Синтаксис

Переменные

Переменные могут быть 3 типов (`number` – число, `string` – строка (текст), `bool` – логическое значение)

Переменная может хранить в себе какое-то значение. Чтобы объявить переменную нужно указать ее тип и название:

```
number n – переменная типа число (может хранить любые числа)
string s – переменная типа строка (может хранить текстовые символы)
bool b – логическая переменная (может хранить два значения true и false)
```

Записать значение в переменную можно по имени:

`n = 55` - записываем в n число 55

записать значение можно и во время объявления:

```
number n = 55
```

чтобы получить значение переменной достаточно написать ее имя:

`n + 5` 60 (в n было значение 55, добавили 5 получилось 60)

Строки пишутся в одинарных ‘ или двойных кавычках “

```
string text = ‘Hello, world!’
print(text)
```

В окне вывода будет написано Hello, world!

Арифметические операции

Сложение:

`1 + 1` 2

Вычитание:

`4 – 2` 2

Умножение:

`5 * 3` 15

Деление:

```
6 / 2                                   3
```

Остаток от деления

```
10 % 3                                  1
```

Результатом таких операций является число.

Операция «+» определена и для строк и возвращает объединенную строку

```
'Hello, ' + 'World!'                    'Hello, World'
'Number = ' + 5.35                      'Number = 5.35'
```

Операции сравнения

Операции сравнения возвращают значение типа bool

Больше:

```
6 > 3                                   true
7 > 100                                 false
```

Меньше:

```
5 < 6                                   true
100 < 1                                 false
```

Больше или равно:

```
5 >= 5                                  true
4 >= 5                                  false
```

Меньше или равно:

```
5 <= 5                                  true
5 <= 4                                  false
```

Равно:

```
5 == 5                                  true
5 == 10                                 false
```

Сравнение применимо только к числам! Кроме «==»

```
‘Hello’ == ‘Hello’        true
‘s’ == ‘ss’               false
true == true              true
false == true             false
```

В операторах также можно использовать и переменные

```
number a = 10
number b = 100

a + b                     110
a < b                     true
a == b                    false
```

Условный оператор

Конструкция вида:

```
if (условие)
{
     1
}
else
{
     2
}
```

Называется условным оператором

Условие должно иметь тип bool, если условие равно true, то выполнится код 1 если нет, то выполнится код 2, который находится после else

```
number a = 10
if (a < 100)
     print(‘a < 100’)
```

```
else
     print('else')            в результате выведется 'a < 100'
```

конструкцию if else можно использовать при записи значения в переменную:

```
number a = 10
string b = if (a == 11) 'Eleven'
           else         'Not eleven'
print(b)                 выведется 'Not eleven'
```

Иногда else можно опустить:

```
if (a == 10)
     print('a is ten')
```

если a не будет равно 10, то ничего не выведется.

Массивы

Если нам нужно хранить в переменной более одного значения, можно воспользоваться массивом. Массив может хранить произвольное количество элементов.

Для объявления массива используется []:

```
number[] numbers = number[10]      массив чисел длиной 10
string[] strings = string[30]      массив строк длиной 30
bool[] booleans = bool[24]         массив логических длиной 24
```

чтобы обратиться к элементу массива:

```
numbers[0] = -12    первый элемент имеет индекс 0, второй 1 и тд.
numbers[5] = 456
numbers[0] + numbers[5]            444
```

Задать массив можно другим способом:

```
string[] s = ['a', 'b', 'c']  массив с элементами 'a', 'b', 'c'
number[] a = [1,2,3,4,5,6]    массив чисел от 1 до 6
```

Получить длину массива можно с помощью функции size:

```
size(a)                       6
```

Циклы

Для повторения одинаковых действий используются циклы.

Repeat повторяет действие указанное количество раз:

```
repeat 100 times
     print('a')

repeat n times
{
     print('a')
     print(n)
}
```

While повторяется, пока истинно условие:

```
number i = 0
while (i < 100)
{
     print(i)
     i++
}
```

Можно сделать цикл, который будет выполняться бесконечно:

```
while (true)
```

```
{
}
```

Классический For:

```
for (number i = 0; i < 100; i++)
{
}
```

выполнится 100 раз, i изменяется от 0 до 100

Обратный For:

```
for (number i = 100; i >= 0; i--)
{
}
```

Также есть упрощенные версии цикла For

```
for number i = 0 to 100
{
}

for number i = 100 down to 0
{
}
```

В следующем примере s по очереди принимает все значения из массива:

```
for string s in [‘hello’, ‘hi’, ‘hello hi’]
{
}
```

Примеры

Объявление переменных

```
number a = 5
number b = (10 + a) * 25 / 34
string str = ‘Hello, world’
bool b = a >= b
number c = if (b > 0) b else -b
```

Объявление массивов

Заполнение массива числами от 0 до 99

```
number[] nums = number[100]

for number i = 0 to size(nums)
        nums[i] = i

string[] colors = [‘red’, ‘green’, ‘blue’]
```

Функции для работы с массивами

`size(T[] array)` – получить размер массива array

`repeatElement(T a, number n)` – массив размера n, в котором все элементы равны a

`range(number n)` – массив чисел размера n, с элементами от нуля до n-1

`range(number start, number n)` – массив чисел размера n, с элементами от start до n-1

`range(number start, number step, number n)` – массив чисел размера n, с элементами от start с шагом step

`sum(number[] array)` – сумма всех элементов массива чисел array

`min(number[] array)` – минимальное значение из массива чисел array

`max(number[] array)` – максимальное значение из массива чисел array

`concat(T[] array1, T[] array2)` – объединение двух массивов одного типа в один большой массив

`sort(T[] array)` – сортировка массива чисел по возрастанию или массива строк по алфавиту

`sortDescending(T[] array)` – сортировка массива чисел по убыванию или массива строк по алфавиту в обратном порядке

`reverse(T[] array)` – развернуть массив (последний элемент становится первым, предпоследний вторым и т.д.)

Функции управления черепашкой

setBackground(string color) – закрасить фон нужным цветом

setColor(string color) – установить цвет карандаша

setFillColor(string color) – установить цвет заливки

beginPolygon() – начать рисовать закрашенную фигуру

completePolygon() – завершить закрашенную фигуру

move(number n) – переместить черепаху вперед на n пикселей

moveTo(number x, number y) – переместить черепаху в точку (x,y)

jump(number x, number y) – переместить черепаху в точку (x,y) не оставляя след от карандаша

reset() – очистить экран и переместить черепаху в начало

rotate(number angle) – повернуть черепаху против часовой стрелки на angle

setStep(number step) – установить длительность шага

getWidth() – получить ширину экрана

getHeight() – получить высоту экрана

getPosX() – получить X координату черепахи

getPosY() – получить Y координату черепахи

setWidth(number width) – установить ширину линии