Python

03

Формы оператора присваивания

```
spam = 'Spam'      # Каноническая форма
spam, ham = 'yum', 'YUM'      # Присваивание кортежей (позиционное)
[spam, ham] = ['yum', 'YUM']  # Присваивание списков (позиционное)
a, b, c, d = 'spam' # Присваивание последовательностей, обобщенное
a, *b = 'spam'              # Расширенная операция распаковывания
# последовательностей
spam = ham = 'lunch'     # Групповое присвоение одного значения
spams += 42         # Комбинированная инструкция присваивания
# (эквивалентно инструкции spams = spams + 42)
```

Ввод входных данных

```
>>> string = input('Input some lint\n')

Input some lint

line

>>> string

'line'
```

Задание 3.01

Ввести с клавиатуры имя пользователя.

Вывести на экран “Hello, [name]!”

Форматирование строк

```
firstname = input('Input your firstname: ')
lastname = input('Input your lastname: ')


string_a = 'Hello, %s %s' % (firstname, lastname)
string_b = 'Hello, {} {}'.format(firstname, lastname)
string_c = 'Hello, {0} {1}'.format(firstname, lastname)
string_d = f'Hello, {firstname} {lastname}'
```

Задание 3.02

Запросить у пользователя два целых числа.

Вывести строку вида “2 плюс 3 равно 5”

Примечание: после ввода переменных преобразовать переменные к типу int

```
>> first_number = int(first_number)
```

Дополнительные функции по работе со строками

```
string = 'Hello,_my_name_is_Alex'
string_list = string.split('_')
result = ' '.join(string_list)


string = 'ping pong'
result_string = string.replace('p', 'k')
```

Задание 3.03

Ввести предложение состоящее из двух слов. Поменять местами слова, добавить восклицательный знак в начало и конец, вывести итоговое предложение на экран.

Комментарии

Когда программы разрастаются то, их становится сложно читать. Чтобы упростить чтение отдельных частей программы в программу добавляются заметки, которые на человеческом языке рассказывают, что программа делает. Эти заметки называются комментариями и начинаются с символа #.

 # compute the percentage of the hour that has elapsed
*percentage = (minute * 100) / 60*
Комментарии можно оставлять в конце выражения:

*percentage = (minute * 100) / 60 # percentage of an hour*
Весь текст, начиная от # и до конца строки, игнорируется Python.

Комментарии призваны ответить на вопрос, почему? Они обычно содержат полезную информацию, которой НЕТ в коде.

Структурное программирование

Структурное программирование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков.

- Последовательность — однократное выполнение операций в том порядке, в котором они записаны в тексте программы.
- Ветвление — однократное выполнение одной из двух или более операций, в зависимости от выполнения заданного условия.
- Цикл — многократное исполнение одной и той же операции до тех пор, пока выполняется заданное условие (условие продолжения цикла).

Логические выражения

Логическими (boolean expression) называются выражения, которые могут принимать одно из двух значений – истина или ложь. В следующем примере используется оператор ==, который сравнивает два операнда и возвращает True, если они равны (equal) или в противном случае False:

```
>>> 5 == 5
True
>>> 5 == 6
False
```

True и False – специальные значения, которые принадлежат к типу bool

Операторы сравнения

a равно 10 и b равно 20

==	Если значения двух операнд равны - возвращает True	(a == b) - Ложь
!=	Если значения двух операнд **НЕ** равны - возвращает True	(a != b) - Истина
>	Если значение левого операнда больше значения правого - возвращает True	(a > b) - Ложь
<	Если значение правого операнда больше значения левого - возвращает True	(a < b) - Истина
>=	Если значение левого операнда больше, либо равно значению правого - возвращает True	(a >= b) - Ложь
<=	Если значение правого операнда больше, либо равно значению левого - возвращает True	(a <= b) - Истина
is	Если левый и правый операнды ссылаются на один и тот же участок памяти - возвращает True	(a is b) - Ложь
is not	Если левый и правый операнды **НЕ** ссылаются на один и тот же участок памяти - возвращает True	(a is not b) - Правда

Логические операторы

Существуют три логических оператора (logical operators): **and**, **or** и **not**. Смысл этих операторов схож с их смыслом в английском языке:

x > 0 and x < 10
это истинно в случае, если x больше 0 и меньше 10.

n%2 == 0 or n%3 == 0
это истинно, если одно из выражений является истинным.

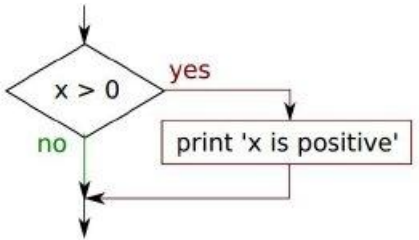
Оператор **not** отрицает логическое выражение, так
not (x > y)
истинно, если x > y является ложью, т.е. x меньше или равно y.

Условное исполнение

Для того чтобы писать полезные программы, почти всегда необходимо проверять условия и изменять поведение программы. Условные инструкции (conditional statements) предоставляют нам такую возможность. Простейшей формой является инструкция if:

```
if x > 0 :
    print('x is positive')
```

Логическое выражение после инструкции if называется условием (condition). Далее следует символ двоеточия (:) и строка (строки) с отступом. Если логическое условие истинно, тогда управление получает выражение, записанное с отступами, иначе – выражение пропускается.



Задание 3.04

Ввести предложение. Если число символов в предложении кратно 3 - добавить ! к концу строки. Вывести строку на экран.

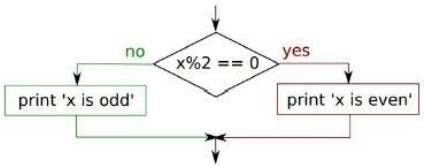
Не существует ограничения на число инструкций, которые могут встречаться в теле if, но хотя бы одна инструкция там должна быть. Иногда полезно иметь тело if без инструкций (обычно оставляют место для кода, который ещё не написан). В этом случае можно воспользоваться инструкцией **pass**, которая ничего не делает.

```
if x < 0 :
   pass # необходимо обработать отрицательные значения!
```

Альтернативное исполнение

Второй формой инструкции if является альтернативное исполнение (alternative execution), в котором существуют два направления выполнения и условие определяет, какое из них выполнится. Синтаксис выглядит следующим образом:

```
if x%2 == 0 :
        print('x is even')
else :
        print('x is odd')
```



Если остаток от деления x на 2 равен 0, то x-четное, и программа выводит сообщение об этом. Если условие ложно, то выполняется второй набор инструкций.

Так как условие может быть либо истинным, либо ложным, тогда точно выполнится один из вариантов. Варианты называются ветвями (branches), потому что они являются ответвлениями в потоке исполнения.

Оператор in

```
string = 'Hello'
if 'e' in string:
    print('YES')
else:
    print('NO')
```

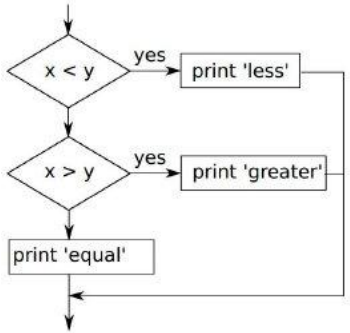
Задание 3.05

Ввести предложение. Если в предложении есть слово code - вывести на экран Yes, иначе вывести на экран No

Последовательность условий

Иногда имеется больше двух вариантов выполнения, тогда нам необходимо больше двух ветвей. В этом случае, можно воспользоваться сцепленными условиями (chained conditional):

```
if x < y:
    print('less')
elif x > y:
    print('greater')
else:
    print('equal')
```



elif является аббревиатурой от "else if". Будет исполнена точно одна ветвь.

Не существует ограничения на количество инструкций elif. Если встречается оператор else, то он должен быть в конце, но может не быть ни одного.

```
if choice == 4:
     print('Bad guess')
elif choice == 5:
    print('Good guess')
elif choice == 8:
    print('Close, but not correct')
```

Каждое условие проверяется в порядке расположения. Если первое условие ложно, то проверяется следующее и т.д. Если одно из условий истинно, то выполняется соответствующая ветка, и инструкция завершается. Даже если верно более чем одно условие, все равно выполняется только первая истинная ветка.

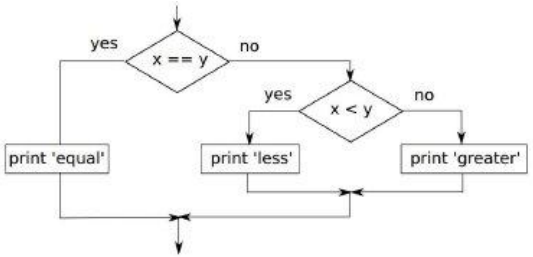
Задание 3.06

Запросить у пользователя возраст. Если возраст меньше 0 - вывести Wrong input, если меньше 18 - вывести CocaCola, иначе - вывести Beer

Вложенные условия

Одно условие может быть вложено в другое. Мы можем записать пример трихотомии (trichotomy):

```
if x == y:
    print('equal')
else:
    if x < y:
        print('less')
    else:
        print('greater')
```



Внешнее условие содержит две ветки. Первая ветка содержит простую инструкцию. Вторая ветка содержит еще одну инструкцию if, которая имеет две ветки. Эти две ветки являются простыми инструкциями, хотя они также могут содержать инструкции.

Логические операторы часто позволяют упростить вложенные условные инструкции. Например, мы можем записать следующий код, используя одно условие:

```
if x > 0:
    if x < 10:
        print ('x is a positive single-digit number.')
```

Инструкция print выполнится только, если мы зададим ее после обоих условий, также мы получим похожий эффект, используя оператор and:

```
if x > 0 and x < 10:
    print ('x is a positive single-digit number.')
```

Условные операторы

Конструкция if

```
If выражение:
    инструкция
```

Конструкция if-else

```
If выражение:
    инструкция1
else:
    инструкция2
```

Конструкция if-elif-else

```
If выражение1:
    инструкция1
elif выражение2:
    инструкция2
else:
    инструкция3
```

```
age = int(input('--> '))
```

Конструкция if

```
If age >= 18:
    print(age)
```

Конструкция if-else

```
If age >= 18:
    print('Yes')
else:
    print('No')
```

Конструкция if-elif-else

```
If age >= 18:
    print('beer')
elif age <= 12:
    print('candy')
else:
    print('orange')
```

Задание 3.07

Ввести строку с клавиатуры

Если длина строки больше 5 - вывести значение на экран

Если длина строки меньше 5 - вывести строку “Need more!”

Если длина строки равна 5 - вывести строку “It is five”

Задания 3.08

Ввести число, проверить на то, что было введено именно число. Возвести его в куб.

Задания 3.09

Вычислить квадратное уравнение $ax^2 + bx + c = 0$ (*)

$D = b^2 – 4ac;$

$x_{1,2} = (-b +/- sqrt (D)) / 2a$

Предусмотреть 3 варианта:

1) Два действительных корня
2) Один действительный корень
3) Нет действительных корней

Задания 3.10

Получить на ввод количество рублей и копеек и вывести в правильной форме, например, 3 рубля, 1 рубль 25 копеек, 3 копейки

Задания 3.11

Ввести почтовый адрес. Проверить доменной имя. В случае если оно gmail.com, вывести на экран имя почтового ящика. Иначе вывести надпись “DOMAIN NAME is not supported’