So...

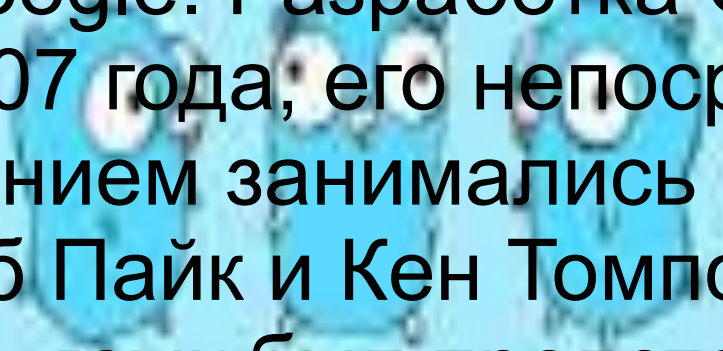
Why Golang?

Go — относительно молодой, но популярный язык программирования. По данным опроса Stack Overflow, именно Golang получил третье место в рейтинге языков программирования, которые хотели бы освоить разработчики.

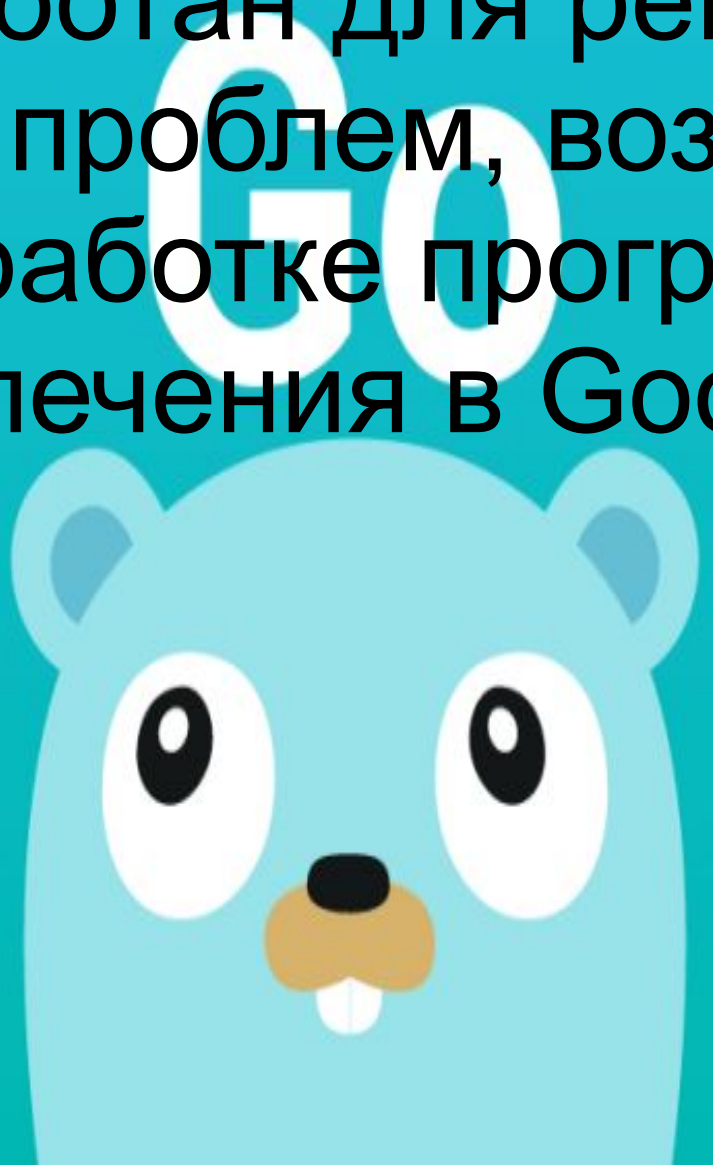
GO ADOPTION - GLOBAL COMPANIES

Go (часто также Golang)

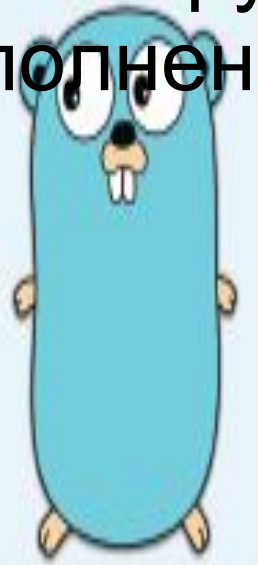
Компилируемый многопоточный язык программирования, разработанный внутри компании Google. Разработка Go началась в сентябре 2007 года, его непосредственным проектированием занимались Роберт Гризмер, Роб Пайк и Кен Томпсон. Официально язык был представлен в ноябре 2009 года.



По словам Роба Пайка, «Go был разработан для решения реальных проблем, возникающих при разработке программного обеспечения в Google».



- **Go** может рассматриваться как попытка создать замену языку Си. При разработке уделялось особое внимание обеспечению высокоэффективной компиляции. Программы на Go компилируются в объектный код и не требуют для исполнения виртуальной машины.



Golang

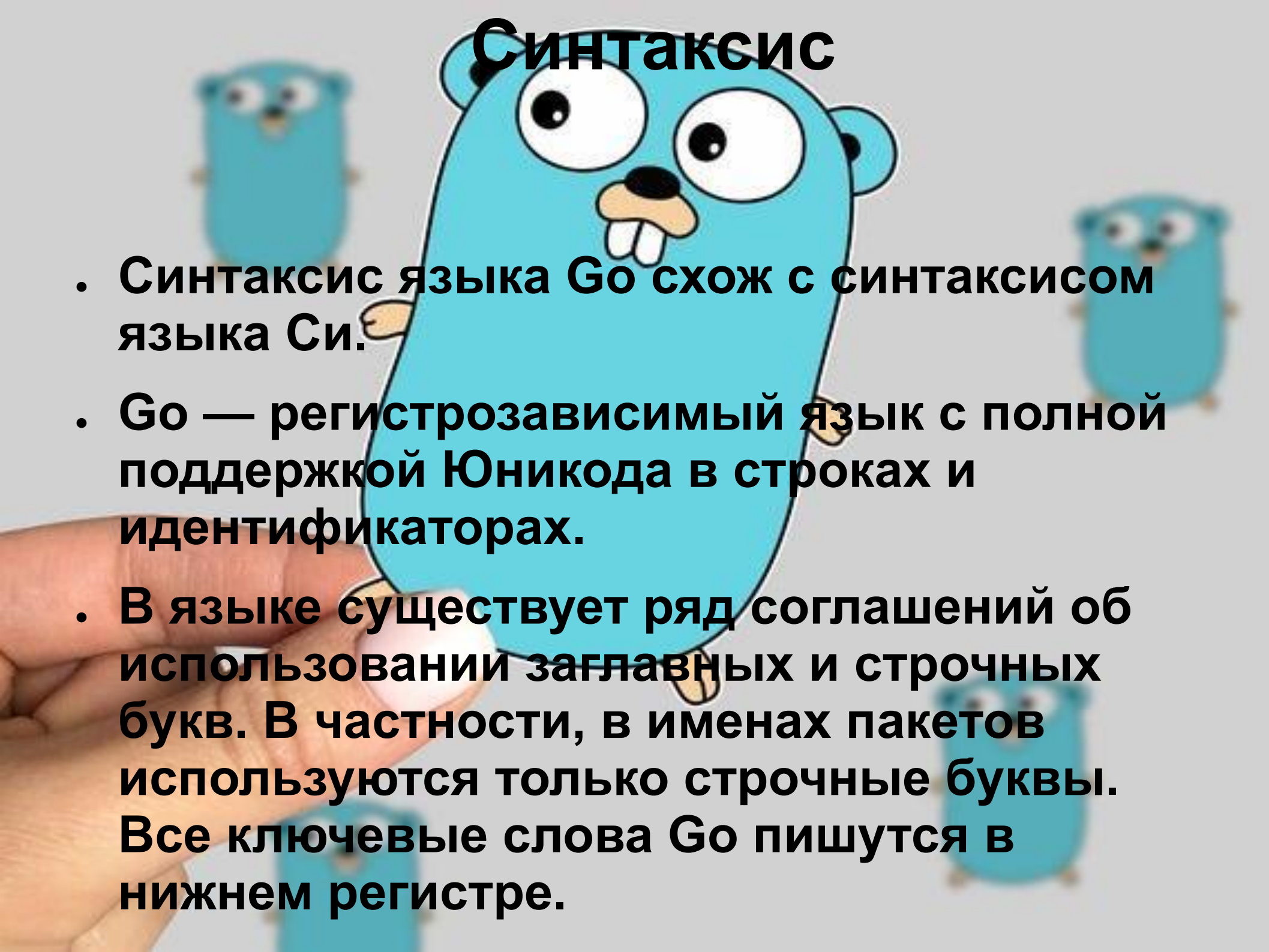
Vs

Python

Основные возможности языка Go

- **Go — язык со строгой статической типизацией. Доступен автоматический вывод типов, для пользовательских типов — «утиная типизация».**
- **Полноценная поддержка указателей, но без возможности применять к ним арифметические операции, в отличие от C/C++/D.**
- **Строковый тип со встроенной поддержкой юникода.**
- **Использование динамических массивов, хеш-таблиц, срезов (слайсов), вариант цикла для обхода коллекции.**
- **Средства функционального программирования: неименованные функции, замыкания, передача функций в параметрах и возврат функциональных значений.**
- **Автоматическое управление памятью со сборщиком мусора.**
- **Средства объектно-ориентированного программирования, но без поддержки наследования реализации (наследуются только интерфейсы). По большому счёту, Go является процедурным языком с поддержкой интерфейсов.**
- **Средства параллельного программирования: встроенные в язык потоки (go routines), взаимодействие потоков через каналы и другие средства организации многопоточных программ.**
- **Достаточно лаконичный и простой синтаксис, основанный на Си, но существенно доработанный, с большим количеством синтаксического сахара.**

Синтаксис



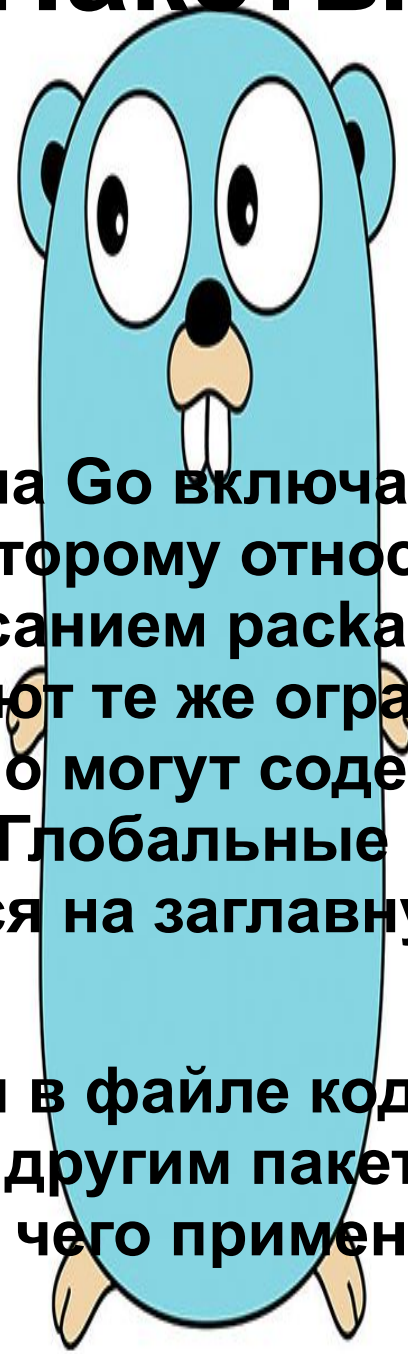
- Синтаксис языка Go схож с синтаксисом языка Си.
- Go — регистрозависимый язык с полной поддержкой Юникода в строках и идентификаторах.
- В языке существует ряд соглашений об использовании заглавных и строчных букв. В частности, в именах пакетов используются только строчные буквы. Все ключевые слова Go пишутся в нижнем регистре.

hello.go ×

```go
package main
import "fmt"

func main() {
    fmt.Printf("hello, world\n")
}
```

Пакеты



- Любая программа на Go включает один или несколько пакетов. Пакет, к которому относится файл исходного кода, задаётся описанием package в начале файла. Имена пакетов имеют те же ограничения, что и идентификаторы, но могут содержать буквы только нижнего регистра. Глобальные объекты, имена которых начинаются на заглавную букву, являются экспортируемыми.
- Для использования в файле кода Go объектов, экспортированных другим пакетом, пакет должен быть импортирован, для чего применяется конструкция import.

Горутина (goroutine) — это функция, выполняющаяся конкурентно с другими горутинами в том же адресном пространстве.

- Запустить горутину очень просто:

```
go normalFunc(args...)
```

- Функция normalFunc(args...) начнет выполняться асинхронно с вызвавшим ее кодом.
- Обратите внимание, горутины очень легковесны. Практически все расходы — это создание стека, который очень невелик, хотя при необходимости может расти.

Обработка ошибок и исключительных ситуаций

- Язык Go не поддерживает типичного для большинства современных языков синтаксиса структурной обработки исключений, предполагающего генерацию исключений специальной командой (обычно throw или raise) и их обработку в блоке try-catch). Вместо этого рекомендуется использовать возврат ошибки как одного из результатов функции (что достаточно удобно, так как в Go функция может возвращать более одного значения):

```go
func ReadFile(srcName string) (result string, err error) {
    file, err := os.Open("file.txt")
    if err != nil {
        // Генерация новой ошибки с уточняющим текстом
        return nil, fmt.Errorf("Ошибка при чтении файла %s: %g\n", srcName, err)
    }
    return result, nil // Возврат результата и пустой ошибки, если выполнение успешно
}
```

So...

Why Golang?

- https://ru.wikipedia.org/wiki/Go
- https://habr.com/ru/post/141853/
- https://habr.com/ru/post/219459/

- https://tour.golang.org/welcome/1