Программирование на

Литература:

1. Изучаем Python. Марк Лутц
2. Язык программирования Python. Сузи Р.А.
3. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих.

История PYTHON

Язык программирования PYTHON был создан в 1991 году голландцем Гвидо ван Россумом.

Свое имя – Пайтон (или Питон) – получил от названия телесериала, а не пресмыкающегося.

После того, как Россум разработал язык, он выложил его в Интернет, где уже целое сообщество программистов присоединилось к его улучшению.

Python активно совершенствуется и в настоящее время. Часто выходят его новые версии. Официальный сайт http://python.org.

Дзэн Питона

Если интерпретатору Питона дать команду import this (импортировать "сам объект"), то выведется так называемый "Дзен Питона", иллюстрирующий идеологию и особенности данного языка. Глубокое понимание этого дзена приходит тем, кто сможет освоить язык Python в полной мере и приобретет опыт практического программирования.

Дзэн PYTHON (философия)

- *Красивое лучше, чем уродливое.*
- *Явное лучше, чем неявное.*
- *Простое лучше, чем сложное.*
- *Сложное лучше, чем запутанное.*
- *Плоское лучше, чем вложенное.*
- *Разреженное лучше, чем плотное.*
- *Читаемость имеет значение.*
- *Особые случаи не настолько особые, чтобы нарушать правила.*
- *Должен существовать один — и, желательно, только один — очевидный способ сделать это.*
- *Если реализацию сложно объяснить — идея плоха.*

- **Python** – высокоуровневый язык программирования общего назначения с акцентом на *производительность разработчика и читаемость кода*
- Python и подавляющее большинство библиотек к нему бесплатны и поставляются в исходных кодах. Более того, в отличие от многих открытых систем, лицензия никак не ограничивает использование Python в коммерческих разработках
- Python имеет ясный синтаксис. В нем сведены к минимуму такие вспомогательные конструкции как скобки, слова-организаторы_блоков. Взамен программист обязан четко соблюдать отступы, которые и являются организаторами блоков. В результате код получается незагруженным лишними элементами и легко читаемым.

Преимущества Python

Скорость выполнения программ написанных на Python очень высока. Это связанно с тем, что основные библиотеки Python написаны на C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня.

В связи с этим вы можете писать свои собственные модули для Python на C или C++

В стандартных библиотеках Python вы можете найти средства для работы с электронной почтой, протоколами Интернета, FTP, HTTP, базами данных, и пр.

Скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях.

Python подходит для любых решений в области программирования, будь то офисные программы, вэб-приложения, GUI-приложения и т.д.

Над разработкой Python трудились тысячи энтузиастов со всего мира. Поддержкой современных технологий в стандартных библиотеках мы можем быть обязаны именно тому, что Python был открыт для всех желающих.

Недостатки PYTHON

- PYTHON, как и другие интерпретируемые языки, имеет сравнительно невысокую скорость выполнения программ. Однако, в случае с Python этот недостаток компенсируется уменьшением времени разработки программы. В среднем, программа на Python в 2-4 раза компактнее, чем её аналог на C++ или Java
- Динамическая типизация вызывает вопросы у методистов по обучению программированию

Интерактивный режим

В основном интерпретатор выполняет команды построчно: пишешь строку, нажимаешь Enter, интерпретатор выполняет ее, наблюдаешь результат.

Возможности языка позволяют использовать его как калькулятор, не зная команд программирования.

2 + 5

3 * (5 - 8)

2.4 + 3.0 / 2

и т.д.

Наберите подобные примеры в интерактивном режиме (Console). Ответ выдается сразу после нажатия.

Синтаксис

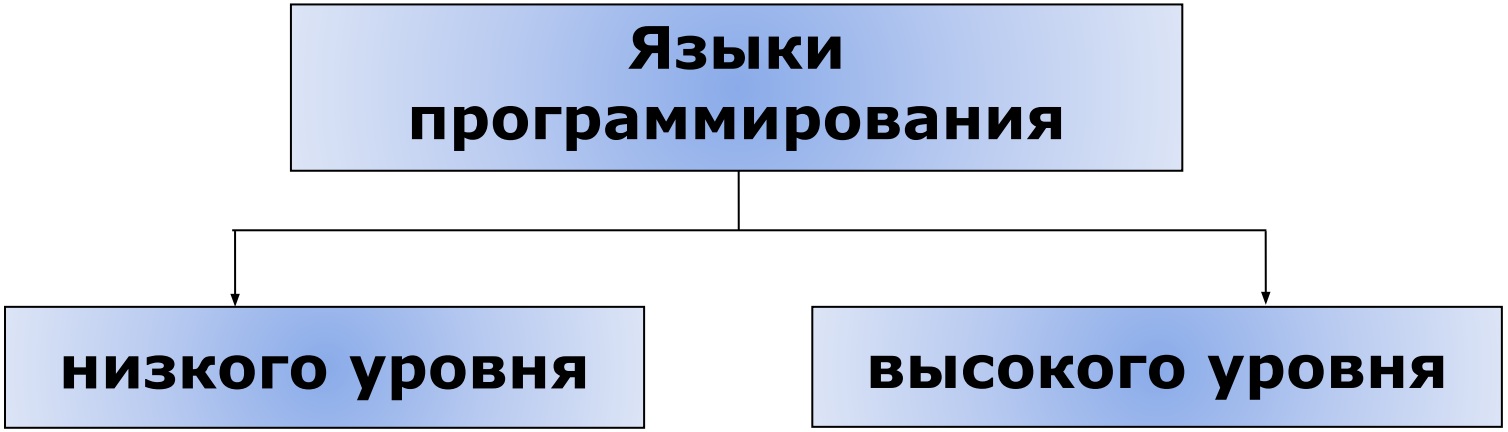
во первых стоит отметить интересную особенность Python. Он не содержит операторных скобок (begin..end в pascal или {..}в Си), вместо этого блоки выделяются отступами: пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием. Однострочные комментарии начинаются со знака фунта «#», многострочные — начинаются и заканчиваются тремя двойными кавычками «"""».

Чтобы присвоить значение переменной используется знак «=», а для сравнения — «==». Для увеличения значения переменной, или добавления к строке используется оператор «+=», а для уменьшения — «-=». Все эти операции могут взаимодействовать с большинством типов, в том числе со строками.

Программа.
Язык программирования

Программа – набор инструкций для определенного исполнителя.

Язык программирования – это формальный язык, предназначенный для записи программ (обычно для ЭВМ).



Компиляторы и интерпретаторы

Транслятор – специальная программа, преобразующая программный код с того или иного языка программирования в машинный код

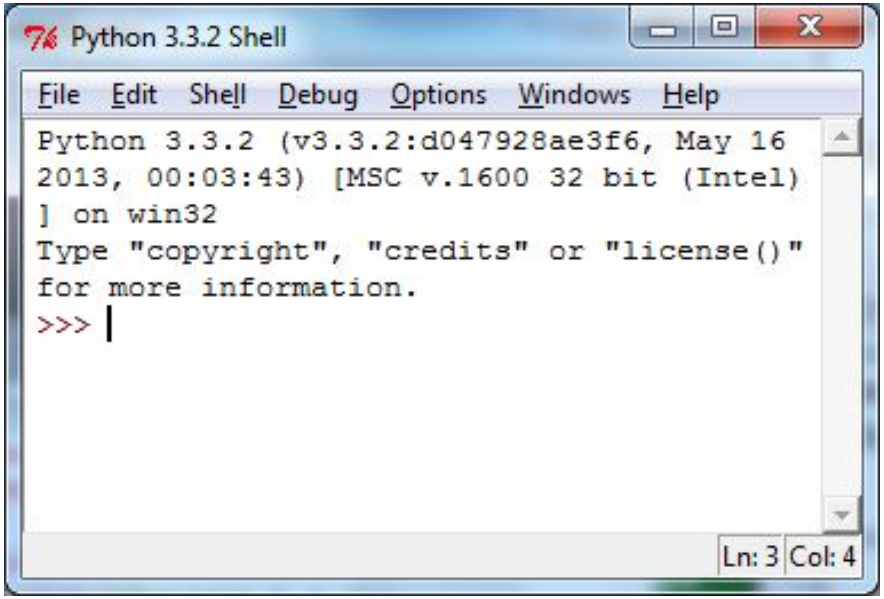
Компилятор
Сразу переводит весь программный код на машинный язык. Создает исполняемый файл.

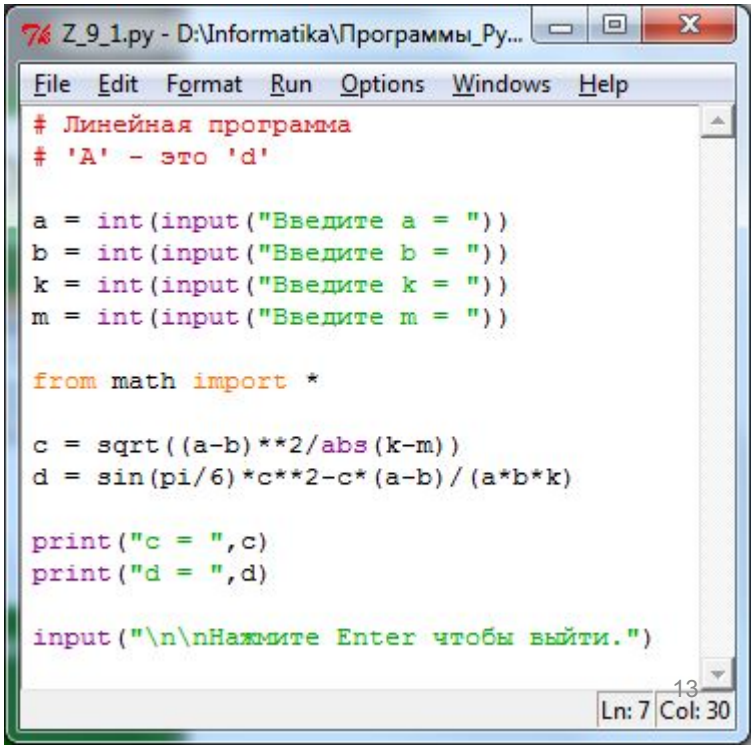
Интерпретатор
Переводит программный код построчно. Напрямую взаимодействует с операционной системой.

Особенности Python

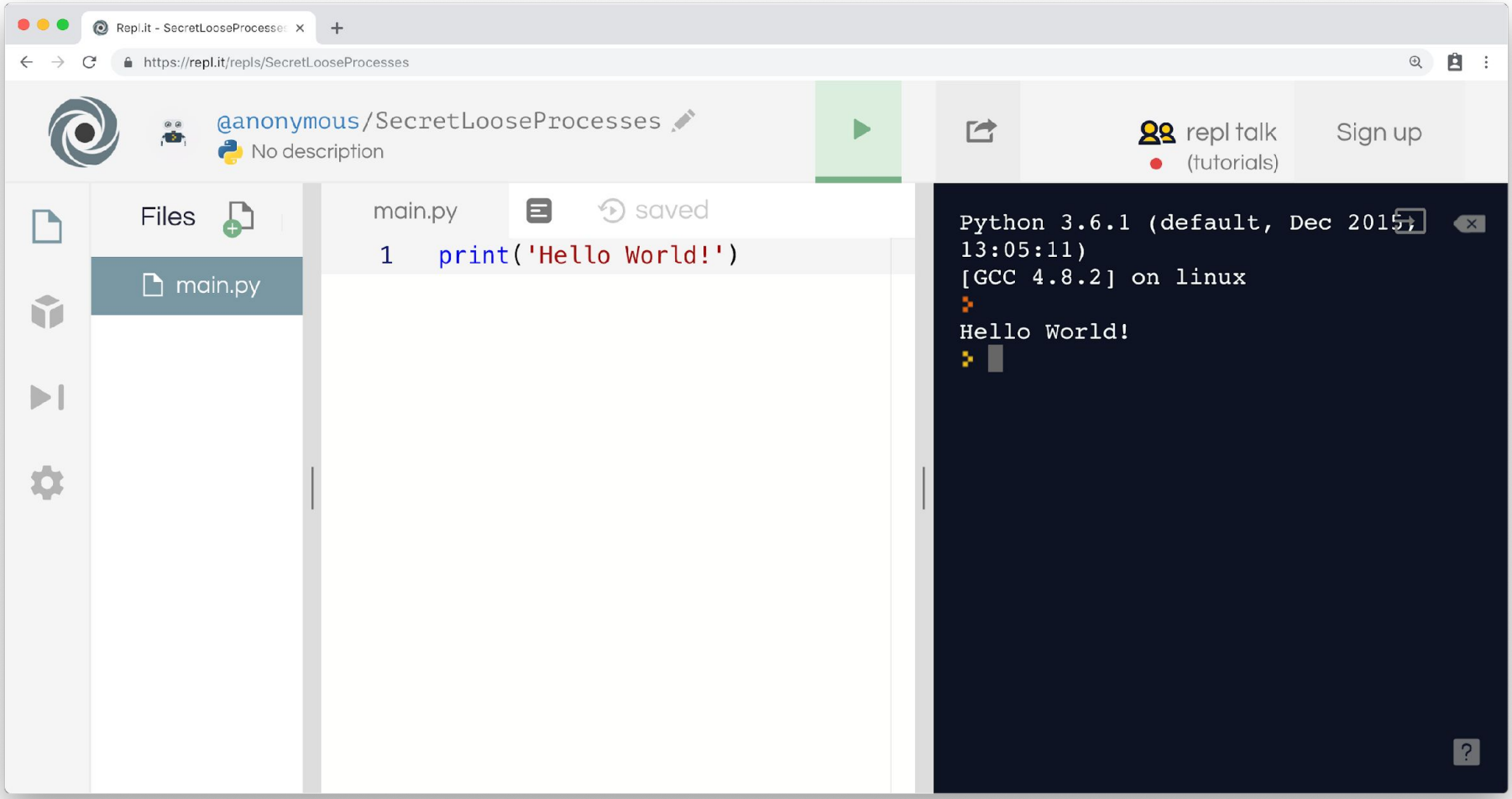
- Интерпретируемый язык
- Ясный синтаксис
- Полноценный универсальный язык
- Свободно-распространяемый интерпретатор

Два режима работы: интерактивный и сценарный





Web IDE:
https://repl.it/languages/python3



Текстовый редактор + терминал

code.py

```
def fibonacci(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)


print("Result is", fibonacci(12))
```

Line 10, Column 1 Spaces: 4 Python

Sublime Text 3

1 — surenkhorenyan@Surens-iMac — ..US/Course_1/1 — -zsh

```
[ surenkhorenyan@Surens-iMac  Course_1/1  python3.6 code.py ]
Result is 144
 surenkhorenyan@Surens-iMac  Course_1/1
```

Terminal

JetBrains PyCharm IDE:
https://www.jetbrains.com/pycharm/

Project
Introduction to Python
Introduction
Variables
Variable definition
Undefined variable
undefined_variable.py
Variable types
variable_type.py
Type conversion
Arithmetic operators
Assignments
Boolean operators

variable_type.py

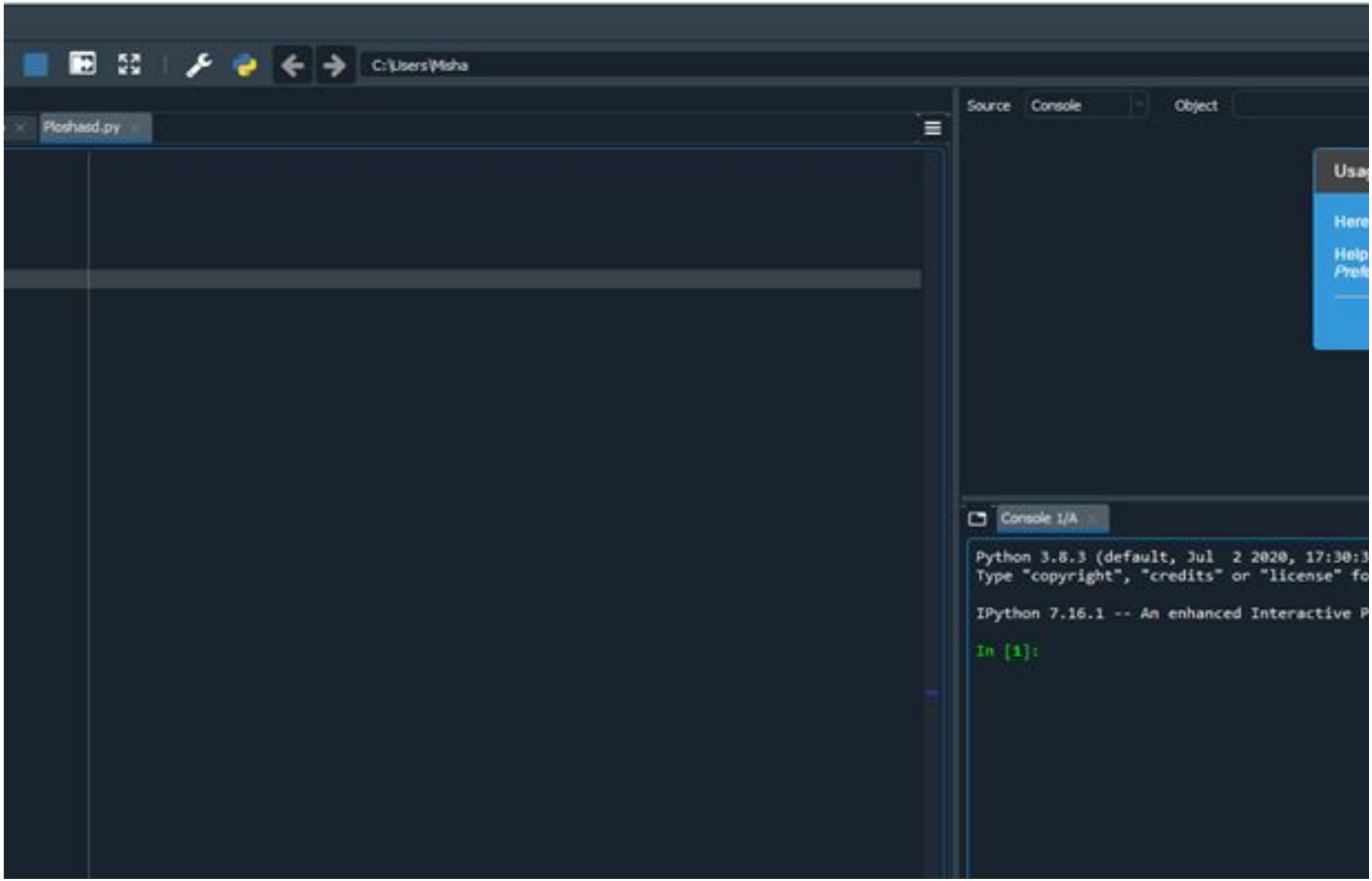
```
number = 9  number: 9
print(type(number))   # print type of variable "number"

float_number = 9.0  float_number: 9.0
print(type(float_number))
```

Debug variable_type
Debugger
Frames
MainThread
Globals (variable_type.py)
Variables
Special Variables
float_number = {float} 9.0
number = {int} 9
Console

```
<type 'int'>

>>>
```

3:1 LF UTF-8

https://www.anaconda.com/products/individual



Основные элементы блок-схем

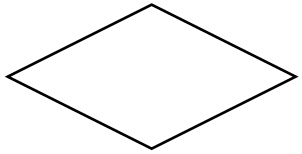
– Начало вычислительного процесса.

– Конец вычислений.

– Блок обмена информацией;
ввод данных и вывод результатов.

– Вычислительный блок;
выполнение операции или группы операций вычислительного процесса.

– Алгоритмический блок;
использование ранее созданных и от дельно описанных алгоритмов.



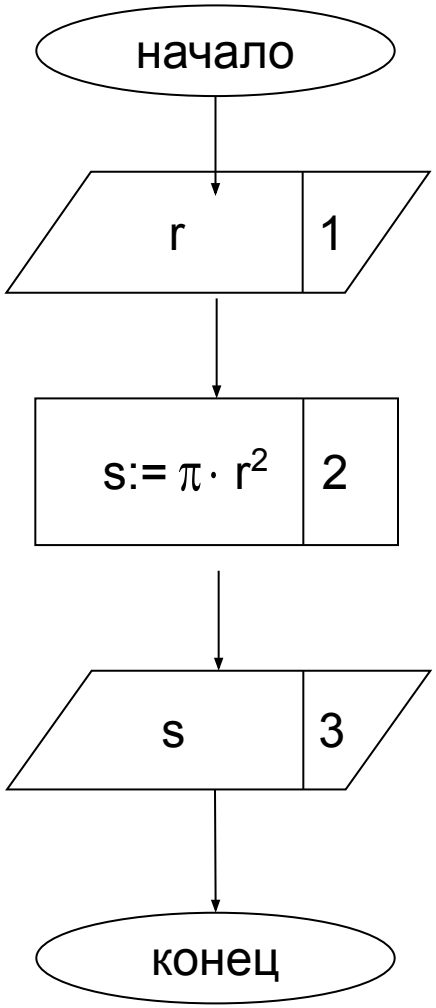
– Логический блок;
выбор направления выполнения алгоритма в зависимости от условия.

– Циклический блок;
организует многократное выполнение вложенных блоков.

СЛЕДОВАНИЕ

Последовательное исполнение блоков один за другим.

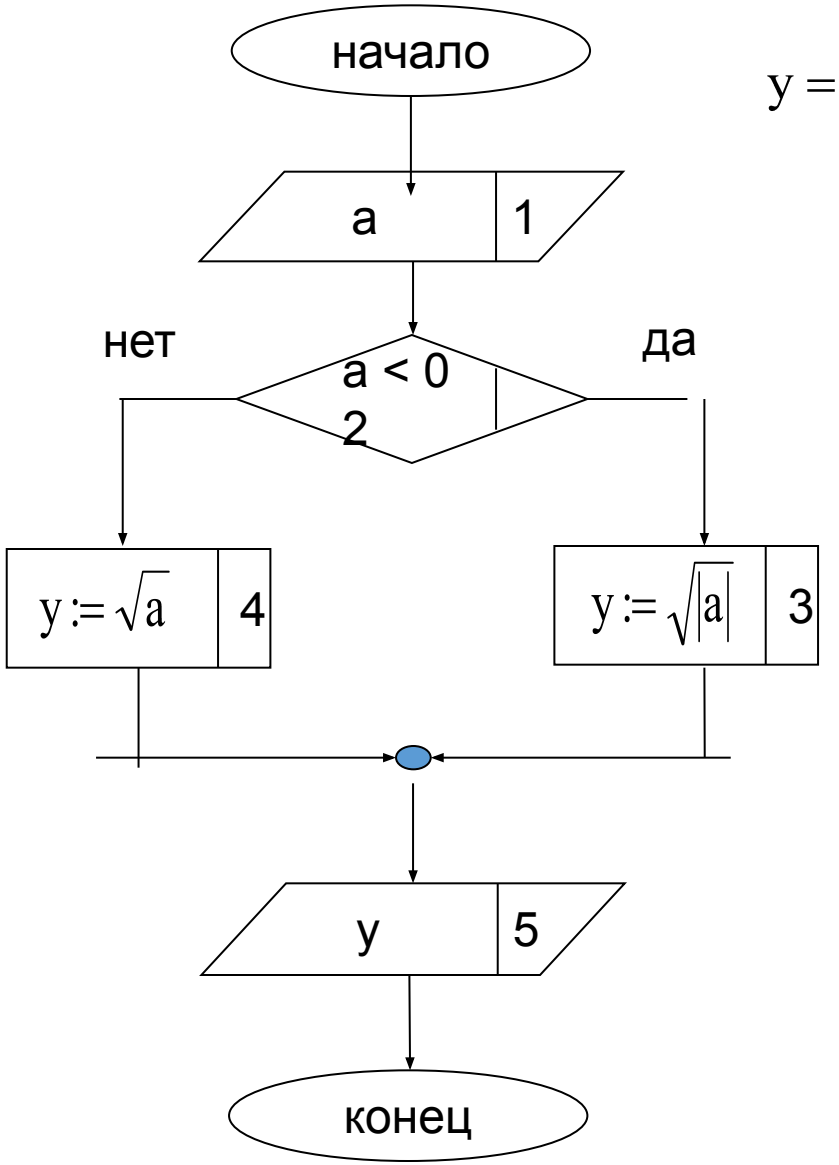
Вычислить площадь круга. $s = \pi \cdot r^2$



ВЕТВЛЕНИЕ

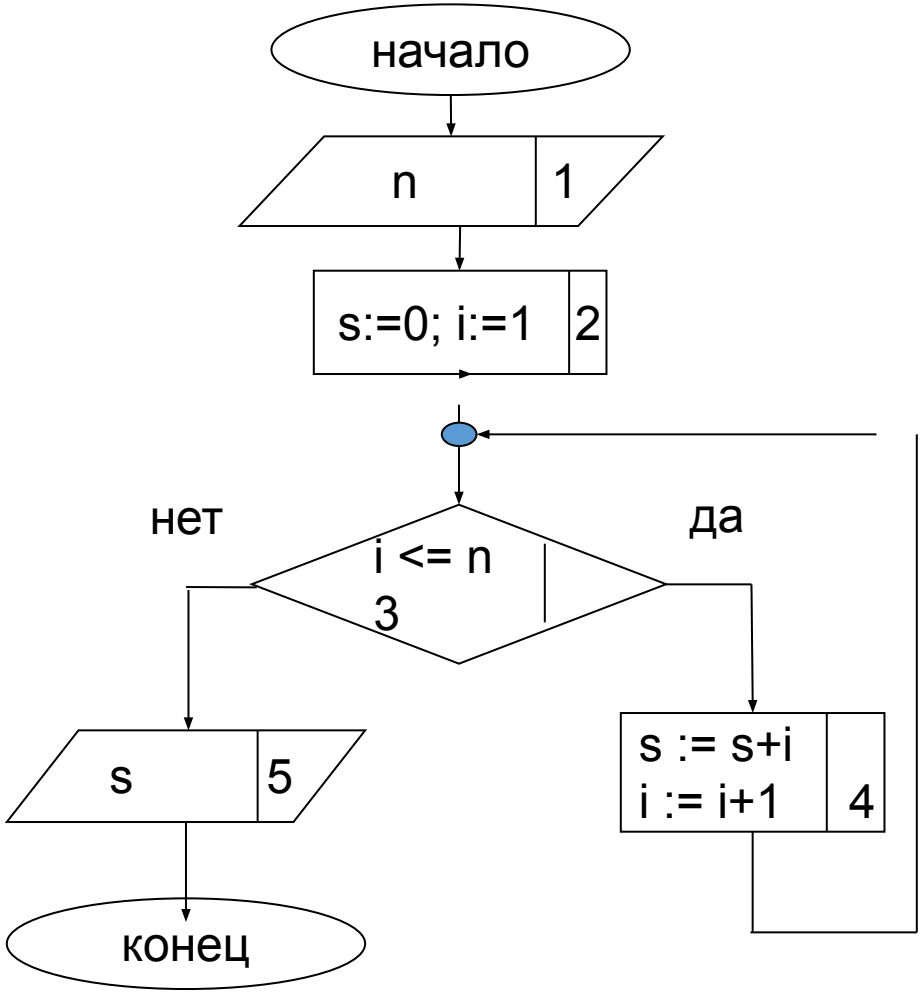
Составить блок-схему алгоритма и программу вычисления функции:

$$y = \begin{cases} \sqrt{|a|}, & \text{если } a < 0 \\ \sqrt{a}, & \text{если } a \geq 0 \end{cases}$$



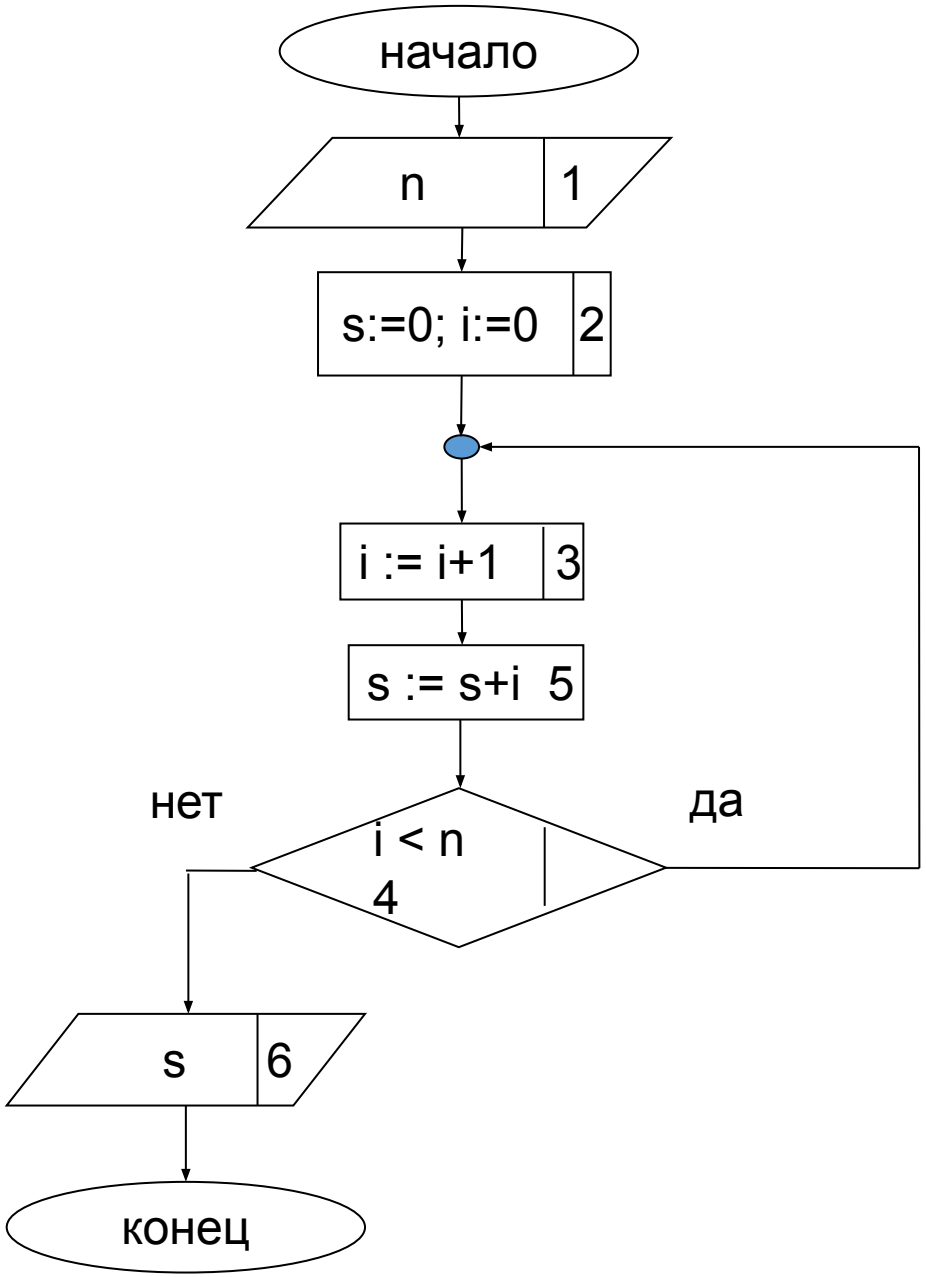
Составить блок-схему алгоритма и программу вычисления суммы n чисел натурального ряда. S=1+2+3+….+n

ЦИКЛ С ПРЕДУСЛОВИЕМ

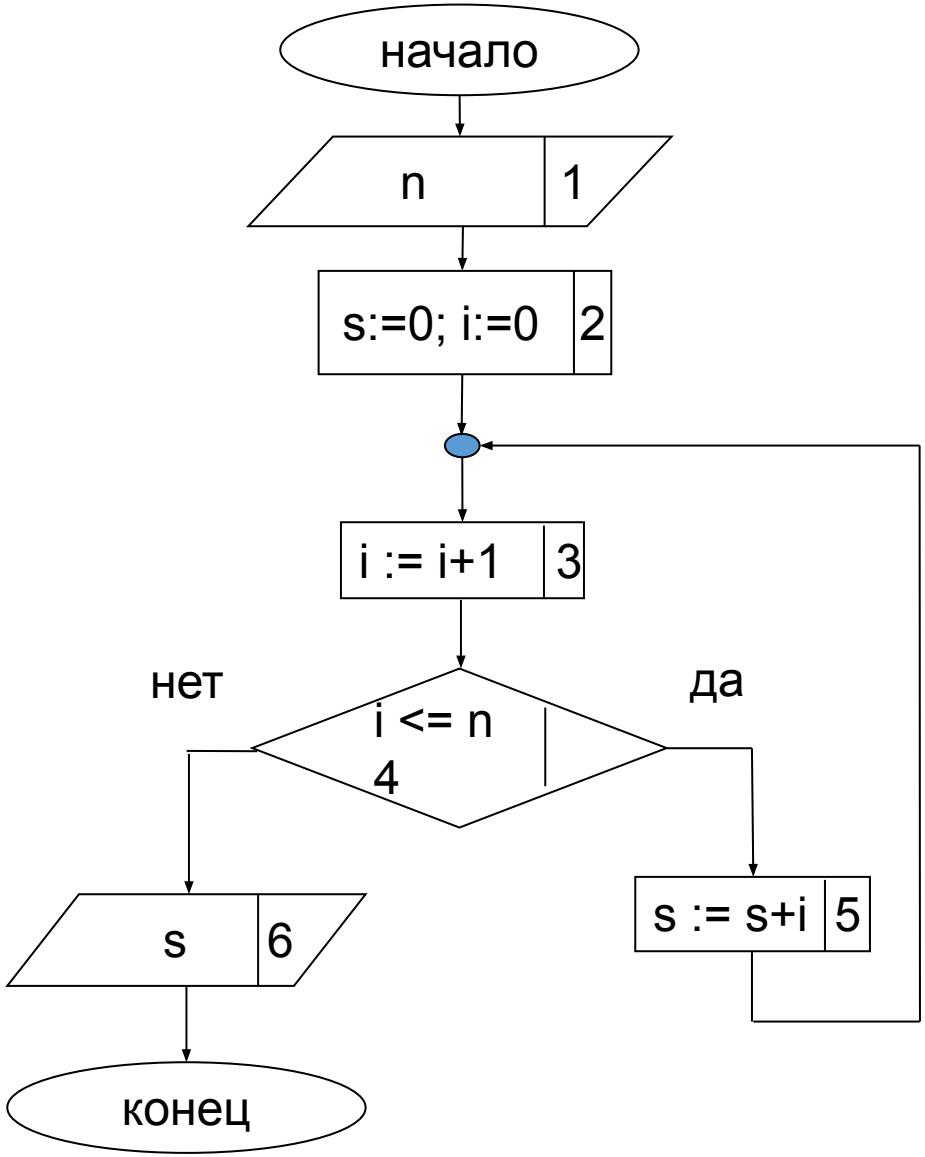


Инструкция **break**– Выход из цикла в произвольном месте

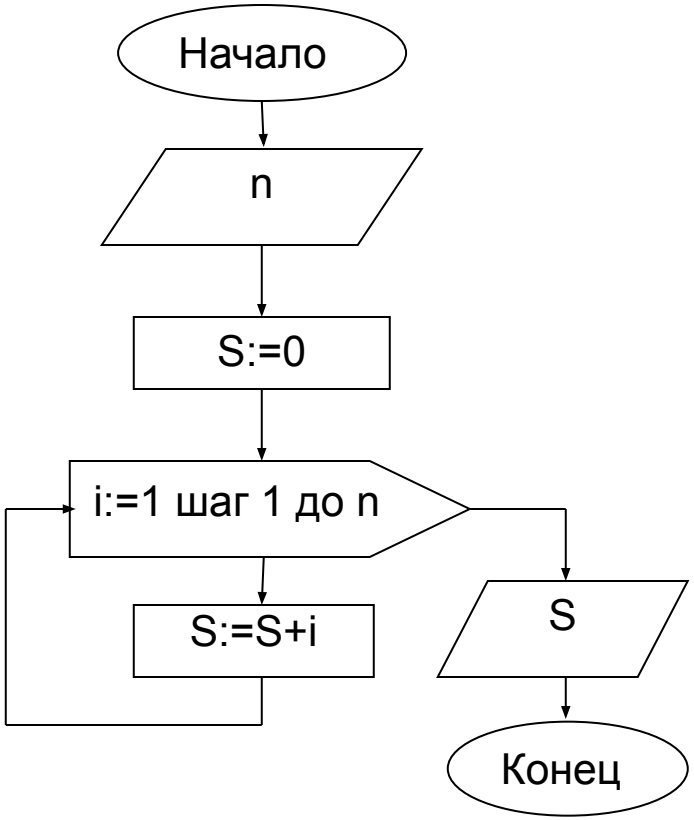
ЦИКЛ С ПОСТУСЛОВИЕМ



ЦИКЛ БЕЗУСЛОВИЯ



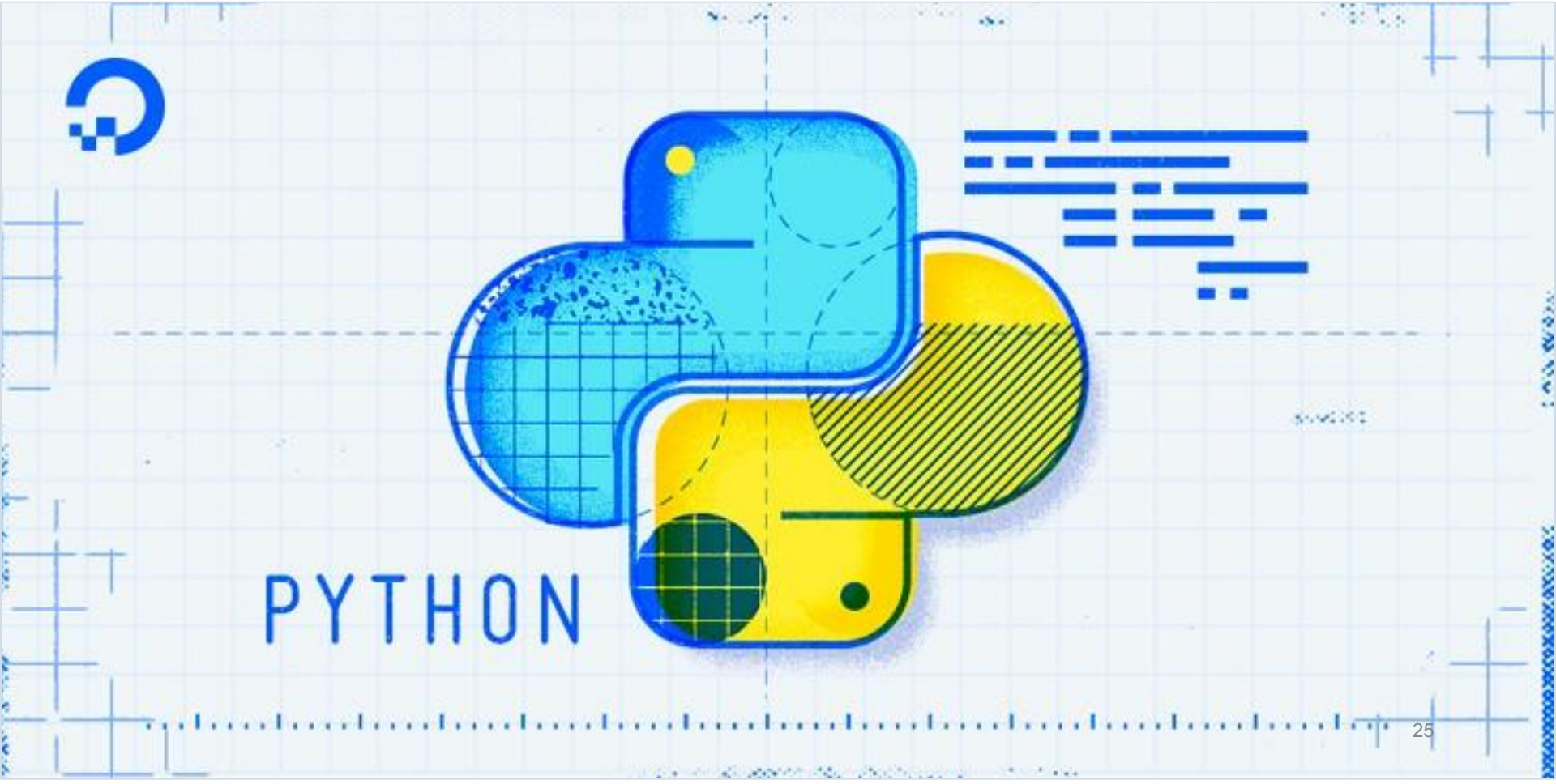
ЦИКЛ С ПАРАМЕТРОМ



Инструкция **break**– Выход из цикла в произвольном месте

Типы данных?

Каждое значение в Python имеет тип. Поскольку в Python всё — объект, типы являются классами, а значения — экземплярами (объектами) этих классов.



Данные и их типы

- ***целые числа (integer)*** – положительные и отрицательные целые числа, а также 0 (**например: 4, 687, -45, 0**).
- ***числа с плавающей точкой (float point)*** – дробные числа (**например: 1.45, -3.789654, 0.00453**). Примечание: разделителем целой и дробной части служит **точка**, а не запятая.
- ***строки (string)*** — набор символов, заключенных в кавычки (**например: "ball", "What is your name?", 'dkfjUUv', '6589'**).

Примечание: кавычки в Python могут быть одинарными или двойными.

Операции. Операции над разными типами данных

Выражение	Результат выполнения
34.907 + 320.65	355.55699999999996
'Hi, ' + 'world :) '	'Hi, world :) '
'Hi, ' * 10	'Hi, Hi, Hi, Hi, Hi, Hi, Hi, Hi, Hi, Hi, '
'Hi, ' + 15	О ш и б к а

Изменение типов данных

int() – преобразует аргумент в целое число

str() – преобразует аргумент в строку

float() – … в число с плавающей точкой

Выражение	Результат выполнения
int ('56')	56
int (4.03)	4
int ("comp 486")	О ш и б к а
str (56)	'56'
str (4.03)	'4.03'
float (56)	56.0
float ("56")	56.0

Математические операторы

Оператор	Описание	Пример	Результат
+	Сложение	7 + 3	10
-	Вычитание	7 - 3	4
*****	Умножение	7 * 3	21
/	Деление (истинное)	7 / 3	2.3333333333333335
******	Возведение в степень	7**3	343
//	Целочисленное деление	7 // 3	2
%	Остаток от деления	7 % 3	1

Переменные в Python

Переменная – это ссылка на область памяти, где хранятся те или иные данные

Имена переменных в Python

1. Имя переменной может состоять только из цифр, букв и символов подчеркивания
2. Имя переменной не может начинаться с цифр
3. Имя должно описывать суть , т.е. нужно давать имена, говорящие о назначении данных, на которые они ссылаются
4. Имя переменной не должно совпадать с командами языка (зарезервированными ключевыми словами)
5. Имя переменной принято начинать со строчной буквы
6. Не следует создавать имена длиннее 15 символов

Чтобы узнать значение, на которое ссылается переменная, находясь в режиме интерпретатора, достаточно ее вызвать (написать имя и нажать Enter).

Пример работы с переменными

```
>>> apples = 100
>>> eat_day = 5
>>> day = 7
>>> apples = apples - eat_day * day
>>> apples
65
>>> |
```

Логические выражения и логический тип данных

Выражение:	Значение:
"Сумма чисел 3 и 5 больше 7"	**Правда – True (1)**
"Сумма чисел 3 и 5 **меньше 7** "	**Ложь – False(0)**

Если результатом вычисления выражения может быть лишь истина или ложь, то такое выражение называется **логическим**.

Логические выражения и логический тип данных

Операторы сравнения

Оператор	Значение	Выражение
==	Равно	A==B
!=	Не равно	A!=B
>	Больше	A>B
<	Меньше	A<B
>=	Больше или равно	A>=B
<=	Меньше или равно	A<=B

Логические выражения и логический тип данных

Примеры работы с логическими выражениями на языке программирования Python (после # написаны комментарии):

```
x = 12 – 5 # это не логическая операция, а
             операция присваивания переменной x
             результата выражения 12 — 5
x == 4     # x равен 4
x == 7     # x равен 7
x != 7     # x не равен 7
x != 4     # x не равен 4
x  > 5     # x больше 5
x  < 5     # x меньше 5
x >= 6     # x больше или равен 6
x <= 6     # x меньше или равен 6
```

Логические выражения и логический тип данных

Логические операции

Логические выражения в результате вычисления принимают логические значения **True** и **False**.

Логические операции: отрицание - **NOT**, логическое умножение - **AND**, логическое сложение - **OR**, исключающее «или» -XOR .

Примеры: (после # написаны комментарии):

x = 8 y = 13

x == 8 **and** y < 15 *# x равен 8 и y меньше 15*

x > 8 **and** y < 15 *# x больше 8 и y меньше 15*

x != 0 **or** y >15 *# x не равен 0 или y больше 15*

x < 0 **or** y >15 *# x меньше 0 или y больше 15*

Ввод и вывод данных

- осуществляется с помощью встроенных функций

Ввод: input (параметры)

Вывод: print (параметры)

Ввод данных

1.

```
>>> input()
1234
'1234'
>>> input()
Hello World!
'Hello World!'
>>>
```

2. Параметр - приглашение

```
>>> input('Введите число:')
Введите число:10
'10'
>>> int(input('Введите число:'))
Введите число:10
10
>>> float(input('Введите число:'))
Введите число:10
10.0
>>>
```

3. Тип данных - строчный

```
>>> input('Введите номер карты:')
Введите номер карты:98765
'98765'
>>> input('Введите имя:')
Введите имя:Иван
'Иван'
>>>
```

4. Присвоение значения переменной

```
>>> name = input ('Введите  Ваше имя:')
Введите Ваше имя: Мария
>>> name
'Мария'
>>>
```

Вывод данных

1. Тип данных строчный

```
>>> print("Программа 'Game Over' 2.0")
Программа 'Game Over' 2.0
>>> print("Тоже", "самое", "сообщение")
Тоже самое сообщение
>>> print("Только",
    "чуть-чуть",
    "побольше")
Только чуть-чуть побольше
```

2. Вывод переменных

```
>>> a = 1
>>> b = 2
>>> print(a, '+', b, '=', a + b)
1 + 2 = 3
>>>
```

3.

sep – параметр, используемый в качестве разделителя

```
>>> a=1
>>> b=2
>>> c=a+b
>>> print(a, b, c, sep = ':')
1:2:3
>>>
```

4.

end – параметр, который указывает на то, что выводится после вывода всех значений, перечисленных в функции **print**.

```
>>> print(a, b, c, sep = '  ', end = '
')
```

Библиотека math

import math # подключение библиотеки **math**

1. math.sin(x) #вызов функции от одного аргумента

   y = math.sin(x) #использование функции в выражении

   print(math.sin(math.pi/2)) # вывод функции на экран

2. from math import *

   y = sin(x)

   print(sin(pi/2))

Библиотека math

Функция	Описание
Округление	
round(x)	Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
trunc(x)	Округление в сторону нуля (так же, как функция int).
fabs(x)	Модуль (абсолютная величина). Эта функция всегда возвращает значение типа float.

int(x), **round(x,n)**, **abs(x)** не требуют подключения модуля math

Библиотека math

Корни, степени, логарифмы	
sqrt(x)	Квадратный корень. Использование: sqrt(x)
pow(a, b)	Возведение в степень, возвращает a^b. Использование: pow(a,b)
exp(x)	Экспонента, возвращает e^x. Использование: exp(x)
log(x)	Натуральный логарифм. При вызове в виде log(x, b) возвращает логарифм по основанию b.
log10(x)	Десятичный логарифм
e	

Библиотека math

Тригонометрия	
sin(x)	Синус угла, задаваемого в радианах
cos(x)	Косинус угла, задаваемого в радианах
tan(x)	Тангенс угла, задаваемого в радианах
asin(x)	Арксинус, возвращает значение в радианах
acos(x)	Арккосинус, возвращает значение в радианах
atan(x)	Арктангенс, возвращает значение в радианах
atan2(y, x)	Полярный угол (в радианах) точки с координатами (x, y).

Библиотека math

(продолжение)

Тригонометрия	
hypot(a, b)	Длина гипотенузы прямоугольного треугольника с катетами a и b.
degrees(x)	Преобразует угол, заданный в радианах, в градусы.
radians(x)	Преобразует угол, заданный в градусах, в радианы.
pi	Константа π

Пример 1.

Даны a, b, k, m.

Определить:

$$C = \sqrt{\frac{(a-b)^2}{|k-m|}};$$

$$A = \sin(\pi/6)\cdot C^2 - \frac{C(a-b)}{a\cdot b\cdot k}.$$

Пример 1. (Исходный код)

```
# Линейная программа

a = int(input("Введите a = "))
b = int(input("Введите b = "))
k = int(input("Введите k = "))
m = int(input("Введите m = "))

from math import *

C = sqrt((a-b)**2/abs(k-m))
A = sin(pi/6)*C**2-C*(a-b)/(a*b*k)

print("C = ", C)
print("A = ", A)

input("\n\nНажмите Enter чтобы выйти.")
```

Строки

- Для объявления строк используются как одинарные, так и двойные кавычки - разницы нет;
- для многострочных текстов используется по три кавычки в начале и в конце - либо одинарные, либо двойные. Внутри можно использовать такие же кавычки.

```
>>> some_line = 'Hello OTUS'
>>> another_line = "Just some text"
>>> multiline = '''Hi!
... This is an example of a 'multiline' text
... Thanks, bye!'''
>>> some_line
'Hello OTUS'
>>> print(some_line)
Hello OTUS
>>> print(repr(some_line))
'Hello OTUS'
```

Индексы

Так как строка является типом коллекции, то к каждому символу можно обратиться по индексу.
Также поддерживаются срезы (slices) - из строки можно получить подстроку, указав через двоеточие необходимые индексы

```
 +---+---+---+---+---+---+
 | q | w | e | r | t | y |
 +---+---+---+---+---+---+
 0   1   2   3   4   5
 6
-6  -5  -4  -3  -2  -1

>>> some_str = "qwerty"
>>> some_str[-1]  # y - последний элемент
'y'
>>> some_str[0]   # q - первый
элемент
>>> some_str[-2]  # t - предпоследний элемент
'q'
't'
>>> some_str[1:4]  # wer - то, что между индексами 1 и 4
'wer'
>>> some_str[:3]  # qwe - с самого начала по 3
'qwe'
```

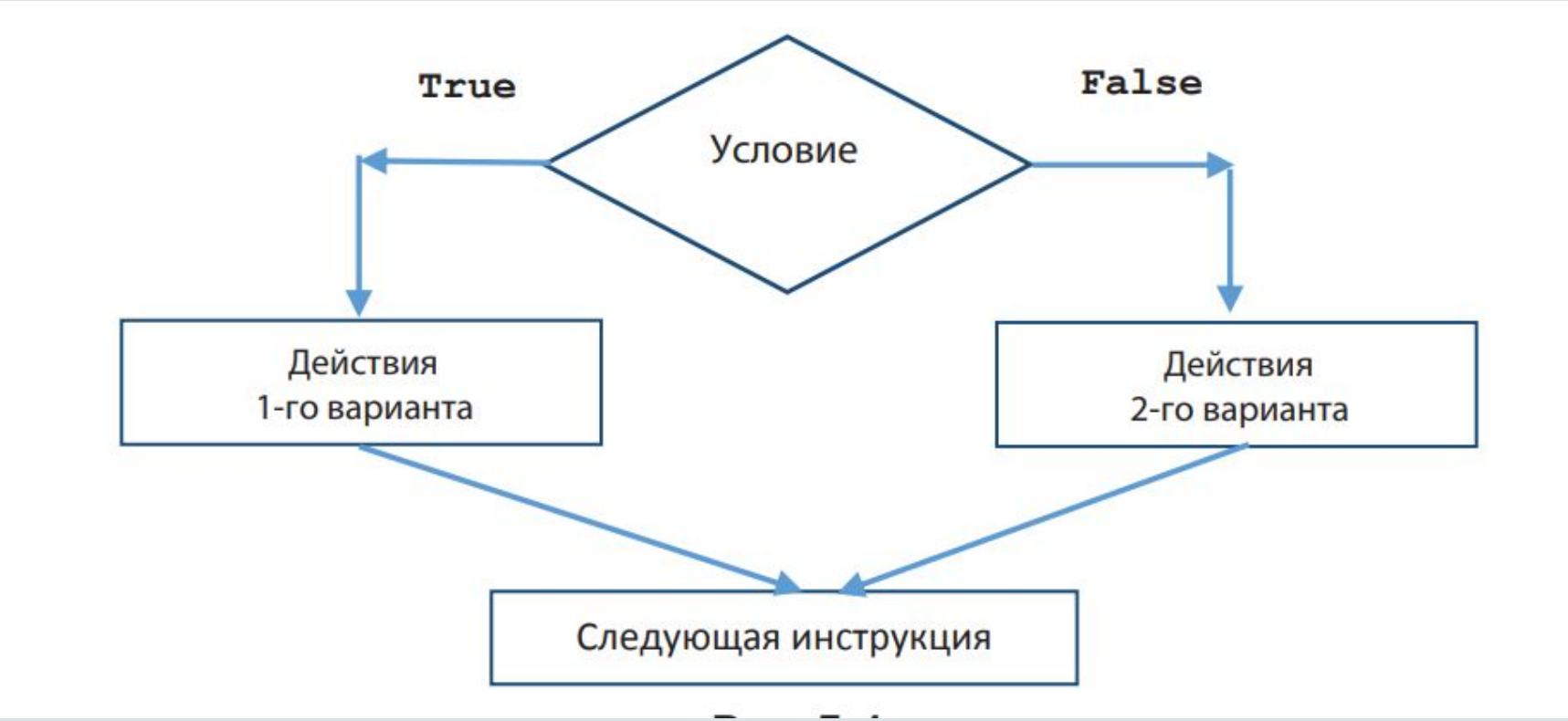
Списки (lists)

- Список — это упорядоченный массив объектов;
- список может содержать в себе любое количество любых объектов;
- списки поддерживают индексирование и срезы.

```
# Создаем массив. Для этого внутри прямоугольных скобок
# через запятую перечисляем значения
>>> first_9_fibonacci_numbers = [0, 1, 1, 2, 3, 5, 8, 13, 21]

# Доступ по индексу
>>> first_9_fibonacci_numbers[3]
2
>>> first_9_fibonacci_numbers[-1]
21

# Срезы возвращают новые списки
>>> first_9_fibonacci_numbers[3:5]
[2, 3]
>>> first_9_fibonacci_numbers[:10]
[0, 1, 1, 2, 3, 5, 8, 13, 21]

>>> first_9_fibonacci_numbers[10]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Условный оператор

Задача: **изменить порядок действий** в зависимости от выполнения некоторого условия.



```
if <условие>:
    <Действия 1-го варианта (1-я серия инструкций)>
else:
     <Действия 2-го варианта (2-я серия инструкций)>
```

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")
```

and «И»

or «ИЛИ»

not «НЕ»

Приоритет :

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

Пример. Даны два вещественных числа a и b. Если первое больше второго, то увеличить каждое число в 2 раза, иначе – уменьшить в два раза.

Соответствующая программа:

```
a = float(input('a = '))
b = float(input('b = '))
if a > b:
    a = a * 2
    b = b * 2
else:
    a = a/2
    b = b/2
print('a =', a)
print('b =', b)
```

Таблица

A	*B*	Not *A*	*A* And *B*	*A* Or *B*	*A* Xor *B*
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

Свойства условий:

1. **Сокращение длинных условий**

• использовать обратный слэш («\»):

if v < 400 and v != 2 and v != 3 and v != 12 and \ v != 13 and v != 22 and v != 23: ...

• взять все условие в скобки (перенос внутри скобок разрешен):

if (v < 400 and v != 2 and v != 3 and v != 12 and v != 13 and v != 22 and v != 23): ...

2.**Разрешены двойные неравенства,**

if A < B < C: ... означает то же самое, что и if A < B and B < C:

3.**Условие - логические функции**, то есть функции, возвращающие результат логического типа

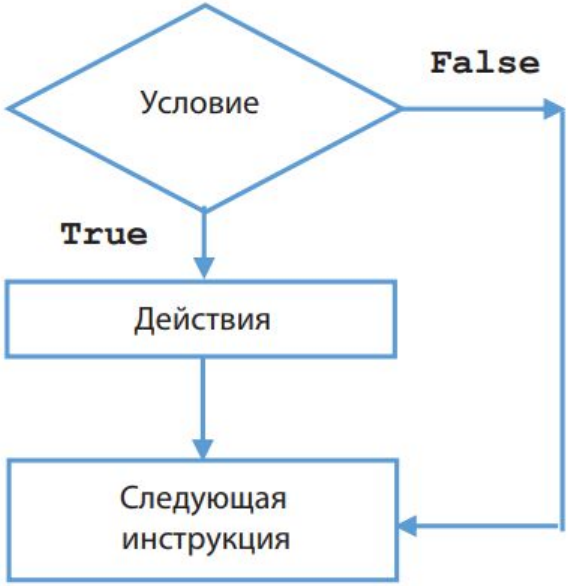
```
 n = int(input('Введите целое число '))
if Chet(n):
   print('Это число четное')
else:
   print('Это число нечетное')
```

где Chet() – функция, возвращающая результат True, если ее параметр (значение, указанное в скобках) является четным числом, и False – в противном случае;

4. **оператор in** (оператор проверки принадлежности), который проверяет, принадлежит ли некоторый объект (число, символ, переменная и т. п.) набору значений (списку, строке, диапазону чисел и т. п.):

```
a = 3
if a in range(10):
    sim = input('Введите символ ')
     s = input('Введите строку символов ')
if sim in s:
      Zvet = 'Зеленый‘
 if Zvet in Raduga:
```

Неполная форма




```
M = a
if b > a:
  M = b
```

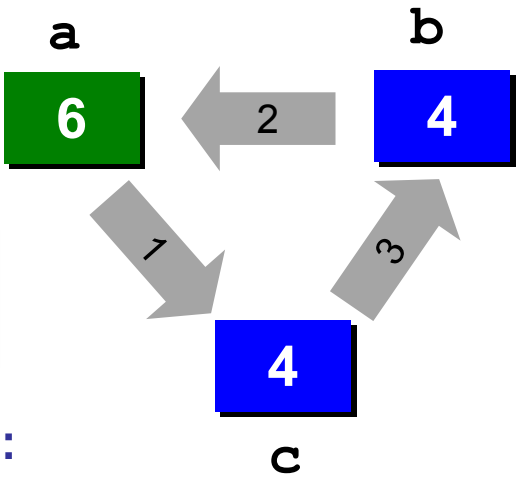
Решение в стиле Python:

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

```
M = a if a > b else b
```

```
if a > b:
  c = a
  a = b
  b = c
```

Что делает?



Можно ли обойтись без переменной **c**?

Решение в стиле Python:

```
a, b = b, a
```

Вложенные условные операторы

Задача: в переменных **a** и **b** записаны возрасты Андрея и Бориса. Кто из них старше?

```
if a > b:
  print("Андрей старше")
else:
  if a == b:
    print("Одного возраста")
  else:
    print("Борис старше")
```

вложенный условный оператор

Каскадное ветвление

```
if a > b:
  print("Андрей старше")
elif a == b:
  print("Одного возраста")
else:
  print("Борис старше")
```

! `elif` = `else if`

Каскадное ветвление

```
cost = 1500
if cost < 1000:
  print ( "Скидок нет." )
elif cost < 2000:
  print ( "Скидка 2%." )
elif cost < 5000:
  print ( "Скидка 5%." )
else:
  print ( "Скидка 10%." )
```

первое сработавшее условие

? Что выведет?

```
Скидка 2%.
```

Циклы в python

Функция range создаст список длинной в «n» элементов

a = range(5, 10)

числа с 5 до 9

b = list(range(1, 10, 2))

Список list [1, 3, 5, 7, 9]

```
for number in [0, 1, 2, 3, 4]:
    print(number)

for number in range(5):
    print(number)


for number in range(10):
    if number % 2 == 0:
        print(number)
```

```
i = 0
while i < 10:
    print(i)
    i = i + 1

while i < 10:
    print(i)
    if i == 5:
        break
    i += 1
```

```
my_list = [1, 2, 3, 4, 5]

for i   in my_list:
    if i == 3:
        print("Item found!")
        break
        print(i)
else:
       print("Item not found!")
```

Задачи:

1. Напечатать таблицу соответствия между весом в фунтах и весом в килограммах для значений 1, 2, ..., 10 фунтов (1 фунт = 453 г).

2. Напечатать таблицу умножения на 7:

1 x 7 = 7

2 x 7 = 14

...

9 x 7 = 63

3. Найти:

а) сумму всех целых чисел от 100 до 500;

б) сумму всех целых чисел от *a* до 500 (значение *a* вводится с клавиатуры; *a* 500);

в) сумму всех целых чисел от –10 до *b* (значение *b* вводится с клавиатуры; *b* –10);

4. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток будет через 3, 6, 9, ..., 24 часа, если первоначально была одна амеба.

5. Гражданин 1 марта открыл счет в банке, вложив 1000 руб. Через каждый месяц размер вклада увеличивается на 2% от имеющейся суммы. **Определить:**

а) прирост суммы вклада за первый, второй, ..., десятый месяц;

б) сумму вклада через три, четыре, ..., двенадцать месяцев.

6. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10% от пробега предыдущего дня. Определить:

а) пробег лыжника за второй, третий, ..., десятый день тренировок;

б) какой суммарный путь он пробежал за первые 7 дней тренировок.

*

«Странный муж» . Некий мужчина отправляется на работу, которая находится на расстоянии 1 км от дома. Дойдя до места работы, он вдруг вспоминает, что перед уходом забыл поцеловать жену, и поворачивает назад. Пройдя полпути, он меняет решение, посчитав, что правильнее вернуться на работу. Пройдя 1/3 км по направлению к работе, он вдруг осознает, что будет настоящим подлецом, если так и не поцелует жену. На этот раз, прежде чем изменить мнение, он проходит ¼ км. Так он продолжает метаться, и после *N*- этапа, пройдя 1/*N* км, снова меняет решение. Определить:

а) на каком расстоянии от дома будет находиться мужчина после 100-го этапа (если допустить, что такое возможно);

б) какой общий путь он при этом пройдет.

Спасибо за внимание!