1 Введение в программирование

1.1 Ввод/вывод и арифметические операции с целыми числами

1.1.1 Первая программа

Начнем с написания программы «Hello, World!» или «Здравствуй, Мир!». Эта программа выводит на экран приветствие миру и является простым способом заставить компьютер выполнить нашу команду.

```
print("Hello World")
```

Функция print() в языке Питон предназначена для вывода заданных объектов на стандартное устройство вывода — обычно экран, также может отправлять их в файл.

1.1.2 Комментарии

Добавление комментариев считается хорошей практикой. Это неисполняемые, но все равно важные части кода. Они не только помогают программистам, работающим над одним и тем же проектом, но также тестировщикам, которые могут обращаться к ним для ясности при тестировании белого ящика.

Комментарии — это способ выражения того, что делает программа на самом высоком уровне. Это отмеченные строчки, которые комментируют код. В Python они бывают двух типов: одно- и многострочные.

Однострочные комментарии нужны для написания простых, быстрых комментариев во время отладки. Такие комментарии начинаются с символа решетки # и автоматически заканчиваются символом окончания строки (EOL).

```
# Пример комментария
print("Hello World")
```

Python позволяет писать комментарии на нескольких строках. Они называются многострочными или блочными. Такой тип комментирования подходит для описания чего-то более сложного.

В Python есть такая особенность, как задокументированные строки (docstring). С их помощью программисты могут быстро добавлять комментарии для каждого модуля, функции, метода или класса.

Задать docstring в Python можно с помощью тройных кавычек. Нужно добавить один набор в начале и еще один – в конце. Docstring также могут занимать по несколько строк.

```
    def theFunction():
        '''
        Эта функция демонстрирует использование docstring
в Python.
        '''
        print("docstring     python     не     являются
комментариями.")
```

Комментарии и docstring добавляют ценности программе. Они делают программы более читаемыми и пригодными для последующей поддержки.

1.1.3 Переменные и типы данных

При работе с программой пользователи вводят какие-то данные (с клавиатуры, через элементы управления, с помощью файлов), а программа выводит какие-то данные (на экран, на принтер, записывает в файл), поэтому в языках программирования предусмотрены средства, позволяющие хранить и обрабатывать данные в процессе работы программы. Такими средствами являются переменные.

Переменная – это именованная ячейка памяти определенного размера, определяемого типом этой переменной. Переменную можно мыслить как некоторый контейнер для размещения чисел, символов, строк и пр. Тип – это множество значений, которые может принимать переменная.

Типы данных в Python:

- Numbers (числа)
    - Целые числа (int)
    - Вещественные числа (float)
    - Комплексные числа (complex)
- Strings (строки)
- Lists (списки)
- Dictionaries (словари)
- Tuples (кортежи)
- Sets (множества)
- Boolean (логический тип данных)

Эти типы данных можно, в свою очередь, классифицировать по нескольким признакам:

- изменяемые (списки, словари и множества)
- неизменяемые (числа, строки и кортежи)
- упорядоченные (списки, кортежи, строки и словари)
- неупорядоченные (множества)

Присвоение значения переменной – это занесение значения в ячейку памяти, выделенную под эту переменную, при помощи оператора =.

```
number = 10;
length = 2.5;
name = "Alex"
```

1.1.4 Вывод информации на экран

Рассмотрим синтаксис функции print(). Самый простой пример:

```
print()
```

Даже если функция не получает никаких аргументов, все равно необходимо вставлять после названия пустые скобки, что значит для интерпретатора выполнить функцию, а не просто ссылаться на нее.

В результате этого будет выведен неотображаемый символ пустой строки, она появиться на экране, не нужно путать это с пустой строкой, в которой вообще нет никаких символов.

Но чаще всего нужно передать какое-то сообщение пользователю, к примеру:

```
print('Ваш текст')
```

Как было сказано, print() можно вызывать без всяких аргументов при необходимости создать пустую строку, или указывать один аргумент для вывода сообщения. Кроме того, функция может принимать любое количество позиционных аргументов, разделяя их запятой, что очень удобно при необходимости объединения нескольких элементов.

Полная версия print выглядит так:

```
print(object1, object2, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

- *objects — объект/объекты которые необходимо вывести;
- sep — разделитель между объектами. В качестве своего значения можно передавать строку или None (по умолчанию пробел " ");
- end — символ в конце строки (по умолчанию перенос строки \n);
- file — file-like объект [поток] (по умолчанию sys.stdout);
- flush — принудительный сброс потока [работает с версии Python 3.3] (по умолчанию False).

Примеры выводов:

```
>>> message = 'Hello world'
>>> print(message)
Hello world
```

```
>>> print('one', 'two', 'three')  # str
one two three

>>> print(42)  # int
42
```

Пример использования параметра sep:

```
>>> print('home', 'user', 'documents', sep='/')
home/user/documents
```

Второй необязательный параметр — end. Он позволяет предотвратить разрыв строки, когда выведенное сообщение не должно заканчиваться символом новой строки.

```
>>> print('The first sentence', end='. ')
>>> print('The second sentence', end='. ')
>>> print('The last sentence.')
The first sentence. The second sentence. The last sentence.
```

При необходимости можно указывать одновременно два ключевых аргумента:

```
>>> print('Mercury', 'Venus', 'Earth', sep=', ', end='!')
Mercury, Venus, Earth!
```

В чем разница между одинарными и двойными кавычками в Python? Вот сводная таблица, отвечающая на вопрос, следует ли использовать одинарные или двойные кавычки в Python:

	Single quotes	Double quotes
Represented with	‘ ’	“ ”
Primary use case	Identifiers and string literals	Text, string interpolation, quotations
Example	`name = ‘Bob’`	`txt = “Weather is sunny today in California.”`

Если коротко, то вы можете использовать обе кавычки во всех случаях, но двойные кавычки чаще используются с текстом и длинными строками. Никто не запрещает вам использовать одинарные кавычки везде, но вам придется быть более осторожным, поскольку одинарные кавычки более чувствительны к определенным символам.

1.1.5 Ввод информации с клавиатуры

За ввод в программу данных с клавиатуры в Python отвечает функция input. Когда вызывается эта функция, программа останавливает свое выполнение и ждет, когда пользователь введет текст. После этого, когда он нажмет Enter, функция input() заберет введенный текст и передаст его программе, которая уже будет обрабатывать его согласно своим алгоритмам.

Функция input() передает введенные данные в программу. Их можно присвоить переменной.

```
answer = input()
```

Обратите внимание, что в программу поступает строка. Даже если ввести число, функция input() все равно вернет его строковое представление. Но что делать, если надо получить число? Ответ: использовать функции преобразования типов.

```
age = int(input())
price = float(input())
```

1.1.6 Библиотеки Python

Библиотеки (или модули) Python — это файлы с шаблонами кода. Их придумали для того, чтобы людям не приходилось каждый раз заново набирать один и тот же код: они просто открывают файл, вставляют свои данные и получают нужный результат. Пример подключения библиотеки:

```
import math
```

Модуль math – один из наиважнейших в Python. Этот модуль предоставляет обширный функционал для работы с числами.

Для выполнения первой работы вам могут пригодиться две функции из модуля math:

- pow(x,y) – возвращает результат возведения числа в степень;
- abs(x) – возвращает модуль числа.

1.1.7 Основные арифметические операции

Операция	Описание	Пример
+	Сложение	z = x+y
-	Вычитание	z = x-y
-	Изменение знака	z = -x
*	Умножение	z = x*y
**	Возведение в степень	Z = x ** y
//	Деление нацело	z = x//y
/	Деление (хотя бы один вещественный)	z = x/y
%	Остаток от деления, mod (только для целых)	z = x%y