Лекция 3

Язык программирования Python.

Хайрулин Сергей Сергеевич

email: s.khairulin@g.nsu.ru, s.khayrulin@gmail.com

Ссылка на материалы

План

- Лекции/практические занятия
  - Тест
- Дифференцированный зачет в конце семестра
  - Защита задания

Литература

Начальный уровень

- Mark Pilgrim. Dive into Python - http://www.diveintopython.net/
- Марк Лутц. Изучаем Python, 4-е издание // Символ-Плюс 2011.
- ...

Стандарт/Документация

- PEP-8 - https://www.python.org/dev/peps/pep-0008/
- https://www.python.org/
- https://github.com/python/cpython

Экспертный уровень

- Лучано Рамальо: Python. К вершинам мастерства
- Mitchell L. Model. Bioinformatics Programming Using Python // O'Reilly 2010.

Версии Python

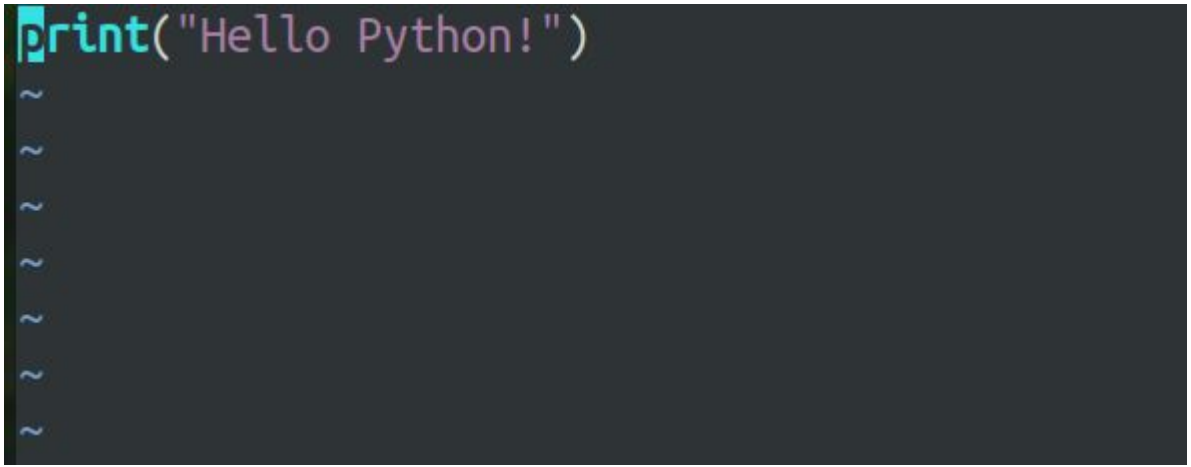
- Python 2 вышел 2010 году последняя версия 2.7.16 - исправлялись только баги(ошибки) с января 2020 года поддержка прекращена.
- Python 3 в появился в 2008, является актуальной версией языка. Текущая стабильная версия 3.8.5 -> в предрелиз 3.9, в разработке 3.10
  - Python 3 не гарантирует совместимости кода с Python 2

План занятия

- Как написать и запустить программу
- Алгоритмы
- Программные блоки
- Логические операторы
- Циклы
  - **while**
  - **for** итерирование над объектами
- Условные операторы
  - if
  - if … else
  - if … elif ...
- Практика

Как написать и запустить программу

```
[17:03:04] serg :: serg-pc  → ~/tmp»
vim my_first_script.py
```



Как написать и запустить программу

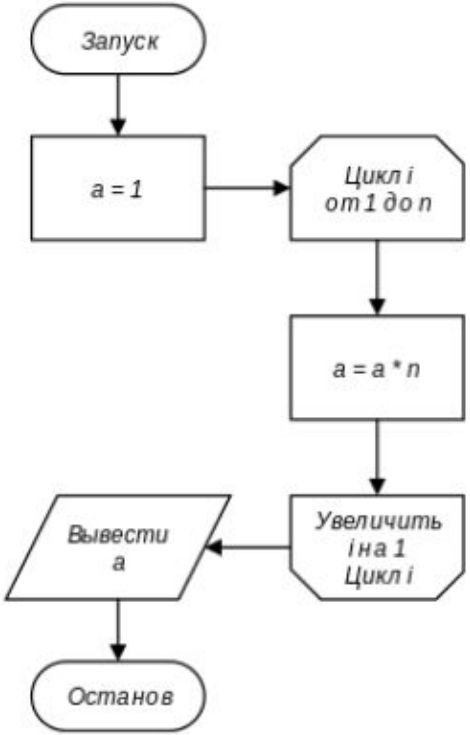
```
[17:06:19] serg :: serg-pc  → ~/tmp»
python my_first_script.py
Hello Python!
```

Алгоритмы

Алгори́тм — набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата. Независимые инструкции могут выполняться в произвольном порядке, параллельно, если это позволяют используемые исполнители.

(Википедия)

Алгоритмы


Запуск
a = 1
Цикл i
от 1 до n
a = a * n
Увеличить
i на 1
Цикл i
Вывести
a
Останов

Свойства

Конечность описания — любой алгоритм задается как набор инструкций конечных размеров, т. е. программа имеет конечную длину.

Дискретность — алгоритм выполняется по шагам, происходящим в дискретном времени. Шаги четко отделены друг от друга. В алгоритмах нельзя использовать аналоговые устройства и непрерывные методы.

Направленность — у алгоритма есть входные и выходные данные. В алгоритме четко указывается, когда он останавливается, и что выдается на выходе после остановки.

Массовость — алгоритм применим к некоторому достаточно большому классу однотипных задач, т. е. входные данные выбираются из некоторого, как правило, бесконечного множества.

Детерминированность (или конечная недетерминированность) — вычисления продвигаются вперед детерминировано, т. е. вычислитель однозначно представляет, какие инструкции необходимо выполнить в текущий момент. Нельзя использовать случайные числа или методы. Конечная недетерминированность означает, что иногда в процессе работы алгоритма возникает несколько вариантов для дальнейшего хода вычислений, но таких вариантов лишь конечное.

Алгоритм вычисления чисел Фибоначчи

$$F_1 = 1,\ F_2 = 1,\ \ldots,\ F_n = F_{n-1} + F_{n-2}$$

```
function Fibo(n)
    if n = 1 or n = 2
        return 1
    endif
    return Fibo(n - 1) + Fibo(n - 2)
endfunction
```

Программные блоки

```
3  int main(int argv, char** argc)
4  {
5      int a = 1;
6      if (a == 1) {
7          std::cout << a << std::endl;
8      }
9      return 0;
10 }
```

```
def main(argv, argc):
    a = 1
    if a == 1:
        print(a)
    return 0
```

Логические операторы

```
In [1]: 1 < 2
Out[1]: True

In [2]: 1 <= 1
Out[2]: True

In [3]: 1 >= 1
Out[3]: True

In [4]: 1 > 2
Out[4]: False

In [5]: 1 != 2
Out[5]: True

In [6]: 1 != 1
Out[6]: False

In [7]: 1 == 1
Out[7]: True
```

Логические операторы

```
In [24]: a = 1

In [25]: b = 1

In [26]: a is b
Out[26]: True
```

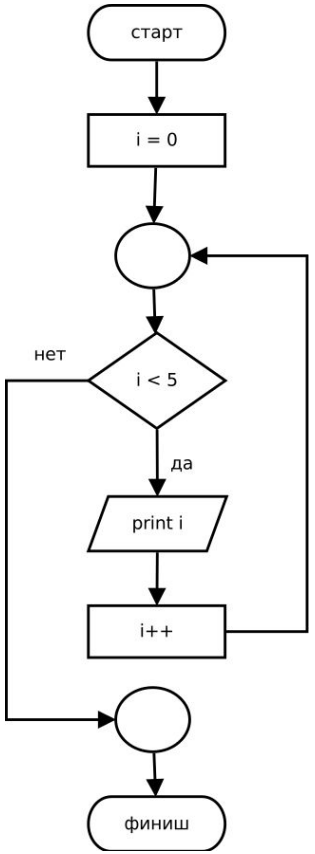
Логические операторы

```
In [27]: not (a is b)
Out[27]: False

In [28]: 1 == 1 and 1 > 0 or 3 > 2
Out[28]: True
```

Циклы

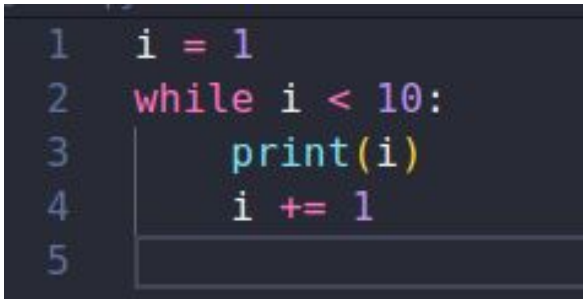
Цикл — разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.



while

Синтаксис

while <logic_expression>:
    do_smth



for

Синтаксис

for <var> **in** <collection>:
    do_some_work

```
7   l = [1,2,3,4,5,6,7,8,9]
8   for i in l:
9       print(l)
10
11  for i in range(1, 10, 1):
12      print(l)
```

Условные операторы

Синтаксис

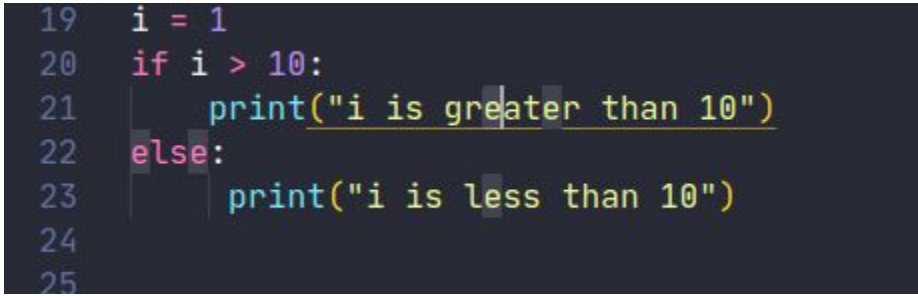
if logic_expression:
    do_some_work

```
15  i = 10
16  if i > 2:
17      print(1)
```

Условные операторы

Синтаксис

if logic_expression:
    do_some_work
else: # если выражение не верно
    do_other_work



Условные операторы

Синтаксис

if logic_expression:
    do_some_work
elif other_logic_expression**:**
    do_other_work
else:
    do_smth_else

```
26  i = 20
27  if i > 30:
28      print("i is greater than 30")
29  elif  i < 30 and i >= 20:
30      print("i is less than 30 but it greater or equal to 20")
31  else:
32      print("To small value for i")
```

Практическая Часть

1. Циклы
2. Условные операторы
3. Циклы + Условные операторы

К этому заданию нужно приступать после выполнения заданий выше
Реализовать программу чат бот. Программа должна уметь общаться с пользователем реагировать на набор заданных фраз фраз:

```
[19:18:57] serg :: serg-pc  → ~/tmp»
python3 chat_bot.py
Put you'r question here: привет
Hello
Put you'r question here: Как дела?
Super! Thanks for asking
Put you'r question here: ты кто?
I don't understand. Please rephrase your message.
Put you'r question here: пока
Bye
[19:19:25] serg :: serg-pc  → ~/tmp»
```