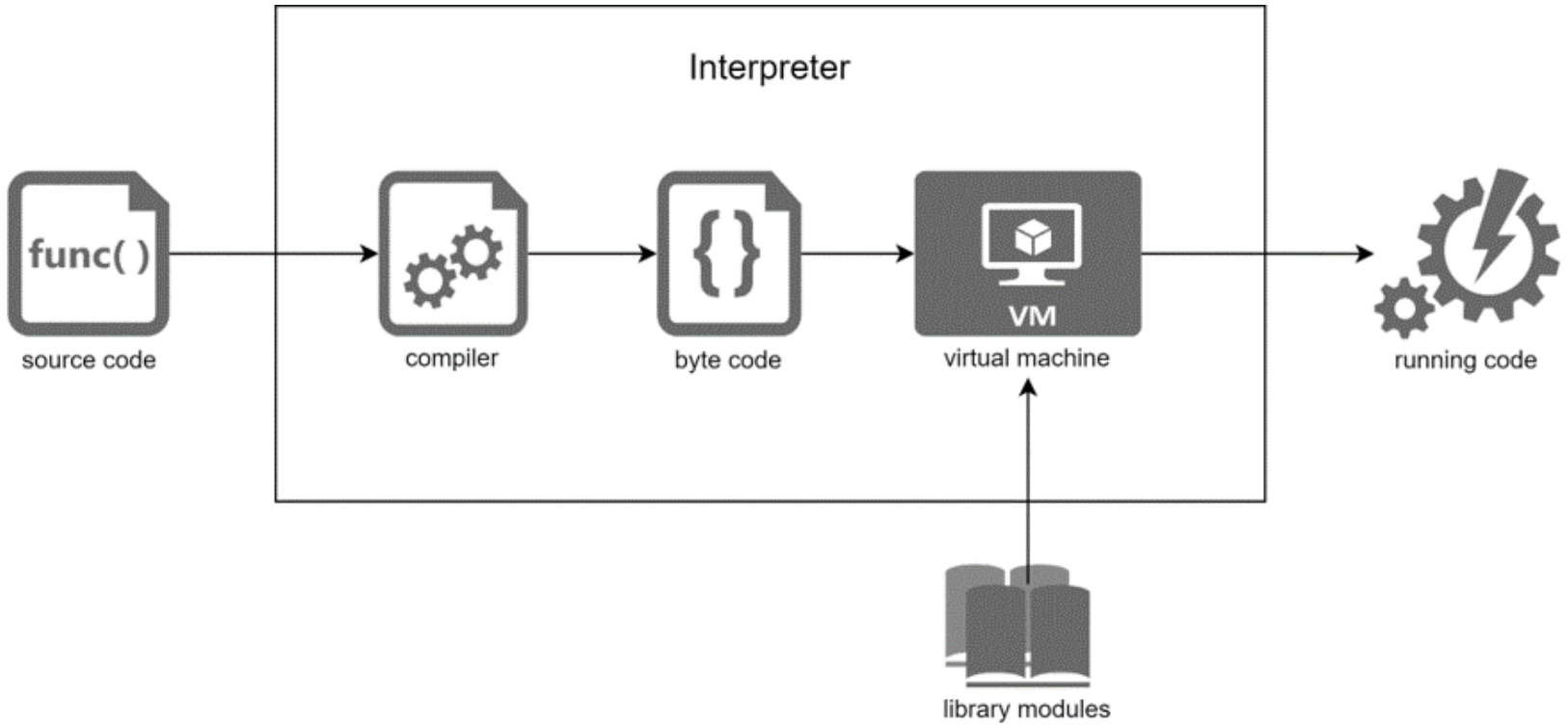
Базовые концепции

Интерпритатор – программная прослойка между кодом и процессором



Фундамент синстаксиса

Литералы

10, **15.5**, **“some text”** - литералы, буквальная запись данных (в памяти данные всегда хранятся в бинарной форме, однако, запись может влиять на то, каким именно образом эти данные хранятся и обрабатываются);

Переменные

x = 0 – ассоциация значения и переменной;

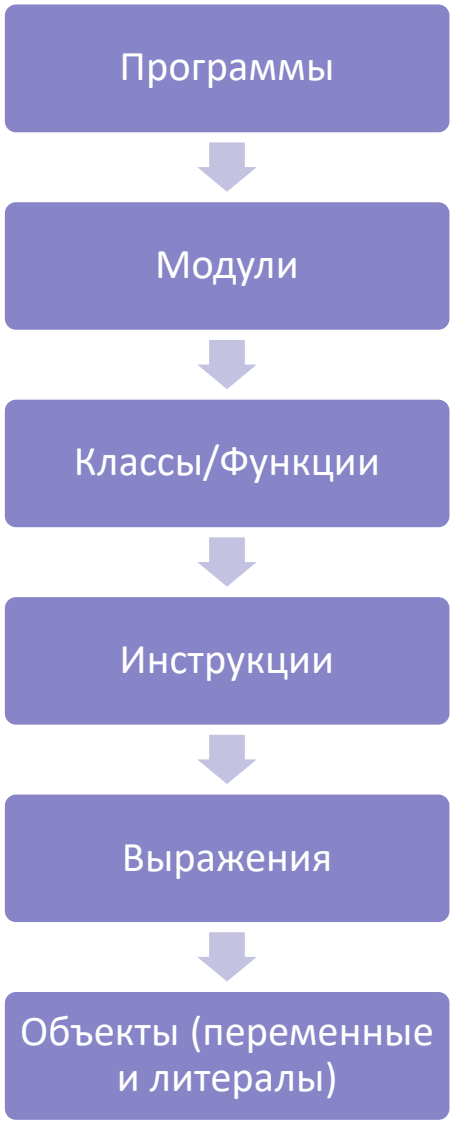
переменная в Python это не контейнер объекта, а только ссылка на сам объект в памяти, таких ссылок может быть сколько угодно и они не зависят от внутренного содержимого объекта;

Выражения

x + 7 – выражение, может быть сведено к некоторому объекту (результату вычисления выражения), может включать в себя литералы и переменные, а так же результаты некоторых инструкций;

Инструкции

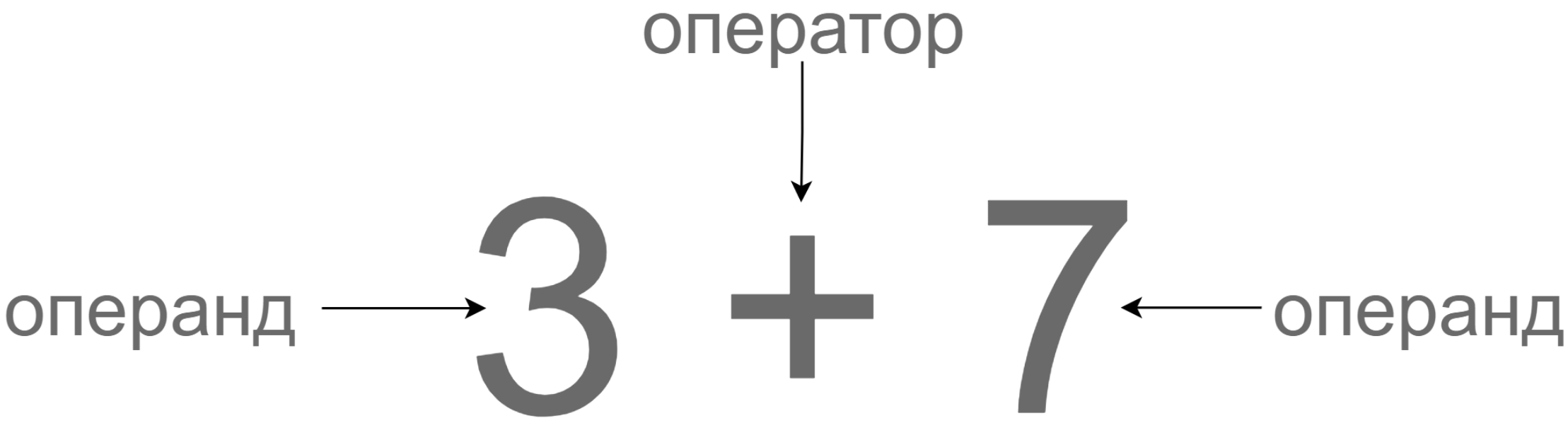
print(x) – инструкция, представляет из себя некоторое действие, может включать в себя литералы, переменные и выражения;



Концептуальная иерархия

Самая важная единица иерархии это объект, в Python почти всё является объектом первого рода/класса

Операторы и операнды



val = “test” - стандартная форма

x, y = 3, 4 - позиционное

[x, y] = [3, 4] - списком

c, a, t = “cat" - последовательностью

x += 1 - инкрементивное

Оператор присваивания

В Python оператор ‘=‘ отвечает и за инициализацию переменной, и за присваивание значения

Правила именования переменных

Имена переменных должны начинаться с подчёркивания либо буквы, за которой может следовать любое количество подчёркиваний, букв или цифр: **_test45, xyz, _42smile**, ~~**&^%#**~~

Регистр симлов имеет значение: **TEST ≠ test**

Зарезервированные слова: **False, None, True, and, as, assert, async, await, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield**

** степень

% остаток

// целочисленное деление

/ деление

* произведение

- вычитание

+ сложение

Математические операторы

Указаны в соответствии с их приоритетом

== равно

!= не равно

> больше

< меньше

>= больше либо равно

<= меньше либо равно

Операторы сравнения

```
operator1 condition:
        operator1_body_logic
        …
        operator2 condition:
                operator2_body_logic
        …
        operator1_body_logic
        …
```

Блоки

определяют логику сложных конструкций (например, условных или циклических),

отделяются друг от друга отступами

Характерные черты синтаксиса

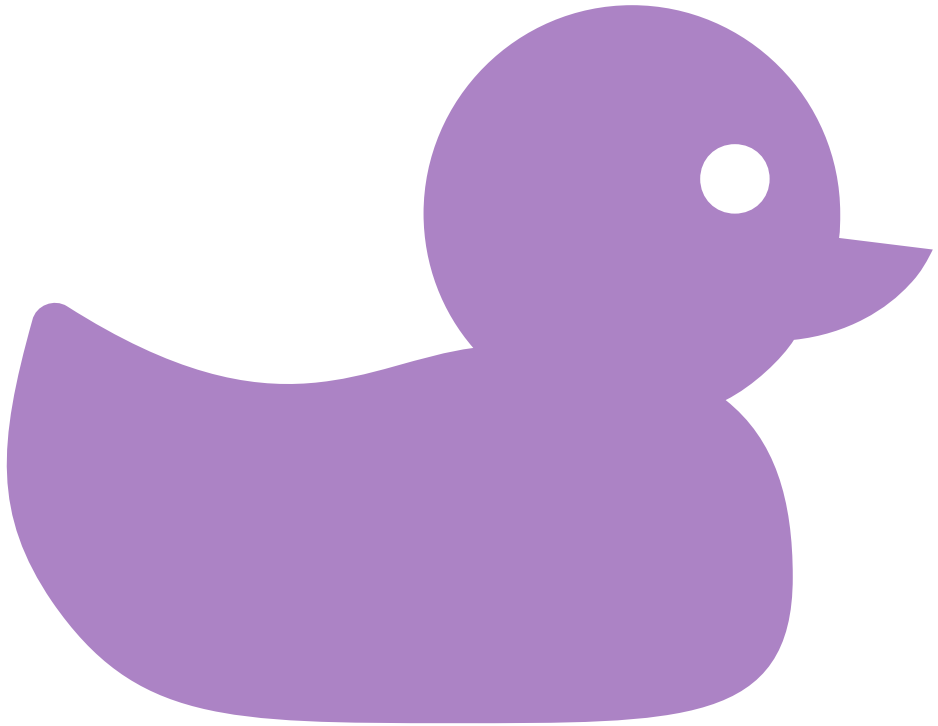
Операторы выполняются друг за другом, если не указано иное

Границы блоков и операторов определяются автоматически

Составные операторы = заголовок + : + тело с отступом

Пустые строки, пробелы и комментарии игнорируются

Строки документации игнорируются, но отображаются специальными инструментами (IDE)



Коротко о динамической типизации

«Утиная» типизация – если это крякает, как утка, и плавает, как утка, то это утка