Тема урока: ввод-вывод данных

1. Вывод данных, команда `print()`
2. Ввод данных, команда `input()`
3. Решение задач

Вывод данных, команда print

я

Для вывода данных на экран используется команда `print()`.

Внутри круглых скобок пишем, что хотим вывести на экран. Если это текст, то обязательно указываем его внутри кавычек. Кавычки могут быть одинарными или двойными. До и после текста мы ставим только одинаковые кавычки.

Например, следующий код:

```
print('Мы изучаем язык Python')
```

выведет на экран текст:

```
Мы изучаем язык Python
```

То, что мы пишем в круглых скобках у команды `print()`, называется **аргументами** или **параметрами** команды.

Команда `print()` позволяет указывать несколько аргументов, в таком случае **их надо отделять запятыми**. Если вы не будете писать запятые между аргументами, Python воспримет это как синтаксическую ошибку.

Например, следующий код:

```
print('Скоро я', 'буду программировать', 'на языке', 'Python!')
```

выведет на экран текст:

```
Скоро я буду программировать на языке Python!
```

Ввод данных, команда input

Для считывания данных в языке Python используется команда `input()`.

Рассмотрим следующую программу:

```
print('Как тебя зовут?')
name = input()
print('Привет,', name)
```

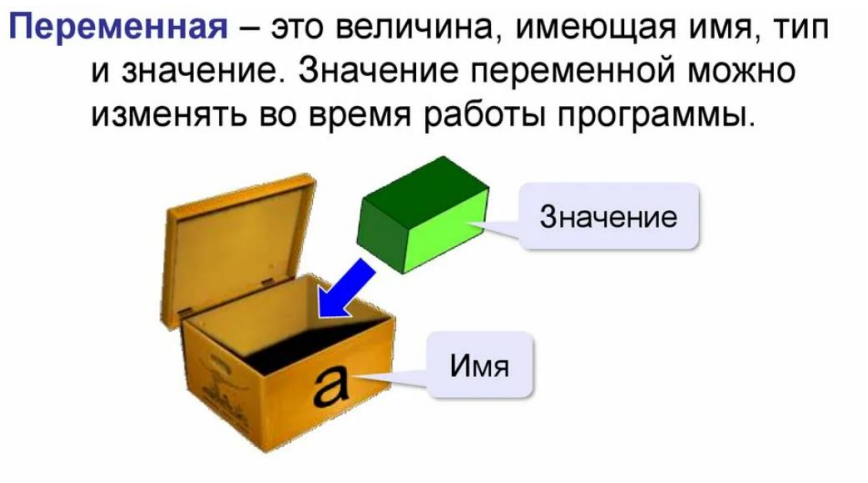
Сначала программа распечатает текст на экран «Как тебя зовут?». Далее программа будет ждать от пользователя ввода данных. Ввод данных реализуется с помощью команды `input()`.

Примечание. Очень часто перед считыванием данных мы печатаем некоторый текст, чтобы пользователь, который вводит эти данные, понимал, что именно от него требуется. Например, в программе

```
print('Как тебя зовут?')
name = input()
print('Привет,', name)
```

мы сначала выведем текст «Как тебя зовут?», а уже потом считаем данные.

Переменные



Имя переменной

1. В имени переменной используйте только латинские буквы a-z, A-Z, цифры и символ нижнего подчеркивания (_);
2. Имя переменной не может начинаться с цифры;
3. Имя переменной по возможности должно отражать её назначение.

Итак, если вы хотите, чтобы у вас была **переменная с каким-то именем и каким-то значением**, нужно написать на отдельной строчке:

```
<имя переменной> = <значение переменной>
```

Запомни: Python — регистрочувствительный язык. Переменная `name` и `Name` — две совершенно разные переменные. Для именования переменных принято использовать стиль **lower_case_with_underscores** (слова из маленьких букв с подчеркиваниями).

Тема урока: работа с целыми числами

1. Целочисленный тип данных
2. Преобразование строки к целому числу
3. Операции над целыми числами

```
number_1 = -10
number_2 = 3
#Основные операции над числами
number_3 = number_1 + number_2
number_4 = number_1 - number_2
number_5 = number_1 * number_2 #умножение
number_6 = number_1 / number_2 #математическое деление
print(number_3, number_4, number_5, number_6)
#Целочесленное деление - отбрасывает дробную часть
number_7 = number_1 // number_2
print(number_7)
#Деление с остатком
number_8 = number_1 % number_2
print(number_8)
```

Порядок выполнения операций

В математике существует порядок выполнения операций, определяющий, какие операции должны выполняться раньше других, даже если в выражении они написаны правее. Порядок выполнения операций в Python аналогичен порядку выполнения операций, которые вы изучали на уроках математики.

Python считает, что в переменных `num1` и `num2` находится **текст**, поскольку команда `input()` по умолчанию считывает именно **текст**. Для того, чтобы явно указать, что требуется работать с переменными целого типа, надо написать так:

```
num1 = int(input())
num2 = int(input())
print(num1 + num2)
```

Запомни: для того, чтобы считать одно целое число, мы пишем следующий код:

```
num = int(input())
```

Целочисленное деление

Для положительных чисел оператор целочисленного деления ведёт себя как обычное деление, за исключением того, что он отбрасывает десятичную часть результата. Рассмотрим работу данного оператора на примерах:

```
print(10 // 3)
print(10 // 4)
print(10 // 5)
print(10 // 6)
print(10 // 12)
```

Результатом выполнения такой программы будет:

```
3
2
2
1
0
```

Деление с остатком

Оператор деления с остатком возвращает остаток от деления двух целых чисел. Рассмотрим работу данного оператора на примерах:

```
print(10 % 3)
print(10 % 4)
print(10 % 5)
print(10 % 6)
print(10 % 12)
print(10 % 20)
```

Результатом выполнения такой программы будет:

```
1
2
0
4
10
10
```