2. Типизация в языках программирования (статическая и динамическая, явная и неявная, сильная и слабая). Реализация типизации в Python.

Языки программирования делят на 2 типа: *типизированные* и *нетипизированные*. К первому например относятся C, Python, Scala, PHP и Lua, а ко второму — язык ассемблера, Forth и некоторые другие.

Типизированные языки разделяются еще на несколько пересекающихся категорий, которые и расписаны ниже.

Статическая / динамическая типизация. Статическая типизация - переменная связывается с типом в момент объявления и тип не может быть изменен позже:
C++ со статической типизацией:

```
int main(){
    int y;
    y = 2.1;
    //Ошибка!
}
```

Python с динамической типизацией:

```
y = 1
y = "MEOW"
print(y)
```

Достоинства:

- хороша для написания сложного, но быстрого кода;
- хорошо работает автодополнение в IDE;
- многие ошибки исключаются на стадии компиляции.

Недостатки:

- многословный код;
- сложность написания обобщенных алгоритмов и универсальных коллекций;
- сложности с работой с данными из внешних источников.

Сильная / слабая типизация. Сильная типизация выделяется тем, что язык не позволяет смешивать в выражениях различные типы и не выполняет автоматические неявные преобразования, например нельзя вычесть из строки множество. Языки со слабой типизацией выполняют множество неявных преобразований автоматически, даже если может произойти потеря точности или преобразование неоднозначно.
Слабая типизация в JS:

```
var check = "КОТ МЯУ";
if (8.3/check != NaN) {
    console.log("Мы только что разделили double на КОТА");
}
```

```
console.log(8.3/check) //Будет NaN
```
Сильная типизация в Python:
```
check = "КОТ МЯУ"
print(8.3/check) #Ошибка TypeError, 8.3 на кота не разделить
```
Преимущества сильной типизации:

- Надежность
- Скорость
- Понимание работы программы
- Определенность.

Преимущества слабой типизации:

- Удобство использования смешанных выражений (например из целых и вещественных чисел).
- Абстрагирование от типизации и сосредоточение на задаче.
- Краткость записи.

Явная и неявная типизация
Тип новых переменных / функций / их аргументов в ЯП с явной типизацией нужно задавать явно.
C++ с явной типизацией:
```
int y = 2;
```
Python с неявной типизацией:
```
y = 2
```
Достоинства явной типизации:

- Наличие у каждой функции сигнатуры (например int add(int, int)) позволяет без проблем определить, что функция делает.
- Программист сразу записывает, какого типа значения могут храниться в конкретной переменной, что снимает необходимость запоминать это.

Достоинства неявной типизации

- Сокращение записи — def add(x, y) короче, чем int add(int x, int y).
- Устойчивость к изменениям. Например если в функции временная переменная была того же типа, что и входной аргумент, то в явно типизированном языке при изменении типа входного аргумента нужно будет изменить еще и тип временной переменной.

А что там с Python в итоге?

— Язык Python имеет сильную типизацию. Т.е. не позволяет смешивать в выражениях различные типы и не выполняет автоматические неявные преобразования (кроме некоторых особых случаев). Основной особый случай: арифметические выражения с различными типами чисел (int, float, complex) - тут

производится автоматическое преобразование к наиболее сложному типу из участвующих в выражении.
— В языке Python используется динамическая типизация, т.е. "тип переменной" (являющейся по сути ссылкой на объект) может меняться во время ее жизни.
— В Python используется неявная типизация т.к. Мы не объявляем тип переменной по типу: