Основы алгоритмизации и программирования

Пушкарев Александр Николаевич
к.т.н., преподаватель
ГАПОУ ТО «Колледж цифровых и педагогических технологий»

Литература

1. Бизли Д. Python. Подробный справочник. – СПб.: Символ-Плюс, 2010. – 864 с.
2. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
3. Лутц М. Изучаем Python. – СПб.: Символ-Плюс, 2011. – 1280 с.
4. Лутц М. Программирование на Python. Том 1. – СПб.: Символ-Плюс, 2011. – 992 с.
5. Лутц М. Программирование на Python. Том 2. – СПб.: Символ-Плюс, 2011. – 992 с.
6. Россум Г., Дрейк Ф.Л.Дж., Откидач Д.С. Язык программирования Python. – 2001. – 454 с.
7. Саммерфилд М. Программирование на Python 3. Подробное руководство. – СПб.: Символ-Плюс, 2009. – 608 с.
8. Сузи Р.А. Язык программирования Python (курс лекций). – М.: Бином-пресс, 2006. – 326 с.
9. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. – 126 с.

Условный оператор if
Цикл for
Цикл while

Переменные

- Для хранения промежуточных и конечных результатов работы программы в языке Python предусмотрена возможность использования переменных.
- Переменная – это ссылка на данные, имеющая заданное программистом имя (например, MyName, data, i, variable3 и т.п.).
- Чтобы связать переменную с данными, используется операция присваивания – справа от переменной пишется знак равенства и указывается значение, на которое переменная будет ссылаться:

  welcome = 'Hello, world!'

- После этого к данным можно будет обратиться по имени переменной: welcome
- В результате на экране появится строка: 'Hello, world!'

Вывод сообщения

- Написание программ на языке Python, как и на любом другом языке программирования, предполагает возможность оповещения пользователя. Для вывода программой сообщений в окно консоли (окно, в котором вводятся команды и выводятся результаты их выполнения) используется команда **print**. В скобках после этой команды следует указать выводимое значение или текст, оформленный в двойные кавычки.

- Пример вывода текстового сообщения:
  print("Hello, world!"**)**

- В результате на экран консоли будет выведено сообщение:
  Hello, world!

Вывод сообщения

- Команда **print** позволяет выводить на экран консоли не только заранее определённый текст, но и результаты произведённых программой вычислений. Для этого в скобках следует указать имя переменной, содержащей данные результаты.

- Пример вывода на экран консоли факториала числа **5** (равен 120):

```
i = 1 * 2 * 3 * 4 * 5
print(i)
```

- Сообщение можно формировать из частей, разделяемых запятыми:

```
i = 1 * 2 * 3 * 4 * 5
print("Факториал числа 5 равен ", i)
```

Комментарий в языке Python

- Комментарии используются для вставки в код пояснений к фрагментам программы. Такие пояснения обращены к человеку, читающему код, а не к обрабатывающей данный код ЭВМ.

  Знак **#** превращает в комментарий все символы после него до конца текущей строки.

- Пример использования комментария:

  Действие1

  Действие2

  # *Действие3* **не будет выполнено**

  Действие4 **#** **а** *Действие4* **– будет**

Операции в языке Python

- В языке Python доступны следующие математические операции.

Операция	Запись	Пример
Сложение чисел x и y	x + y	5 + 2 **# = 7**
Вычитание числа x из числа y	x – y	5 – 2 **# = 3**
Умножение числа x на число y	x * y	5 * 2 **# = 10**
Деление числа x на число y	x / y	5 / 2 **# = 2.5**
Целочисленное деление (целая часть частного от деления)	x // y	5 // 2 **# = 2**
Остаток от деления (число, которое не поделилось нацело)	x % y	5 % 2 **# = 1**
Возведения числа x в степень числа y	x ** y	5 ** 2 **# = 25**
Смена знака числа x	-x	-5 **# = -5**

Логические выражения

- Язык Python поддерживает логические выражения, каждое из которых может являться истинным (**True**) или ложным (**False**).

Логическое выражение	Запись	Пример
Число x больше числа y	x > y	5 > 2 **# = True**
Число x меньше числа y	x < y	5 < 2 **# = False**
Число x неменьше (больше или равно) числа y	x >= y	5 >= 5 **# = True**
Число x небольше (меньше или равно) числа y	x <= y	5 <= 5 **# = True**
Число x равно числу y	x == y	5 == 2 **# = False**
Число x неравно числу y	x != y	5 != 2 **# = True**
Число x меньше числа y и больше числа z	y > x > z	8 > 5 > 2 **# = True**
Число x больше числа y и меньше числа z	y < x < z	8 < 5 < 2 **# = False**

Логические операции

- Для работы с логическими выражениями в языке Python применяются логические операции, результатом каждой из которых может быть одно из значений **True** (Истина) или **False** (Ложь), именуемых логическими константами (постоянными величинами).

Логическая операция	Запись	Пример
Логическое умножение (требует истинности обоих условий)	x and y	True and True **# = True** True and False **# = False**
Логическое сложение (требует истинности хотя бы одного условия)	x or y	True or True **# = True** True or False **# = True**
Логическое отрицание (условие меняется на противоположное)	not x	not True **# = False** not False **# = True**
Принадлежность x множеству Y	x in Y	5 in [1, 2, 3, 4, 5] **# = True** 8 in [1, 2, 3, 4, 5] **# = False**

Ветвление

- Ветвление – выполнение одного из возможных действий (или ряда действий), выбор которого определяется истинностью заданного условия или значением заданного выражения.

- В Python условным оператором является оператор **if** (в переводе с английского означает «если»).

Инструкция if

- Инструкция **if** в простейшей своей форме организует выполнение действия по результатам проверки некоторого условия:

  if *Условие* **:**
  Действие

- Пример:

```
if 5 > 2 :
  print("Выражение верно")
```

Инструкция if

- В случае, если условие не выполняется, может быть выполнено альтернативное действие, указываемое после ключевого слова **else** (в переводе с английского – «иначе») :

```
if Условие :
  Действие1
else :
  Действие2
```

- Пример:

```
if 5 > 2 :
   print("Выражение верно")
else :
   print("Выражение неверно")
```

Инструкция if

- Следует заметить, что влияние на работу программы оказывают отступы, выставляемые перед строками кода. Чтобы установить отступ, следует перед строкой кода поставить символ табуляции (клавиша Tab). По результатам проверки условия может быть выполнено как одно действие, так и несколько.

```
if Условие :
  Действие11
  Действие12
  …
  Действие1N
else :
  Действие21
  Действие22
  …
  Действие2N
```

Инструкция if

- Пример выполнения блока действий в каждой из веток ветвления:

```
if 5 > 2 :
  print("Выражение верно")
  print("Пять больше двух")
else :
  print("Выражение неверно")
  print("Пять не больше двух")
```

Инструкция if

- Инструкция **if** позволяет организовывать и более сложные ветвления.

  Например, можно организовать вложенную проверку условия.

```
if Условие1 :
  Действие1
else :
  if Условие2 :
       Действие2
  else :
       Действие3
```

Инструкция if

- Пример создания вложенного ветвления:

```
if 5 > 2 :
  print("Пять больше двух")
else :
  if 5 < 2 :
        print("Пять меньше двух")
  else :
        print("Пять равно двум")
```

Инструкция if

- Однако для решения данной задачи можно воспользоваться ключевым словом **elif** (от английского «else if» – «иначе если»), позволяющим упростить оформление вложенности.

```
if Условие1 :
  Действие1
elif Условие2 :
  Действие2
else :
  Действие3
```

Инструкция if

- Пример создания вложенного ветвления с использованием ключевого слова **elif**:

```
if 5 > 2 :
  print("Пять больше двух")
elif 5 < 2 :
  print("Пять меньше двух")
else :
  print("Пять равно двум")
```

- Следует заметить, что количество веток, начинающихся с **elif**, неограниченно, как и количество вложенных друг в друга ветвлений.

Циклы

- Цикл – выполнение определенное количество раз некоторого действия (последовательности действий).

- В Python основными циклами являются:

  – цикл **while**;

  – цикл **for**.

Цикл while

- Цикл **while** организует непрерывное выполнение действия, пока выполняется заданное условие.

```
while Условие :
  Действие
```

- Пример использования цикла **while** для определения наибольшего целочисленного делителя числа **72**, отличного от него самого:

```
n = 71

while 72 % n != 0 :
  n = n – 1

print(n)            # 36
```

Цикл while

- Цикл **while** также позволяет организовать выполнение блока действий за одну свою итерацию (прохождение цикла).

```
while Условие :
 Действие1
 Действие2
 …
 ДействиеN
```

- Пример использования цикла **while** для вывода всех степеней числа **2**, не превышающих **1000**:

```
n = 1

while n < 1000 :
 print(n)
 n = n * 2          # 1 2 4 8 16 32 64 128 256 512
```

Цикл for

- Цикл **for** организует выполнение действия над каждым элементом из указанного множества.

for *ИмяПеременнойЭлемента* **in** *МножествоЭлементов* **:**
 Действие

- Пример использования цикла **for** для вывода номеров месяцев, завершающих кварталы в году:

```
for month in [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ] :
 if month % 3 == 0 :
      print(month)          # 3  6  9  12
```

Цикл for

- Цикл **for** также позволяет организовать выполнение ряда действий за одну свою итерацию.

for *ИмяПеременнойЭлемента* **in** *МножествоЭлементов* **:**
Действие1
Действие2
…
ДействиеN

Цикл for

- Пример использования цикла **for** для подсчёта количества букв и слов в предложении:

```
phrase = 'To be or not to be that is the question'
letters = 0
spaces = 1

for symbol in phrase :
 if symbol != ' ' :
      letters = letters + 1
 else :
      spaces = spaces + 1

print('В предложении ', letters, 'букв и ', spaces, 'слов.')

# В предложении 30 букв и 10 слов.
```

Инструкция break

- Инструкция **break** позволяет принудительно прервать выполнение текущего цикла и начать выполнение следующего за ним кода.

```
while Условие1 :
 Действие1
 if  Условие2 :
       break   #  В случае выполнения Условия2
               #  цикл сразу прервётся
               #  и следующим будет выполнено Действие0
 …
 ДействиеN

Действие0
```

Инструкция break

- Пример использования инструкции **break** для определения наибольшего целочисленного делителя числа **72**, отличного от него самого :

```
n = 71

while n > 0 :
 if 72 % n == 0 :
      break
 n = n – 1

print(n)            # 36
```

Инструкция continue

- Инструкция **continue** позволяет принудительно начать выполнение цикла с новой итерации.

```
while Условие1 :
 Действие1
 if  Условие2 :
      continue    # В случае выполнения Условия2
                  # цикл сразу начнёт новую итерацию
                  # с выполнения Действия1
 …
 ДействиеN
```

Инструкция continue

- Пример использования инструкции **continue** для вывода всех чисел от **1** до **10**, кроме **5** :

```
for el in range(1, 11) :
 if el == 5 :
      continue
 print(el)          # 1, 2, 3, 4, 6, 7, 8, 9, 10
```