Python. Числовые типы данных. Условный оператор. Логический тип.

Симченко Наталья Николаевна,
Доцент кафедры
геометрии и компьютерных наук Оренбургского государственного университета

Список источников

Книги

- Тонни Гэддис. Начинаем программировать на Python
- Н. А. Прохоренок, В. А. Дронов. Python 3. Самое необходимое
- https://pythonworld.ru/

Онлайн-курсы

- Пайтонтьютор https://pythontutor.ru/
- "Поколение Python"

https://stepik.org/course/58852/syllabus

https://stepik.org/course/68343/syllabus

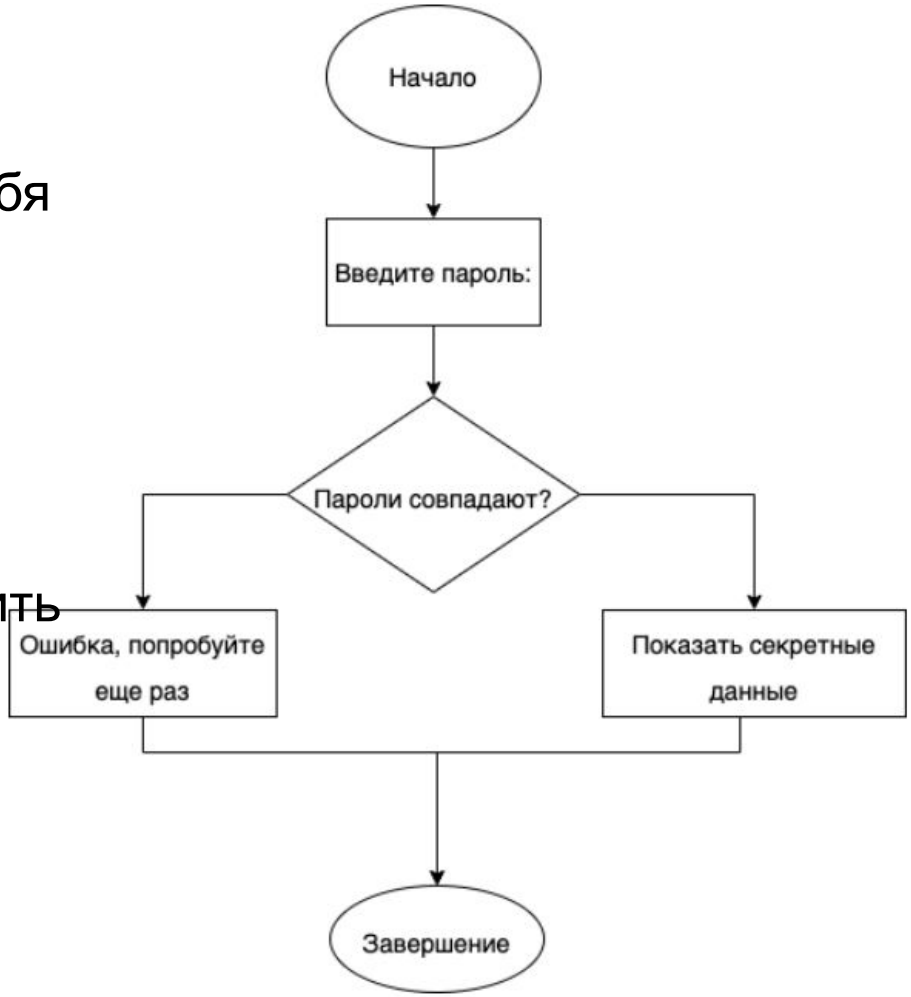
Программирование

Программирование – это деятельность по созданию программного обеспечения.

Программирование включает в себя разработку алгоритмов решения различных практических задач и их реализацию в виде компьютерных программ.

Алгоритм - понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленной цели.

Программа - это алгоритм, переведенный на понятный компьютеру язык, который будет выполнен на компьютере.



Особенности Python

- Интерпретируемый язык программирования высокого уровня
- Динамическая типизация
- Высокоуровневые структуры данных
- Поддерживает структурное, объектно-ориентированное, функциональное программирование
- Активно развивающийся
- Большое количество различных библиотек
- Области применения – анализ данных, веб-разработка, системное программирование

Среды разработки

- Интерпретатор + IDLE
https://www.python.org/

- Wing IDE,
http://wingware.com/

- PyCharm,
https://www.jetbrains.com/ru-ru/pycharm/

Язык Python

Языки программирования – это формальные языки, предназначенные для записи алгоритмов, исполнителем которых является компьютер. Алгоритмы, записанные на этих языках, называют **программами**.

Гвидо ван Россум

Одним из самых популярных современных языков программирования является Python (произносится «па́йтон» или просто «питон»). Его разработал в 1991 году нидерландский программист Гвидо ван Россум. Этот язык непрерывно совершенствуется, сейчас используется версия Python 3.

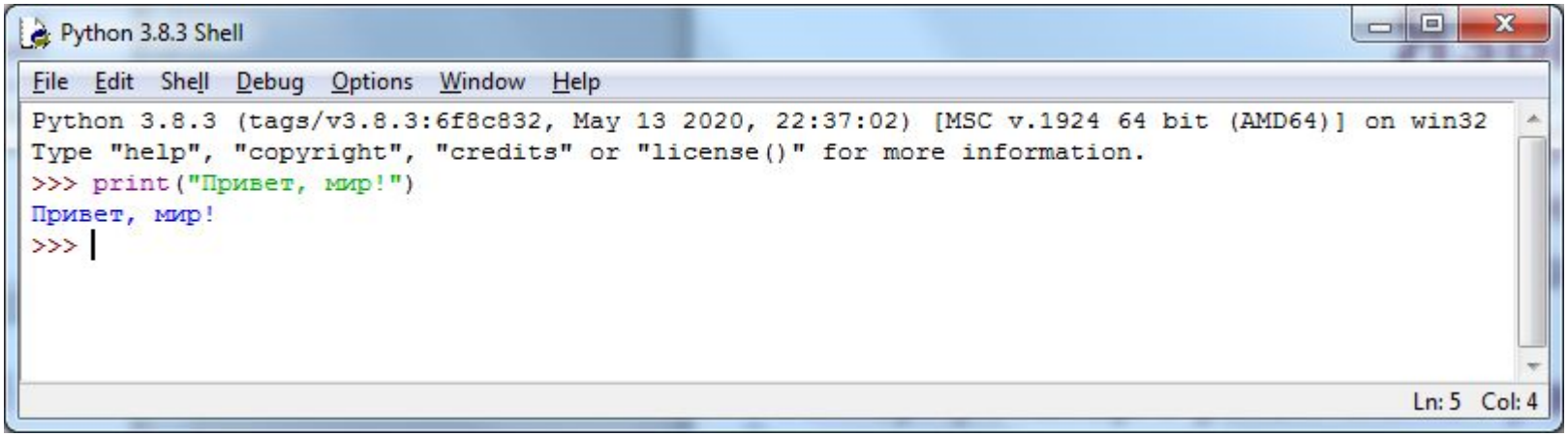
Язык Python применяется для обработки различных данных, математических вычислений, создания изображений, работы с базами данных, разработки веб-сайтов.

Язык Python

Чтобы установить Python в операционной системе Microsoft Windows, нужно скачать программу-установщик с сайта **www.python.org** .

Интерпретатор Python может работать в двух режимах:
- в командном режиме (введённая команда сразу выполняется);
- в программном режиме (программа записывается в файл с расширением **.py** и при запуске выполняется целиком).

Интерпретатор запускается в меню Пуск → Программы → Python 3.10 → IDLE. В открывшемся окне Python Shell символы **>>>** означают приглашение ввести команду, ввод команды завершается нажатием клавиши Enter. На следующей строке отобразится результат.

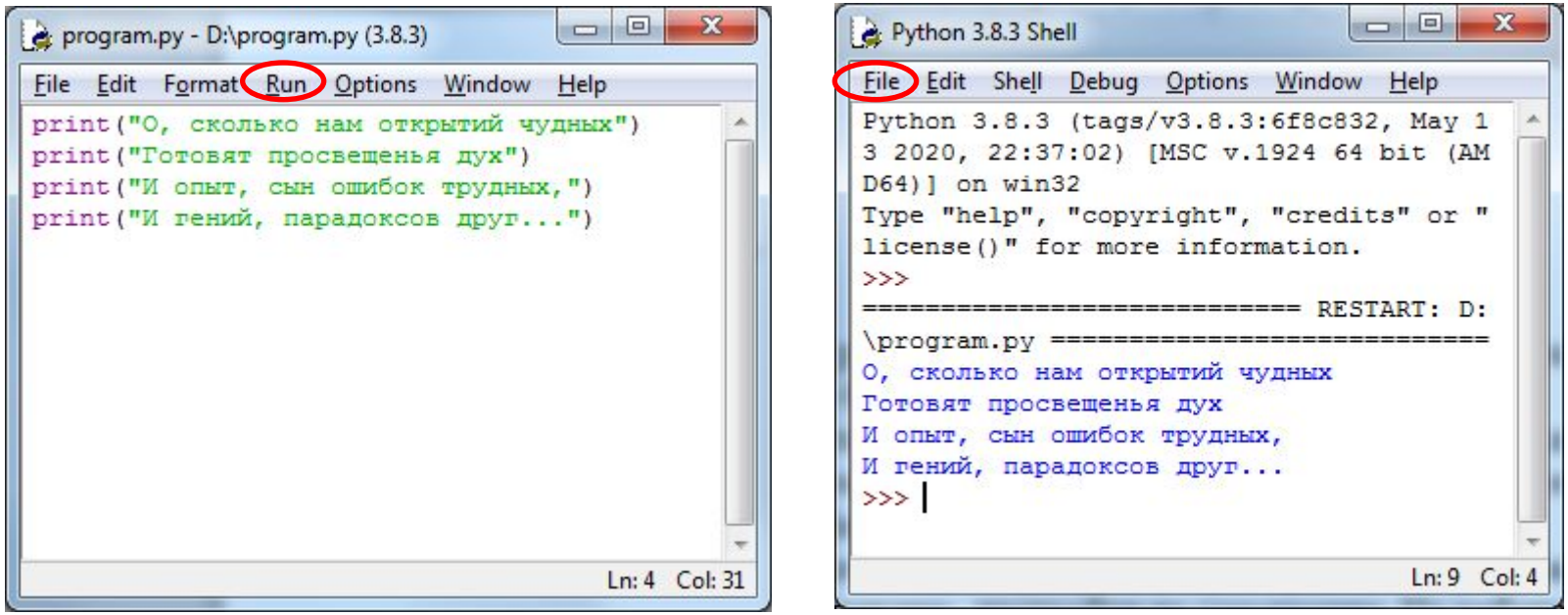


Пример работы в командном режиме

Язык Python

Для создания файла с программой в меню File нужно выбрать New File. В открывшемся окне набрать текст программы (скрипт), сохранить его под каким-нибудь именем в меню File → Save As, запустить на выполнение в меню Run → Run Module или нажав клавишу F5.

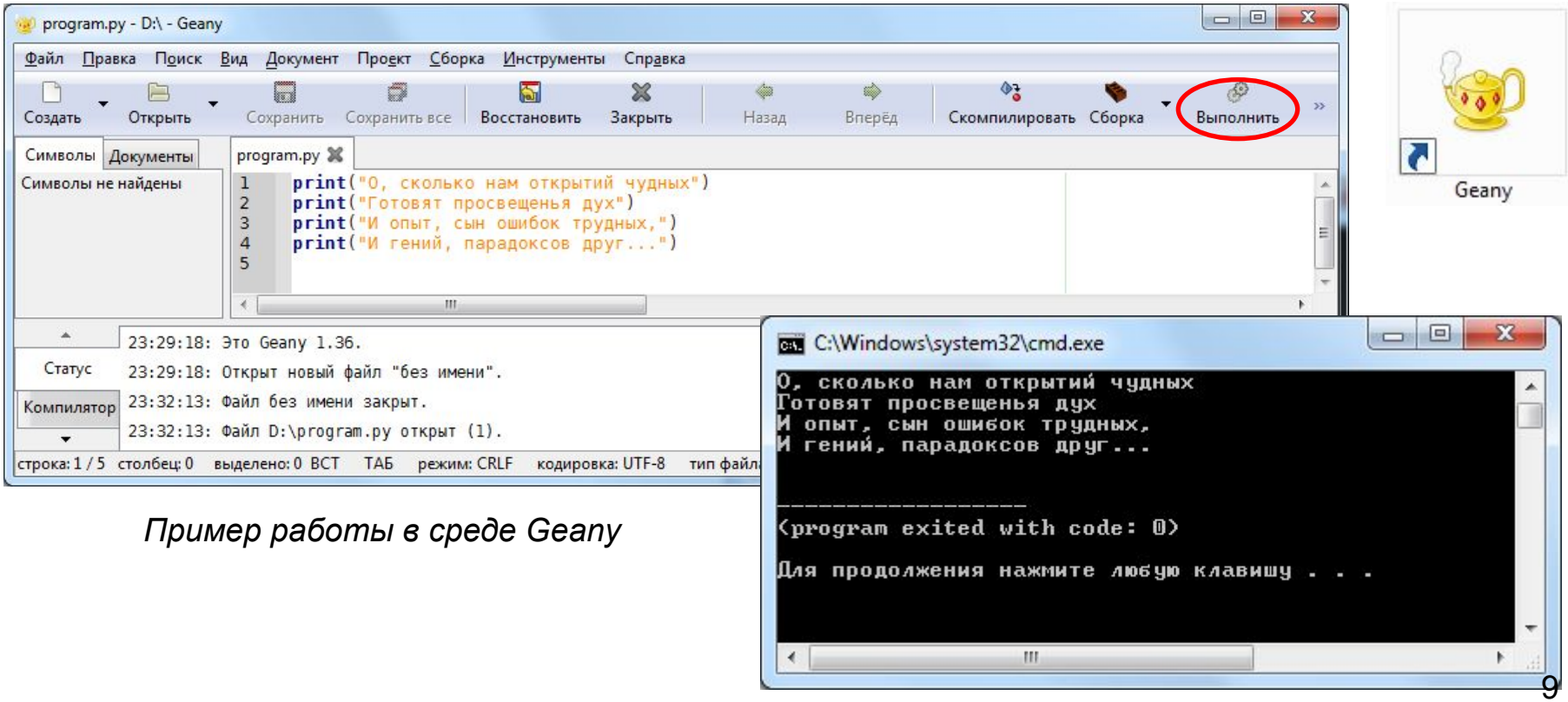
Результат работы программы отобразится в окне Python Shell.



Пример работы в программном режиме

Язык Python

Более удобной является работа в IDE – интегрированной среде разработки. Простой и русифицированной средой разработки является Geany. Её можно скачать с сайта www.geany.org. При сохранении файла расширение **.py** надо добавлять самому. Запуск программы на выполнение командой Выполнить или клавишей F5. Результат работы программы отобразится в отдельном консольном окне.



Пример работы в среде Geany

Общие сведения о языке программирования Python

Алфавит языка Python (набор допустимых символов) состоит из букв латинского алфавита (причём *заглавные и строчные буквы различаются*), цифр и специальных знаков (знаков препинания, арифметических и других). Русские буквы могут использоваться только при выводе текста на экран и в комментариях к программе.

Служебные слова – цепочки символов, имеющие фиксированное смысловое значение.

Величины в программе представлены в виде констант и переменных.

Константы – величины, не изменяющие своего значения при выполнении программы.

Переменные – величины, которые могут изменять свое значение при выполнении программы. Каждая переменная имеет имя, тип и значение.

Имя переменной (идентификатор) – любая отличная от служебных слов последовательность латинских букв, цифр и символа подчеркивания "_", не может начинаться с цифры.

N, N1, massa, massa_tela – правильно;
1N, масса, massa tela – неправильно.

Значения переменных хранятся в ячейках оперативной памяти.

Тип переменной определяет способ хранения данных в памяти компьютера и допустимые операции над ними.

Основные типы данных в языке Python

Название	Обозначение	Допустимые значения
Целочисленный	`int` *(«integer»)*	Сколь угодно большие целые числа, размер ограничен оперативной памятью
Вещественный	`float` *(«floating point»)*	Любые числа с дробной частью (с плавающей точкой)
Строковый	`str` *(«string»)*	Произвольная последовательность символов из таблицы Unicode
Логический	`bool` *(«boolean»)*	False («Ложь») или True («Истина»)

Целая часть числа от дробной отделяется точкой.

Строковое значение заключается в двойные или одинарные кавычки.

Тип переменной определяется автоматически в момент присваивания ей значения и может изменяться по ходу выполнения программы.

Выражения и операции

Выражение – это конструкция, возвращающая значение некоторого типа.

Простыми выражениями являются переменные и константы.

Сложные выражения строятся из простых с помощью операций, функций и скобок. Данные, к которым применяются операции, называются **операндами**.

Используется линейная форма записи выражений (в одну строку).

Арифметические операции

Операция	Обозначение	Пример
Сложение	+	`3 + 4 = 7`
Вычитание	-	`7 - 2 = 5`
Умножение	*	`2 * 2 = 4`
Деление	/	`8 / 2 = 4`
Целочисленное деление	//	`9 // 2 = 4`
Остаток от деления	%	`9 % 2 = 1`
Возведение в степень	**	`2 ** 3 = 8`

Выражения и операции

Логические выражения могут содержать величины или выражения, которые сравниваются между собой с помощью операций сравнения.

Логическое выражение может принимать лишь два значения: «истина» или «ложь».

Операции сравнения

Операция	Символы	Пример
равно	==	`x == 0`
не равно	!=	`x != 0`
больше	>	`x > 0`
меньше	<	`x < 0`
больше или равно	>=	`x >= 0`
меньше или равно	<=	`x <= 0`

Выражения и операции

Приоритет выполнения операций:

1) операции в скобках;
2) возведение в степень;
3) умножение и деление (в том числе // и %);
4) сложение и вычитание.

Операции одинакового приоритета выполняются в порядке записи слева направо.

Если выражение слишком длинное и не помещается в одной строке, необходимо заключить всё выражение в скобки (перенос внутри скобок разрешён).

Например:

$$\frac{(a+b)h}{2}$$

1 2 3

```
(a+b)*h/2
```

$$v+\frac{at^2}{2}$$

4 2 1 3

```
v+a*t**2/2
```

Оператор (команда) присваивания

Оператор присваивания записывает в переменную, имя которой находится слева от знака «=» значение выражения, находящегося справа. Старое значение переменной при этом стирается.

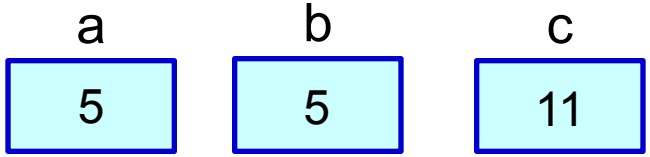
Общий вид оператора:

```
<имя переменной> = <выражение>;
```

Например:

```
a = 5
b = a
c = a+b
c = c+1
```

В памяти:



В языке Python допускается множественное присваивание:

Запись оператора:	*Равносильная запись:*
`a, b = 0, 1`	`a = 0` `b = 1`
`a = b = 0`	`a = 0` `b = 0`

Допускается запись нескольких операторов в одной строке через символ «;».

Оператор вывода

Вывод данных из оперативной памяти на экран осуществляется с помощью оператора (функции) вывода **print** («печатать»):

```
print(<выражение1>, <выражение2>, ..., <выражениеN>)
```

- На экран выводятся значения переменных и выражений, строковые значения выводится на экран без кавычек.
- Выводимые значения разделяются пробелом (по умолчанию).
- После выполнения оператора происходит автоматический переход на новую строку.

Например:

```
print ("Масса равна", m, "кг");
```

Для m=15 на экране появится:

```
Масса равна 15 кг
```

Здесь и далее символом □ обозначен пробел.

Оператор вывода

- Вместо пробела можно использовать другие символы в качестве разделителя, указав их после слова **sep** («separator»).
- Чтобы убрать переход на новую строку после выполнения оператора, используется параметр **end**.

Нужный вариант вывода	Оператор	На экране
По умолчанию	`print (1, 20, 300)`	1□20□300
Без разделителя	`print (1, 20, 300, sep="")`	120300
Через запятую и пробел	`print (1, 20, 300, sep=", ")`	1,□20,□300
Каждое с новой строки	`print (1, 20, 300, sep="\n")`	1 20 300
Без перехода на новую строку	`print (1, end="")` `print (20)`	120

Оператор вывода

Форматный вывод с помощью символьной строки позволяет задать количество позиций на экране, занимаемых выводимой величиной.

- В фигурных скобках задается формат вывода очередного элемента.
- Для целых чисел указывается количество позиций, отводимых на число.
- Если цифр в числе меньше, слева от числа выводятся пробелы.
- Для вещественного числа указывается общее количество позиций, через точку количество позиций в дробной части и буквы: **d** (целое число), **f** (вещественное) или **e** (экспоненциальный формат).

Фрагмент программы	На экране
`print ("{:3}{:3}{:3}".format(13, 7, 22))`	□13□□7□22
`a = 7` `print ("{:4d}{:4d}".format(a, a*a))`	□□□7□□49
`a = 1/3; b = 1/9` `print ("{:7.3f}{:7.4f}".format(a, b))`	□□0.333□0.1111
`a = 1/3` `print ("{:10.3e}".format(a))`	□3.333e-01

Оператор ввода

Для ввода значений переменных с клавиатуры в процессе выполнения программы используется оператор (функция) ввода **input** («ввод»):

```
<имя_переменной> = input()
```

При выполнении оператора:

- компьютер переходит в режим ожидания данных;
- пользователь вводит с клавиатуры данные в виде строки символов;
- для завершения ввода пользователь нажимает клавишу Enter;
- введенная строка записывается в указанную переменную.

Если вводится не строка, а число, необходимо выполнить преобразование типов с помощью функций **int** (для целых) и **float** (для вещественных).

Например:

```
print("Введите слово и два числа:")
x = input()
y = int(input())
z = float(input())
print(x, y, z)
```

На экране:

```
Введите слово и два числа:
ноль
1
2
ноль 1 2.0
```

Оператор ввода

Можно в скобках указать текст подсказки для пользователя.

Например:

```
x = input("Введите слово: ")
y = int(input("Введите целое число: "))
z = float(input("Введите вещественное число: "))
print (x, y, z)
```

На экране:

```
Введите слово: ноль
Введите целое число: 1
Введите вещественное число: 2
ноль 1 2.0
```

Оператор ввода

Можно в одной строке ввести несколько значений через пробел. Для этого используется функция **split** («расщепить»). Затем данные необходимо преобразовать к нужному типу по отдельности.

Например:

```
a, b, c = input("Введите a,b,c через пробел: ").split()
a, b, c = int(a), int(b), int(c)
print (a, b, c)
```

На экране:

```
Введите a,b,c через пробел: 1 2 3
1 2 3
```

Оператор комментария

Используется для включения в программу любых пояснений, предназначенных человеку.

Комментариями считается любой текст после символа **#** до конца строки. При выполнении программы комментарии игнорируются.

Пример программы:

```
# Длина окружности и площадь круга
r = float(input("Введите радиус: "))
c = 2*3.14*r    # длина окружности
s = 3.14*r**2      # площадь круга
print ("c=", "{:7.3f}".format (c))
print ("s=", "{:7.3f}".format (s))
```

На экране:

```
Введите радиус: 10
c=    62.80
s=   314.00
```

Стандартные функции

Функции имеют определенное *имя* и один или несколько *аргументов* в скобках. Функция возвращает свое значение в то место программы, из которого она вызывается.

Некоторые стандартные функции, встроенные в ядро языка Python

Функция	Назначение	Тип аргумента	Тип результата
`abs(x)`	абсолютная величина (модуль числа x)	`int, float`	как у аргумента
`int(x)`	преобразование вещественного числа к целому значению (отбрасывание дробной части)	`float`	`int`
`round(x)`	округление вещественного числа до заданного количества знаков после точки (по умолчанию – до ближайшего целого)	`float`	`int, float`

Стандартные функции

После подключения модуля к его функциям можно обращаться так же, как к встроенным. Например:

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

6 5 1 4 2 3 8 7

```
(-b + sqrt(b**2 - 4*a*c))/(2*a)
```

Можно подключать не все функции, а только необходимую. *Например:*

```
# подключаем функцию randint() из модуля random
from random import randint
```

Стандартные функции модуля `random`

Функция	Назначение	Тип аргумента	Тип результата
`random()`	случайное число из полуинтервала [0, 1)	–	`float`
`randint(a, b)`	случайное число из отрезка [a, b]	`int`	`int`

Установка Python на Windows

Программа бесплатная.

Заходим на https://www.python.org/downloads/ , выбираем "latest python release" и **python 3**.

Windows x86 MSI installer (если система 32-х битная), или

Windows x86-64 MSI installer (если система 64-х битная).

Структура программы

- **Программа** на Python – текст, содержащий последовательность команд (операторов).
- **Оператор** – предложение языка, описывающее определенное действие. Обычно каждый оператор записывается в отдельной строке программы.
- В программе могут быть также определения функций и классов, которые начинаются с ключевого слова def (будут изучаться позже).
- В операторах могут использоваться ключевые слова. **Ключевые слова** – английские слова, имеющие специальные значения. Эти слова зарезервированы и не могут использоваться в другом качестве, например в качестве имен. Среда разработки автоматически их выделяет в тексте программы.

```
>>> import keyword
>>>  keyword.kwlist
[ 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',
```

Константы

- Операторы языка работают с данными (числами, текстами, множествами и пр.), которые хранятся в памяти компьютера

- **Константы** - это данные, которые зафиксированы в тексте программы и не изменяются в процессе ее выполнения
- Примеры констант:
  - **целые** числа: 12, -23, 0b1010001
  - **действительные** числа: 1.0, -7.8
  - **логические**: True (истина), False (ложь)
  - **строковые**: "I'm study Python", 'информатика'

Переменные

- **Переменные** – это данные, которые могут изменять свои значения в ходе выполнения программы
- Переменная имеет
  - **Имя (идентификатор)** - как обращаться к переменной
  - **Значение** – что лежит в переменной
- **Оператор присваивания** устанавливает связь между именем и значением переменной

имя = выражение

- **Имя** переменной может содержать буквы, цифры, знак _ и не может начинаться с цифры

```
>>> x = 2 + 2
>>> x
4
>>> name = "Вася"
>>> name
'Вася'
>>>
```

Арифметические операции

+, -, *
/ -частное от деления
// - целая часть от деления
% - остаток от деления

```
>>> -10//3
-4
>>> -10%3
2
```

```
>>> -10//(-3)
3
>>> -10%(-3)
-1
>>> 10//(-3)
-4
>>> 10%(-3)
-2
```

```
>>> -10.5//3
-4.0
>>> -10.5%3
1.5
>>> -10.5//3.5
-3.0
>>> -10.5%3.5
0.0
```

Арифметические операции

** возведение в степень

```
>>> 3**4
81
```

Все арифметические операции можно сокращать с присваиванием: +=, -=, …

```
>>> a=5
>>> a**=2
>>> a
25
```

```
>>> b=7
>>> b*=3
>>> b
21
```

Математические функции

Модуль math нужно подключить командой import math

sqrt(x) – квадратный корень из x
fabs(x) – модуль x
sin(x), cos(x), tan(x) – тригонометрические функции
floor(x) – округление вниз
ceil(x) – округление вверх

```
>>> import math
>>> math.sqrt(16)
4.0
```

Перенос выражения

Перенос можно делать внутри скобок или с помощью знака \

```
>>> (3+5-
 7)
1
>>> 3+5\
      -7
1
>>> 3+5
8
```

Комментарии

- # комментирует весь текст до конца строки;
- ‘’’ закомментированный текст ‘’’
- “”” закомментированный текст “””

```
a=3+5#-4
""" комментарий
многострочный
"""
```

Вывод данных

- Используется функция print(вып.1, вып.2…)
- По умолчанию между выводимыми объектами ставится пробел, а в конце перевод строки

```
a = 4
b = 6
print(a,'+',b,'=')
print(a+b)
```

```
---------
4 + 6 =
10
>>>
```

- Можно изменить разделитель в параметре **sep** и последний символ в параметре **end**

```
a = 4
b = 6
print(a,'+',b,'=',sep='', end=' ')
print(a+b)
```

```
4+6= 10
>>>
```

Ввод данных

- Вводится целая строка!

```
a = input()          36
print(type(a))       <class 'str'>
```

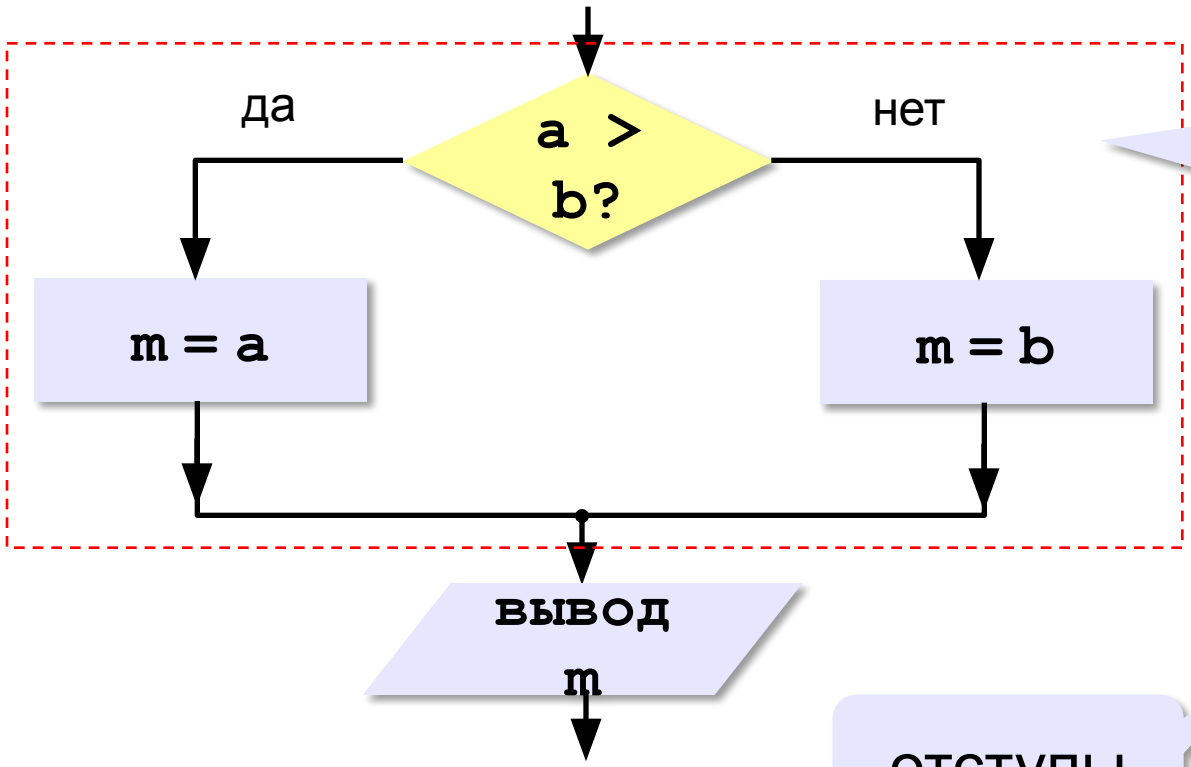
- При необходимости её надо преобразовывать в число:

```
a = int(input())     35
print(a+1)           36
```

```
print("Как тебя зовут?")
s = input()
print(f"Привет, {s}")
```

Условный оператор в Python

Пример – определение максимума из двух чисел



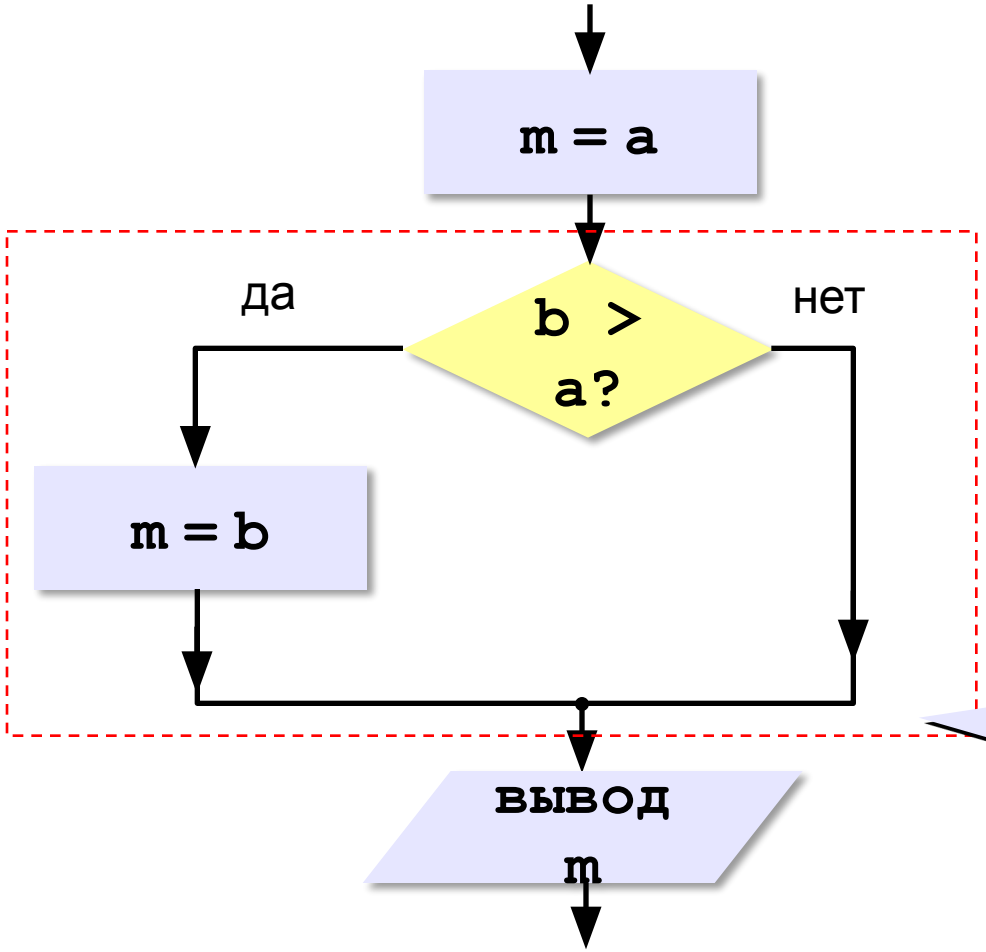
полная форма ветвления

? Если a = b?

```
if a > b:
    m = a
else:
    m = b
```

отступы

Неполная форма




```
m = a
if b > a:
   m = b
```

неполная форма ветвления

Решение в стиле Python:

```
m = max(a, b)
```

```
m = a if a > b else b
```

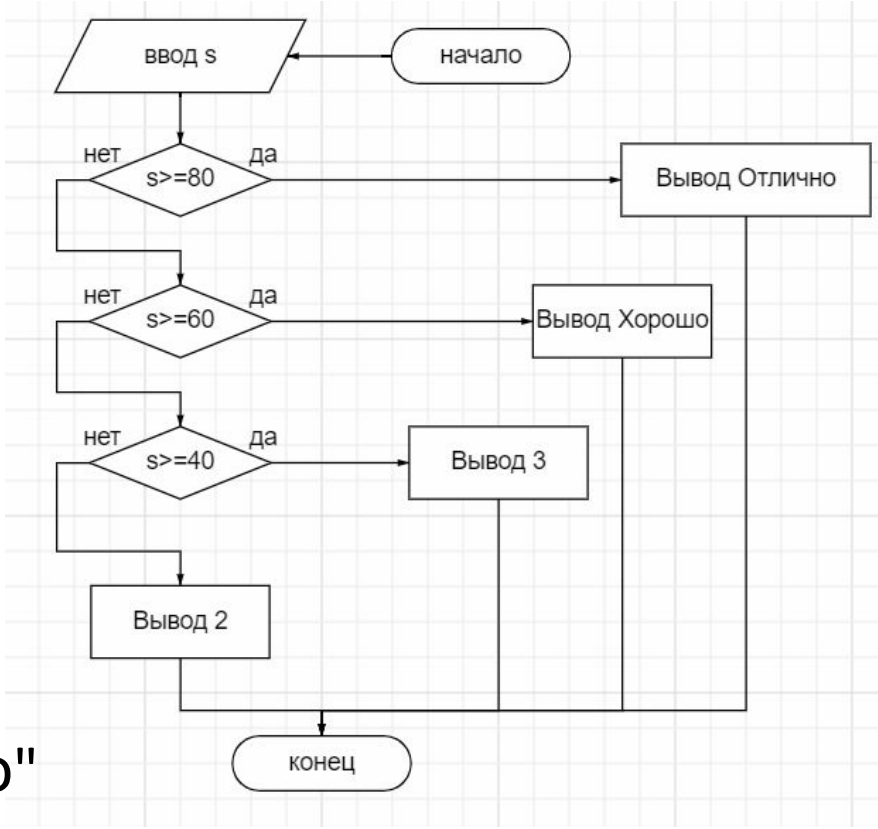
Операции сравнения

`>` `<`	больше, меньше
`>=`	больше или равно
`<=`	меньше или равно
`==`	равно
`!=`	не равно

Вложенные условия

Задача: определить оценки студента на основе введенных баллов

Пользователь вводит оценку
Если оценка >= 80
    Вывод: "отлично"
Иначе если оценка >= 60
    Вывод: "хорошо "
Иначе если оценка >= 40
    Вывод: "удовлетворительно"
Иначе
    Вывод: "неудовлетворительно"



Вложенные условия в Python

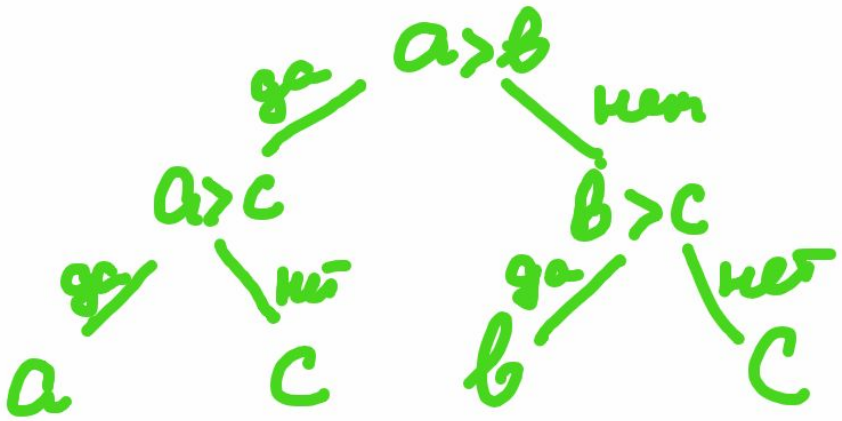
```
score = int(input())
if score >= 80:
    print("отлично")
else:
    if score >= 60:
        print("хорошо")
    else:
        if score >= 30:
            print("удовлетворительно")
        else:
            print("неудовлетворительно")
```

Каскадные условия в Python

```
score = int(input())
if score >= 80:
    print("отлично")
elif score >= 60:
    print("хорошо")
elif score >= 30:
    print("удовлетворительно")
else:
    print("неудовлетворительно")
```

Пример: максимум из трех чисел

```
a = int(input())
b = int(input())
c = int(input())
if a > b:
    if a > c:
        print(a)
    else:
        print(c)
elif b > c:
    print(b)
else:
    print(c)
```




```
a = int(input())
b = int(input())
c = int(input())
if a <= b <= c:
    print(c)
elif b <= a <= c:
    print(c)
elif a <= c <= b:
    print(b)
elif c <= a <= b:
    print(b)
else:
    print(a)
```

Логический тип данных

- Выражения логического типа в Python принимают одно из двух значений True (истина) и False (ложь)
- Логический тип называется bool в честь Джорджа Буля
- Условия ==, !=, <, >, <=, >= вычисляют значение логического типа
- Для логического типа можно использовать специальные логические операции

Логическое умножение (and, и)

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

- Логическое выражение a and b **истинно**, только если **оба** значения a и b **истинны**
- В общем случае значение выражения с оператором and истинно, если истинны **все** объединенные им условия

Пример на логическое умножение

Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

Пример:

```
5
Весна
```

Пример:

```
15
Неверный номер месяца
```

```
m = int(input())
if m >= 6 and m <= 8:
    print('Лето')
```

Логическое сложение (or, или)

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

- Логическое выражение a or b **истинно**, если **хотя бы одно** значение a и b **истинно**
- В общем случае значение выражения с оператором or истинно, если истинно **хотя бы одно** условие

Пример на логическое сложение

```
m = int(input())
if m == 12 or m <= 2:
    print('Зима')
```

Логическое отрицание (not, не)

a	not a
False	True
True	False

- Логическое выражение not a **истинно**, если a ложно и наоборот

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно)

сложное условие

```
if v >= 25 and v <= 40 :
   print("подходит")
else:
   print("не подходит")
```

Приоритет :

1) отношения (**<**, **>**, **<=**, **>=**, **==**, **!=**)
2) **not** («НЕ»)
3) **and** («И»)
4) **or** («ИЛИ»)