



Алгоритм – свойства и способы представления. Типы данных – назначение и роль в программе. Операнды и операторы – вычисление выражений

Турашова Анна Николаевна

Преподаватель

anna1turashova@gmail.com

Telegram: @anna1tur

Советы для эффективного изучения



1. Изучить основы — научиться работать с базовыми инструментами, освоить принципы и интерфейс.

2. Разрабатывать простые проекты и решать задачки - процесс решения простых задач и проектов помогает повторять и закреплять пройденный материал. Сложные проекты/задачи всегда можно разбить на более простые. Если что-то не получается - скорее всего надо упростить задачу разбив ее на несколько или перейти к п.1

3. Формировать свою библиотеку решений - рано или поздно задачи начинают повторяться и Ваши решения помогут сэкономить время и вспомнить информацию

4. Обучайте других - помогайте другим, ведите блог, записывайте видео, найдите единомышленников, делитесь с ними изученной информацией и решениями.

*Каждый дурак может написать код, понятный компьютеру.
Хорошие программисты пишут код, понятный людям*

Мартин Фаулер

Что такое программа?

Программа — это набор инструкций (команд), следуя которым компьютер выполняет определенную задачу или несколько задач.

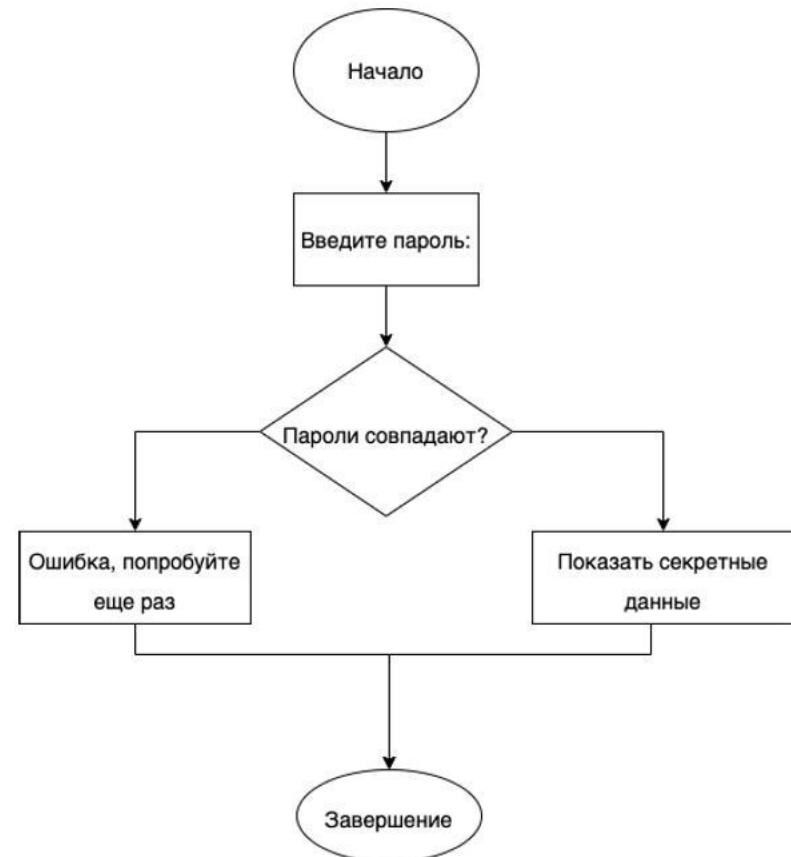


Алгоритм и программа

Алгоритм - это набор простых строго определенных последовательностей действий, объясняющих, как выполнить задачу.
Алгоритмы состоят из нескольких шагов.

Шаг алгоритма - это каждое отдельно действие алгоритма.

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





Исходный код программы

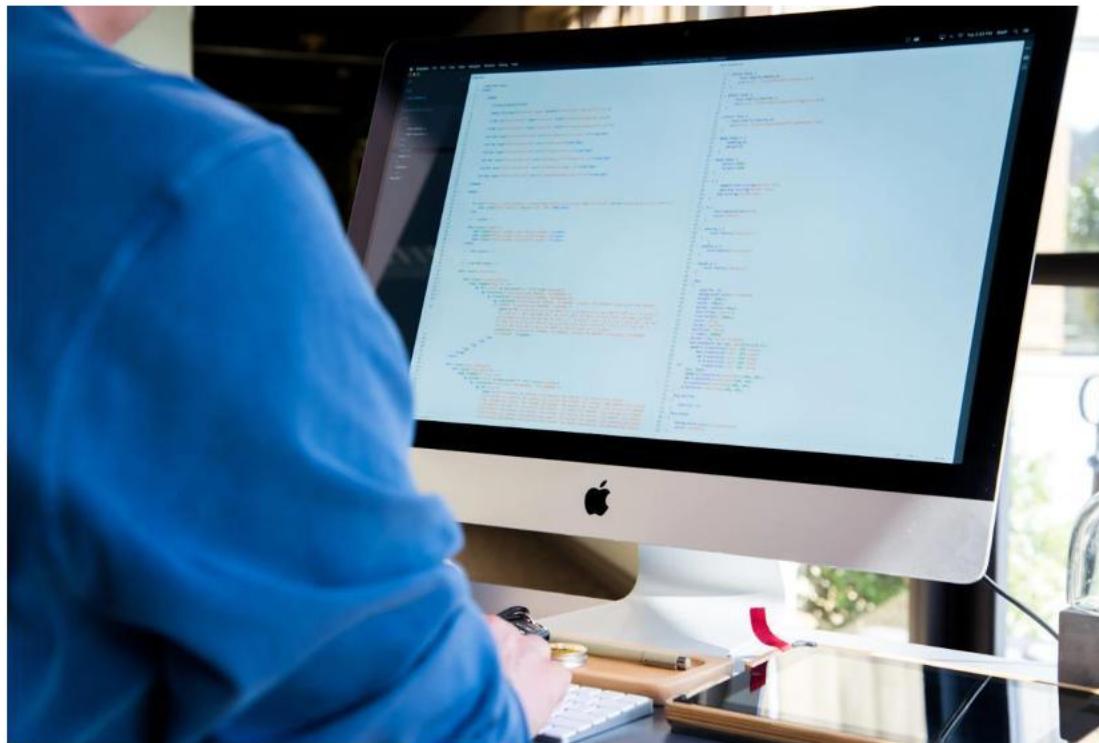
Исходный код — текст компьютерной программы на каком-либо языке программирования, который может быть прочитан человеком.

```
29 def update():
30     ...move_paddle()
31     ...move_ball()
32     ...ball_hits_paddle()
33
34 def move_ball():
35     ...global ball_xdir
36     ...global ball_ydir
37
38     ...if ball.right > WIDTH or ball.left <= 0:
39         ...ball_xdir *= -1
40
41     ...if ball.bottom > HEIGHT or ball.top <= 0:
42         ...ball_ydir *= -1
43
44     ...ball.x += ball_speed * ball_xdir
45     ...ball.y += ball_speed * ball_ydir
46
47 def ball_hits_paddle():
48     ...global ball_xdir
49     ...global ball_ydir
50
51     ...if paddle.colliderect(ball):
52         ...ball_xdir *= -1
53         ...ball_ydir *= -1
```



Что такое программирование?

Программирование — это процесс создания программ, который заключается в написании пошагового кода программы, который объясняет компьютеру, что и как ему делать.





Свойства алгоритмов

Дискретность. Алгоритм разбивается на последовательность отдельных шагов(предписаний), четко отделенных друг от друга. Только выполнив один шаг, можно приступить к выполнению следующего.

Понятность. Все шаги алгоритма должны содержаться в системе команд исполнителя.

Детерминированность – не должно быть шагов, смысл которых может восприниматься неоднозначно. На каждом шаге исполнитель должен получить четкую команду и должно быть ясно, какая команда будет выполняться следующей.

Результативность. Алгоритм должен прекратиться за конечное число шагов и должен получиться определенный результат

Массовость – алгоритм должен решать не конкретную задачу а целый класс однотипных задач, отличающихся параметрами. (исходными данными)



Способы записи алгоритмов

- **Словесная** – псевдокод, кулинарный рецепт, инструкция, правила выполнения операций.
- **Графическая** – блок-схемы. Блок-схема-ориентированный граф, указывающий порядок исполнения команд алгоритма.
- **Язык программирования** - специальный язык для записи алгоритмов для последующего автоматического выполнения на компьютере



Процесс создания программы

1. Компьютеры не умеют думать. Задача программиста - дать набор инструкций (команд) для решения задачи компьютером
2. Программист пишет подробный набор инструкций на языке программирования, который будет преобразован в набор команд понятных компьютеру. Язык выбирается в зависимости от решаемой задачи.
3. В итоге, исходный код программы становятся двоичным кодом - простейшим компьютерным языком, состоящим из команд в виде набора 0 и 1 при помощи специальных программ: **компилятором** или **интерпретатором**.

Пример: переход дороги по светофору





Программирование до создания компьютеров

- Впервые принцип программирования предложен Бэббиджем для автоматического ткацкого станка, построенного Жаккардом.
- Теоретически управляющие конструкции – ветвление и цикл описала ученица Бэббиджа - Ада Лавлейс.
- Теоретическим фундаментом для программирования стала созданная в 30-е годы прошлого века теория алгоритмов (Тьюринг, Черч, Пост, Клини).
- С появлением в 40-е годы прошлого века ЭВМ программирование превратилось в практическую деятельность человека.



Машинный язык

- Первоначально программы записывались на машинном языке – языке команд процессора в виде цифровых кодов операций и адресов операндов и результатов.
- Все данные – двоичные числа.
- Разные ЭВМ, разные процессоры имеют свои системы команд. Процесс очень трудоемкий.

Пример: сложение двух ячеек памяти
0100 0010 0011 0100
ADD (0010), (0011), (0100)



Ассемблер

- Ассемблер – набор имен для машинных команд.
- Ассемблер зависит от типа процессора, требует полного учета особенностей конкретной ЭВМ. Такие языки относят к языкам **низкого уровня**.
- Программа на Ассемблере не может непосредственно выполняться на компьютере. Выполнение требует двух этапов.
- 1 этап. Автоматический перевод программ на язык машинных команд (**трансляция**). При этом выделяются необходимые ресурсы – память и т.д.
- 2 этап. **Выполнение** программы.
- Различают два способа трансляции – **интерпретация и компиляция**

```
mov r1, r3
bl 0x80003c0 <GPIO_Init>
uint32_t seconds = 0;
mov.w r3, #0
str r3, [r7, #28]
uint8_t a = 3;
mov.w r3, #3
strb r3, [r7, #27]
uint8_t b = 2;
mov.w r3, #2
strb r3, [r7, #26]
uint8_t c = max(a,b);
ldrb r2, [r7, #26]
ldrb r3, [r7, #27]
cmp r2, r3
```



Компиляция

- Специальная программа (**компилятор**) анализирует текст программы, выявляет синтаксические ошибки и после этого создает программу из машинных инструкций, готовую к выполнению.
- Сам процесс компиляции может быть медленным, но программа в результате работает очень **быстро**.
- Есть возможность **найти множество ошибок** сразу.

Что такое компилятор?

Компилятор - транслирует исходный код записанный на языке программирования в бинарный код. Компиляцию программы достаточно выполнить один раз.





Плюсы и минусы компилятора

Достоинства компиляторов:

- Быстрота работы программ;
- Отсутствие надобности компилятора на компьютере пользователя.

Недостатки компиляторов:

- Программа зависит от ОС и устройства, под которую была скомпилирована.
- При внесении изменений требуется перекомпиляция кода.



Интерпретация

- Программа – **интерпретатор** построчно читает, анализирует и выполняет исходный код, записанный на языке программирования.
- Процесс интерпретации программы происходит **каждый раз** во время ее запуска – это **медленно**.
- Если программа не дошла до какой-то строки, она **не сможет найти ошибку** в ней.

Что такое интерпретатор?

Интерпретатор - построчно читает, анализирует и выполняет исходный код записанный на языке программирования. Процесс интерпретации программы происходит каждый раз во время ее запуска.





Плюсы и минусы интерпретатора

Достоинства интерпретаторов:

- Независимость от ОС (переносимость кода).
- При внесении изменений НЕ требуется перекомпиляция кода.

Недостатки интерпретаторов:

- Для запуска программы требуется наличие интерпретатора.
- “Низкая” скорость работы.



Сравнение интерпретаторов и компиляторов

Достоинства компиляторов:

- Быстрота работы программ;
- Отсутствие надобности компилятора на компьютере пользователя.

Недостатки компиляторов:

- Программа зависит от ОС и устройства, под которую была скомпилирована.
- При внесении изменений требуется перекомпиляция кода.

Достоинства интерпретаторов:

- Независимость от ОС (переносимость кода).
- При внесении изменений НЕ требуется перекомпиляция кода.

Недостатки интерпретаторов:

- Для запуска программы требуется наличие интерпретатора.
- “Низкая” скорость работы.



ЯЗЫКИ ВЫСОКОГО УРОВНЯ

- более выразительны и наглядны (близки к естественным языкам), что повышает наглядность и понятность текста
- независимы от типа компьютера, набор операций выбирается из соображений удобства программиста
- поддерживают различные типы данных, а не только числа.
- Первый ЯВУ для научных применений **Фортран** (переводчик формул) разработан в **1954** г. программистами фирмы IBM.
- Существует множество различных языков программирования, ориентированных на разные применения.
- Неоднократно были попытки создать универсальный язык, но такой язык всегда получался очень сложным, перегруженным конструкциями. (PL/1 – IBM 1967, ADA, 1979 г.).





Сравнение языков



Python

```
print('Hello, World!')
```

```
#include <iostream>  
  
int main() {  
    std::cout << "Hello, World!";  
    return 0;  
}
```

C++

Pascal

```
begin  
    writeln('Hello, World!')  
end.
```

```
class Program  
{  
    Ссылок: 0  
    static void Main()  
    {  
        System.Console.WriteLine("Hello, World!");  
    }  
}
```

C#



Парадигмы программирования

- **Процедурное программирование** – требует точного описания алгоритма как последовательности выполняемых команд (Pascal, C, Basic).
- **Объектно-ориентированное программирование** – реализация обработки информации как взаимодействия множества объектов. Хорошо подходит для разработки больших и сложных программных комплексов (Delphi, C++, C#, Java, Python).
- **Декларативное** – алгоритм не строится, специальным образом описываются исходные данные и результат, а исполняющая система сама выбирает и выполняет алгоритм (Prolog – логическое, LISP – функциональное)



Современные языки программирования

Объектно-ориентированные

- C++
- Delphi
- Go
- Java
- JavaScript
- Kotlin
- Objective-C
- Perl
- PHP
- Python
- Ruby
- Swift
- Visual Basic

Структурные (Процедурные)

- Basic
- Pascal
- Фортран
- Си

Функциональные

- F#
- Haskell
- Lisp

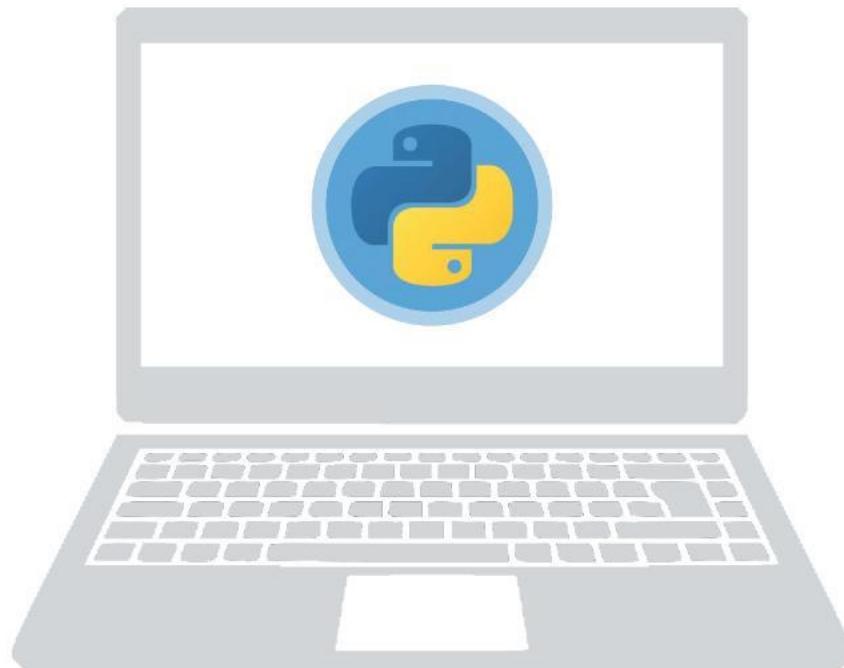
Мультипарадигменные

- C++
- C#
- Delphi
- Lisp
- Python
- Swift



Язык Python

Python - современный язык программирования, позволяющий создавать различные компьютерные программы, например: веб-сайты, игры, приложения. Для запуска исходного кода на языке Python используется специальная программа интерпретатор языка Python.





Особенности Python

- Интерпретируемый язык программирования высокого уровня
- Динамическая типизация
- Высокоуровневые структуры данных
- Поддерживает структурное, объектно-ориентированное, функциональное, программирование
- Активно развивающийся (последняя версия 05.10.21, python 3.10.0)
- Большое количество различных библиотек
- Области применения – анализ данных, веб-разработка, системное программирование



Установка Python

Для установки Python перейти на сайт [Python.org](https://www.python.org) и скачать последнюю версию Python для вашей ОС

Для запуска и редактирования Python программ скачать PyCharm Community: <https://www.jetbrains.com/ru-ru/pycharm/download/>

Если вы используете другой редактор кода, то установите расширение Python



Операторы

Оператор - наименьшая автономная часть языка программирования; команда; Он предписывает, что делать компьютеру.

Операции - любые действия, которые совершаются с помощью операторов.

Арифметические операторы - символы, с помощью которых выполняются математические действия в языке программирования.



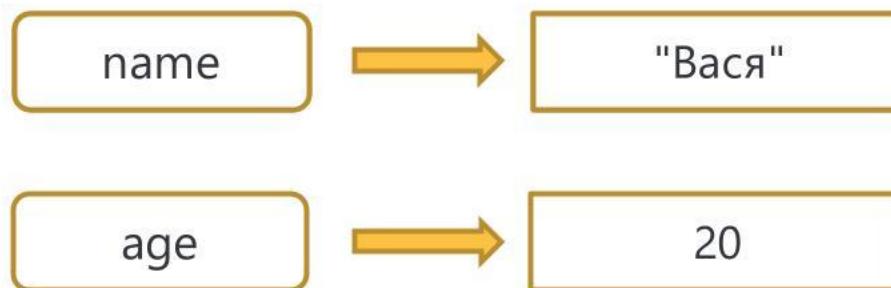


Арифметические операторы

Символ	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
//	Целочисленное деление
%	Деление по модулю
**	Возведение в степень

Переменная

Переменная - имя, которому с помощью оператора присваивания присвоено значение. В Python оператор присваивания устанавливает связь между именем и значением, которое хранится в виде объекта в памяти компьютера.





Ключевое слово

Ключевое слово - это слово, «зарезервированное» за программой и имеющее специальное значение. Ключевые слова могут быть командами или параметрами. В каждом языке программирования есть свои ключевые (зарезервированные) слова. Они не могут использоваться в качестве имен для переменных.

Получение списка ключевых слов в Python:

```
>>> 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',  
'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield' ]
```



Типы данных

Тип данных - описывает множество значений и операций над этими значениями

Тип данных	Примеры
int	1, 5, -9, 0
float	2.5, 2.3333333
bool	True, False
str	"Hello world"
NoneType	None
list	[4, 1, 0, 9]
tuple	(5, 3, 9, 2)
dict	{ name: 'ivan', kg: 63 }
set	{ 5, 9, 1, 0 }
complex	2 + 3j



Преобразование типов данных

Функция	Примеры
int()	int("1234")
float()	float("1234.43")
bool()	bool(2), bool("a")
str()	str(234)
list()	list("abcde")
tuple()	tuple("egda")
dict()	dict([('k', 'v')])
set()	set([2,2,2,1,1])
complex()	complex(123)



Ввод и вывод данных

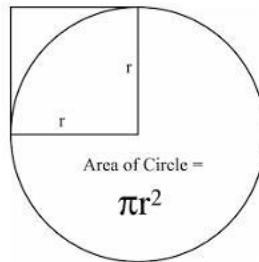
Функция **print()** выводит строку на экран

Функция **input()** позволяет получить данные от пользователя и возвращает строку



Задачи

1. Напишите программу Python, чтобы получить версию Python, которую вы используете.
2. Напишите программу, которая преобразует кг в граммы. На входе получает значение в кг, а на выходе значение в граммах
3. Напишите программу Python, которая принимает радиус круга от пользователя и вычисляет площадь



4. Напишите программу Python, которая принимает имя и фамилию пользователя, и выводит сперва фамилию, а потом имя с пробелом между ними.
5. Напишите программу Python, которая вычисляет периметр и площадь прямоугольника
6. Напишите программу Python, которая принимает два числа a и b , далее находит сумму, разность, произведение и деление a на b . В результате отобразите ответы для каждой из арифметической операции.



Ссылки

<https://python.org> - сайт языка Python

<https://pythontutor.com/visualize.html> - визуализация выполняемого python кода

https://www.dropbox.com/s/4000zdw0dy55xm2/blockly_linear.zip?dl=1 - задания для освоения хода выполнения программы

<https://www.dropbox.com/sh/vi950kslomc4jgn/AABNZj0Glb-HMuKk3Tld0xsa?dl=0> - исходный код примеров занятия



Входит в ГК Аплана



АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning
и очное
обучение



Ежегодные награды
Microsoft,
Huawei, Cisco и
другие

Направления обучения:

Информационные технологии

Информационная безопасность

ИТ-менеджмент и управление проектами

Разработка и тестирование ПО

Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск,
Хабаровск, Красноярск, Тюмень, Нижний
Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис
в Москве

Ресурсы более 400
высококлассных
экспертов и
преподавателей

Разработка
программного
обеспечения и
информационных
систем

Программы по
импортозамещению

Сеть региональных учебных центров
по всей России

Крупные заказчики



100+
сотрудников



АКАДЕМИЯ АЙТИ



Спасибо за внимание!

Центральный офис:

Москва, Варшавское шоссе 47, корп. 4, 7 этаж

Тел: +7 (495) 150-96-00

academy@it.ru

academyit.ru