы «не заказывали»:

**2 + 3 = 5**

Действительно, функция **print** вставляет между выводимыми значениями так называемый *разделитель* (или сепаратор, англ*. separator*). По умолчанию разделитель – это пробел, но мы можем его изменить, указав новый разделитель после слова sep:

**print(a,"+", b,"=",c, sep="")**

Здесь мы установили пустой разделитель (пустую строку). Теперь все работает как надо, лишних пробелов нет.

В принципе, можно было бы обойтись и без переменной c, потому что элементом списка вывода может быть арифметическое выражение, которое сразу вычисляется и на экран выводится результат:

**print(a,"+", b,"=",a+b, sep ="")**

В Python можно использовать *форматный вывод*: указать общее количество знакомест, отводимое на число. Например, программа

**a = 123**

**print("{:5d}".format(a) )**

выведет значение целой переменной a, заняв ровно 5 знакомест:

**◦◦123**

Поскольку само число занимает только 3 знакоместа, перед ним выводятся два пробела, которые обозначены как «◦». Фигурные скобки в строке перед словом **format** показывают место, где будет выведено значение, записанное в скобках после этого слова (это аргумент функции **format** –данные, которые ей передаются). Запись «:5d» означает, что нужно вывести целое число в десятичной системе счисления (англ. *decimal –* десятичный) в пяти позициях. Можно выводить сразу несколько значений, например, так

**a = 5**

**print ("{:5d}{:5d}{:5d}".format (a, a\*a, a\*a\*a))**

Значения, которые должна вывести функция **format**, перечислены в скобках через запятую. Результат будет такой:

**◦◦◦◦5◦◦◦25◦◦125**