

# ЗНАКОМСТВО СО SWIFT В PLAYGROUND. ПЕРЕМЕННЫЕ И КОНСТАНТЫ. ЧИСЛОВЫЕ И СТРОКОВЫЕ ТИПЫ.



НИКИТА КАЗАКОВ



# НИКИТА КАЗАКОВ

Старший iOS-разработчик





# План занятия

1. [Запуск Playground](#)
2. [Описание интерфейса Playground](#)
3. [Наш первый код](#)
4. [Структура программы](#)
5. [Переменные и константы](#)
6. [Типы данных](#)
7. [Числовые типы данных](#)
8. [Строковые типы данных](#)

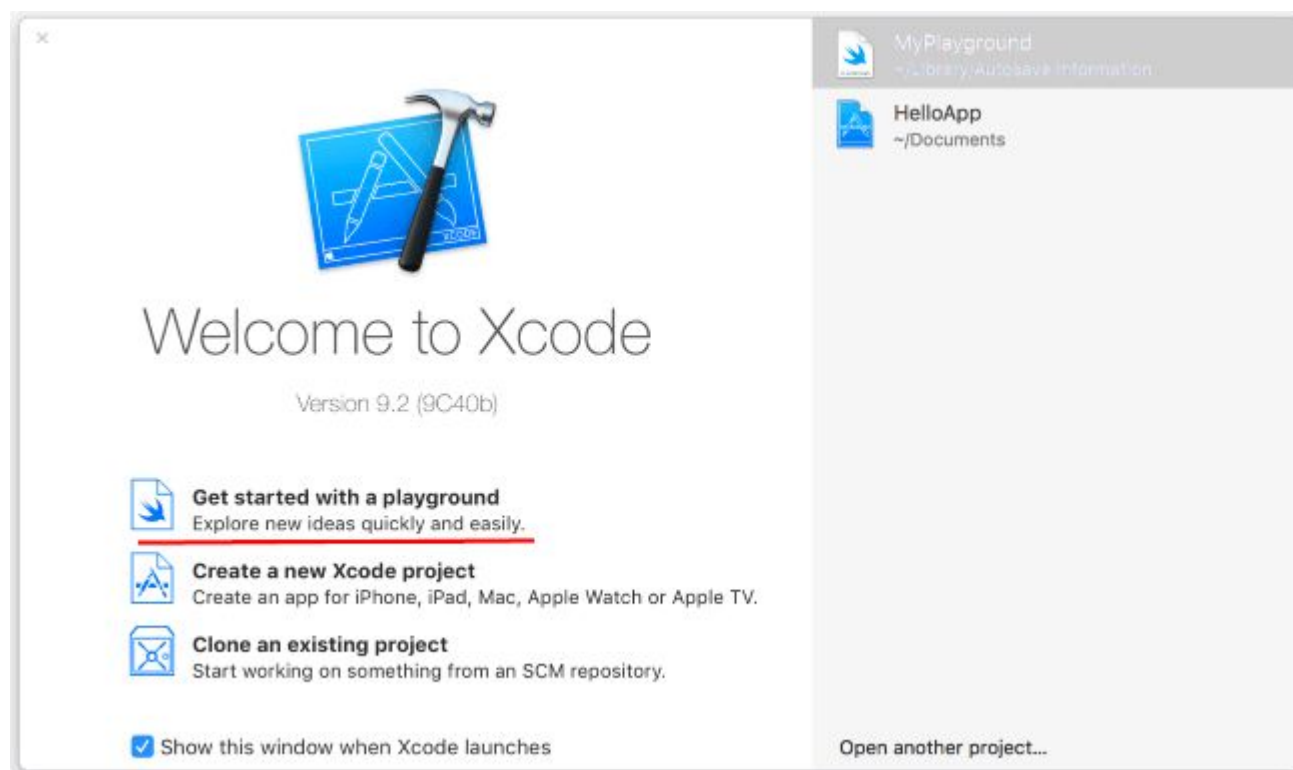


## Вспоминаем прошлые занятия

- Какой язык основной для iOS-разработки?
- В какой среде разработки мы будем писать код?

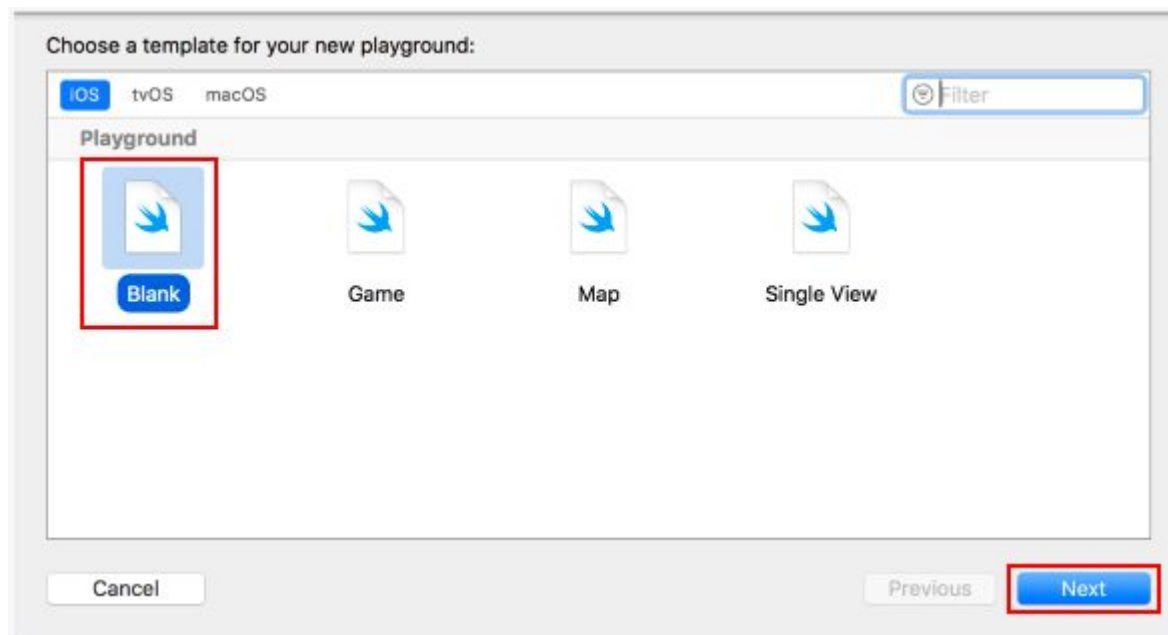
# Запуск Playground

Запускаем Xcode. Выбираем «Get started with a playground»:



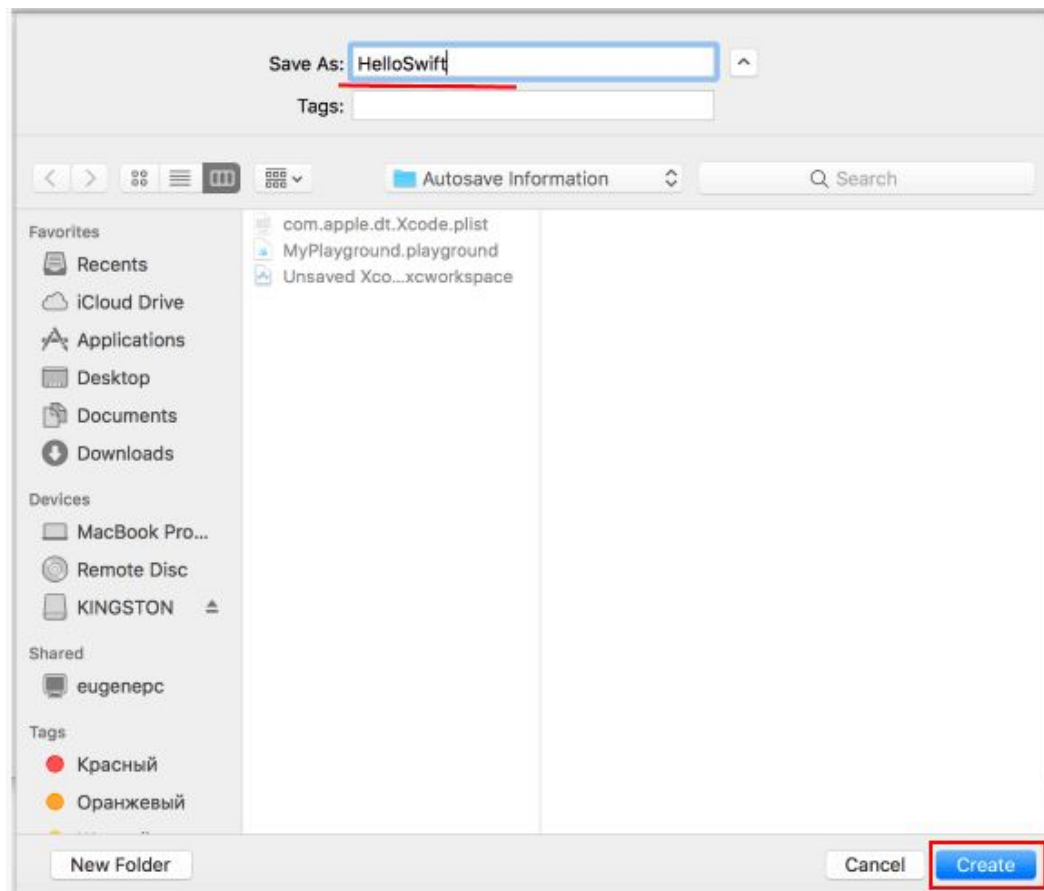
# Запуск Playground

Выбираем тип проекта Blank:



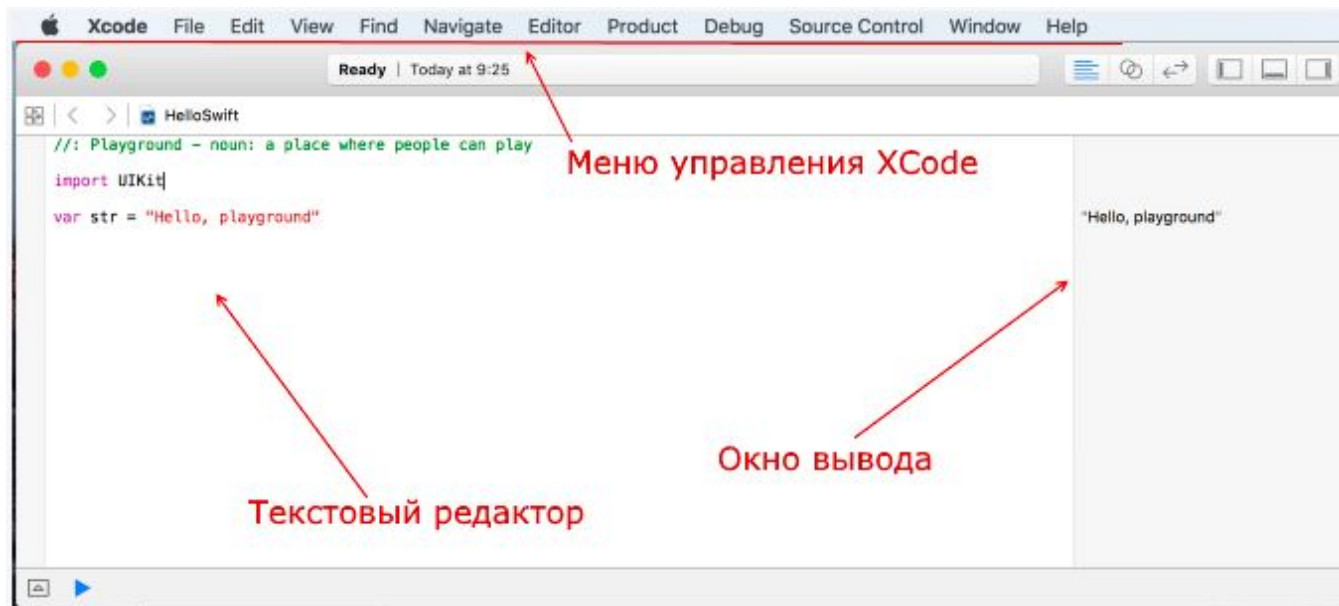
# Запуск Playground

Указываем имя проекта и нажимаем Create:



# Описание интерфейса Playground

Окно Playground напоминает текстовый редактор, в нем мы будем изучать базовые понятия языка Swift. Вот основные блоки Playground:







# Описание интерфейса Playground

- Меню управления Xcode – стандартное меню macOS программы;
- Текстовый редактор – поле для ввода кода;
- Окно вывода – вывод выполнения функций, значения переменных в real-time;
- Status bar – идентификатор работы playground (загрузка, исполнение, готово);
- Show/Hide Debug Area – Показать/Скрыть консоль;
- Консоль – вывод результата работы программы при использовании консоли.

# Наш первый код

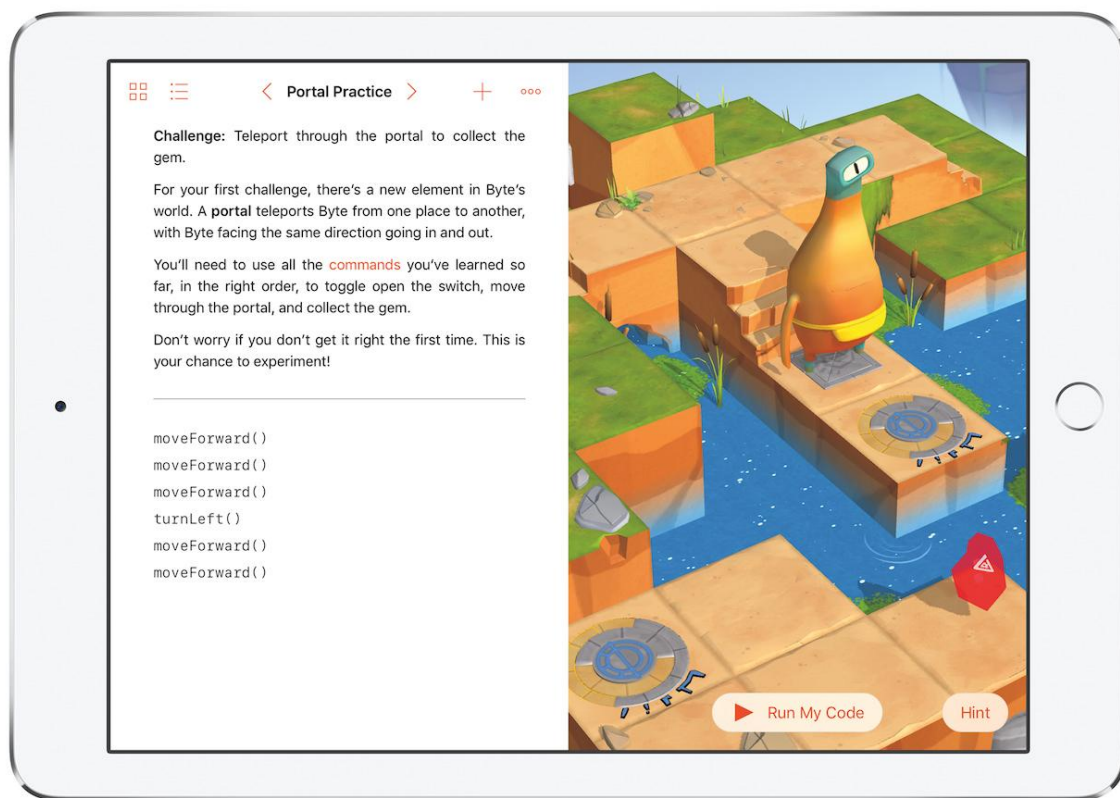
Запишем в текстовый редактор следующий код:

```
print("Hello world!")
```



# Приложение Swift Playgrounds

Swift Playgrounds — интерактивное приложение на iPad и Mac от Apple для изучения основ Swift.



# Структура программы на Swift

Программа состоит из множества команд, каждая из команд называется инструкцией (statement). Например, мы с вами уже использовали данную инструкцию:

```
print("Hello world!")
```

Каждая инструкция пишется с новой строки:

```
print("hello world!")  
print("welcome to swift")
```

# Структура программы на Swift

Swift имеет C-подобный синтаксис, т.е. родственен языкам C, C++, C#, Java.

```
print("hello world!"); print("welcome to swift")
```

Структурные блоки в Swift обозначаются фигурными скобками. Например:

```
class Book {      // начало блока класса
    func print() { // начало блока функции
        print("печать книги")
    }             // конец блока функции
}                 // конец блока класса
```

# Комментарии

В Swift можно писать комментарии к коду: однострочные и многострочные.

```
/*  
    Первая программа на Swift  
    печать строки hello world  
*/  
print("hello world!")
```

```
// Печать строки hello world  
print("hello world!")  
  
// Печать строки welcome to swift  
print("welcome to swift")
```

# Переменные и константы (var let)

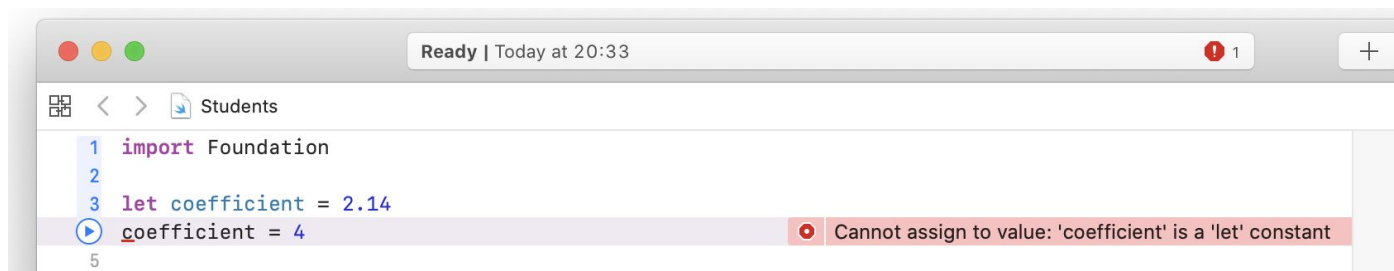
Переменные и константы — примитивы языка.

```
var coefficient = 12.7

// Переменную можно менять!
coefficient = 12.3

let pi = 3.14
```

Если мы попытаемся переопределить константу, то получим ошибку:



# Именованние переменных и констант

Стоит избегать однобуквенных или символьных названий:

```
let 😊 = "Smile"  
var c = 3
```

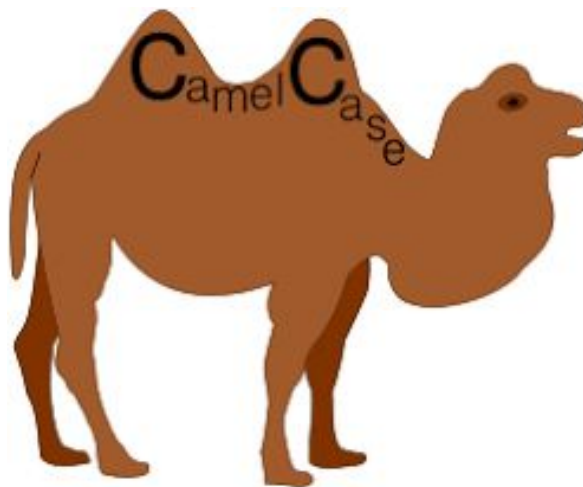
Лучше использовать стиль именованния CamelCase:

```
let userName = "Ivan"
```



---

Почему стиль написания кода носит название Camel Case?



# Переменные

Переменная — объект в программе для хранения значений.

```
var name: String
var count: Int
var isLoading: Bool
```

В примере используются основные типы данных:

- `Int` — целое числовое значение;
- `String` — строковое значение, в котором хранятся символы;
- `Bool` — логическая переменная, ее значение может быть истина или ложь.

# Переменные

**Инициализация переменных** — это присвоение переменным конкретного значения. Можно инициализировать в момент объявления, либо присвоением значения в инициализаторе.

```
var name: String = "Vadim"  
var count: Int  
var isLoading: Bool  
  
init(count: Int, isLoading: Bool) {  
    self.count = count  
    self.isLoading = isLoading  
}
```

# Задача

Даны два числа A и B. Нужно поменять значения местами и вывести в консоль, чему они равны до и после замены.

```
var a: Int = 15
var b = 32

print("До A = \(a)")
print("До B = \(b)")

var c = a
a = b
b = c

print("После A = \(a)")
print("После B = \(b)")
```

# Типы данных

Типы данных бывают строковые, числовые и логические.

Пример:

```
var name: String  
var count: Int  
var isLoading: Bool
```

# Числовые типы данных

- `Int` — целое число;
- `Double` — 64-битное число с плавающей точкой, содержит до 15 чисел в дробной части;
- `Float` — 32-битное число с плавающей точкой, содержит до 6 чисел в дробной части.

Разница между Double и Float:

```
9 var latitude: Double
10 latitude = 36.166667
11
12 var longitude: Float
13 longitude = -86.783333
14
15 longitude = -86.783333
16 longitude = -186.783333
17 longitude = -1286.783333
18 longitude = -12386.783333
19 longitude = -123486.783333
20 longitude = -1234586.783333
21
22
```

36.166667

-86.78333

-86.78333

-186.7833

-1286.783

-12386.78

-123486.8


-1234587

Тип Double получает более корректное отображение числа в отличие от Float:

```
var longitude: Double
```


В данном примере мы видим основное отличие между типами Double и Float. Отличие заключается в отображении числа и его дробной части.

Однако при попытке ввода большого числа вроде 987654321,987654321 результат будет округлен до 987654321,9876543.



Большие и длинные числа мы можем указывать в коде следующим образом:

```
let starsCount = 2_000_000_000_000_000
```



\* Необходимо быть аккуратным при работе с числовыми типами данных, особенно при работе с приложениями, требующими высокой точности (например, банковские приложения). Далее мы убедимся в этом.

Более подробно об этом можно почитать здесь:

<https://losthuman.ru/categories/code/2018/10/07/swift-tipy-i-operatsii-chast-2.html>



# Операции с числовыми данными

- Сложение +

```
var a = 14  
var b = 16  
var c = a + b // 30
```

\* Складывать можно не только числа, но и строки

- Вычитание -

```
var a = 18  
var b = 5  
var c = a - b // 13
```

- Унарный минус. Умножение на `-1`. Данная операция используется для смены знака значения переменной.

```
var a = -8  
var b = -a // 8  
var c = -b // -8
```

- Умножение `*`

```
var a = 3  
var b = 4  
var c = a * b // 12
```

- Деление `/`

```
var a = 12  
var b = 3  
var c = a / b // 4
```

---

При делении стоит учитывать, какие данные участвуют в делении, и какой результат мы хотим получить.

Например, в следующем случае выполняется деление дробных чисел:

```
let n : Double = 14
let d : Double = 4
let x : Double = n / d // 3.5
```

Результатом операции с числами `Double` является значение типа `Double`, которое равно `3.5`. Но если мы возьмем значения типа `Int`, то результат будет иным:

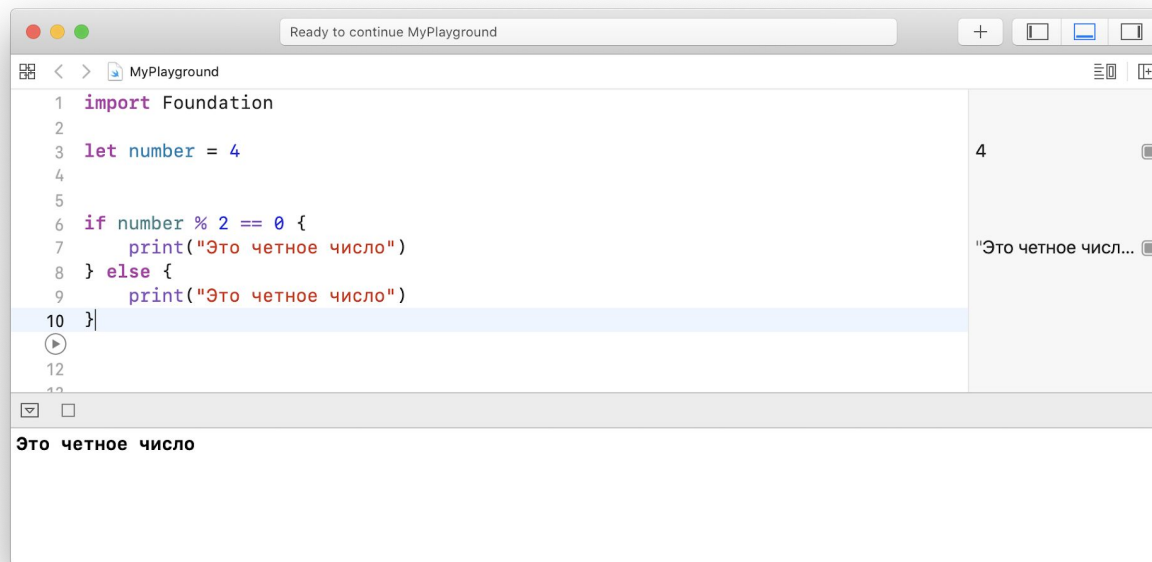
```
let n : Int = 14
let d : Int = 4
let x : Int = n / d // 3
```

При делении целочисленных значений дробная часть отбрасывается.

- Остаток от деления %

```
var a = 11
var b = 3
var c = a % b // 2
```

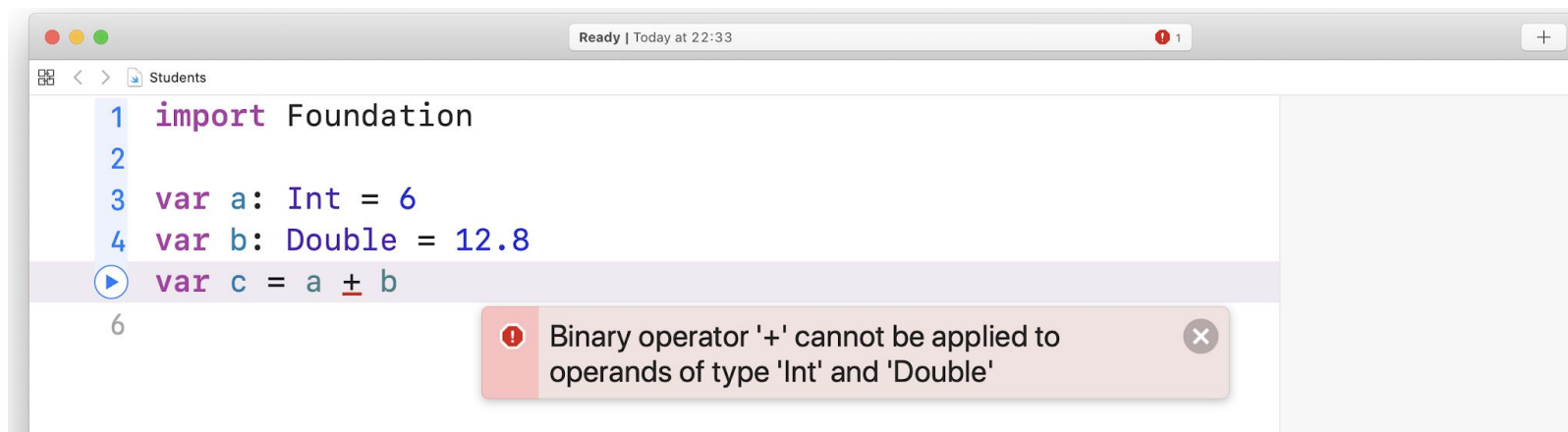
Так решаются задачи на определение четности числа:



# Арифметические операции с присваиванием

Любые действия с переменными можно делать, если они одного типа данных. Также работайте и с арифметическими операциями.

Если мы не будем соблюдать это правило, то получим ошибку:



The screenshot shows a code editor window titled 'Students' with a status bar indicating 'Ready | Today at 22:33' and '1' error. The code is as follows:

```
1 import Foundation
2
3 var a: Int = 6
4 var b: Double = 12.8
5 var c = a + b
```

The line `var c = a + b` is highlighted with a blue play button icon. Below the code, a red error message box is displayed:

Binary operator '+' cannot be applied to operands of type 'Int' and 'Double'

---

Есть арифметические операции, которые выполняются с присваиванием:

- `+=`

```
var a = 16
a += 10
print(a) // 26
// Аналогичная запись
// a = a + 10
```

- `-=`

```
var a = 20
a -= 6
print(a) // 14
// Аналогичная запись
// a = a - 6
```

- `*=`

```
var a = 20
a *= 6
print(a) // 120
// Аналогичная запись
// a = a * 6
```

---

- /=

```
var a = 20  
a /= 4  
print(a) // 5  
// Аналогичная запись  
// a = a / 4
```

- %=

```
var a = 22  
a %= 4  
print(a) // 2  
// Аналогичная запись  
// a = a % 4
```

# Строковые типы данных

В данном блоке есть 2 типа данных: **Character** и **String**.

**Character** — символ, он может содержать только 1 символ.

**String** — строка, она может не содержать символов или содержать их множество.

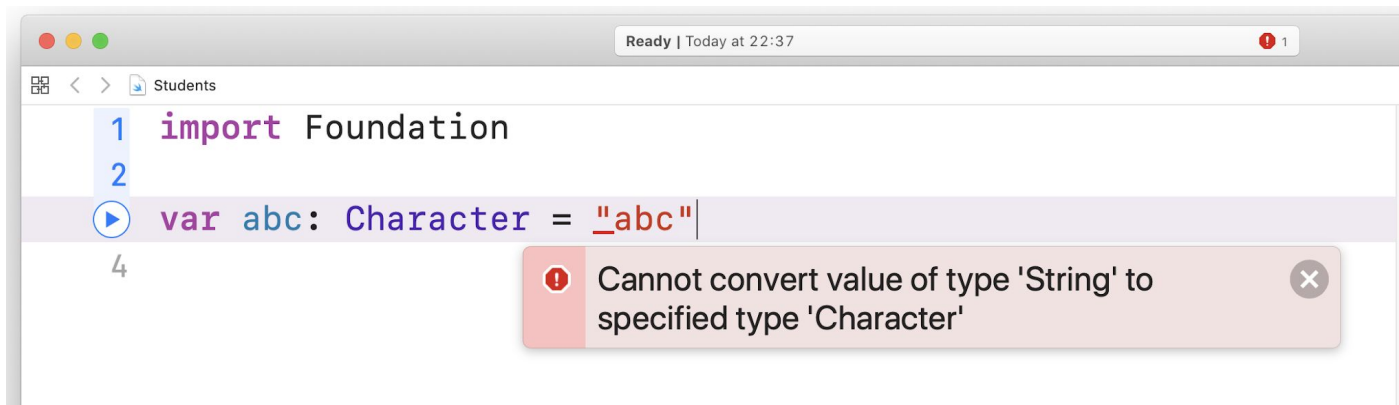
**String** — не просто набор **Character**, это отдельный тип данных.

```
var a: Character = "a"  
var hello: String = "hello"
```



# Строковые типы данных

Если в переменную с типом `Character` записать больше 1 символа, произойдет ошибка:



```
1 import Foundation
2
3 var abc: Character = "abc"
4
```

Cannot convert value of type 'String' to specified type 'Character'

# Преобразование String и Character

Получение String из Character:

```
var myChar: Character = "a"  
var myString = String(myChar)
```

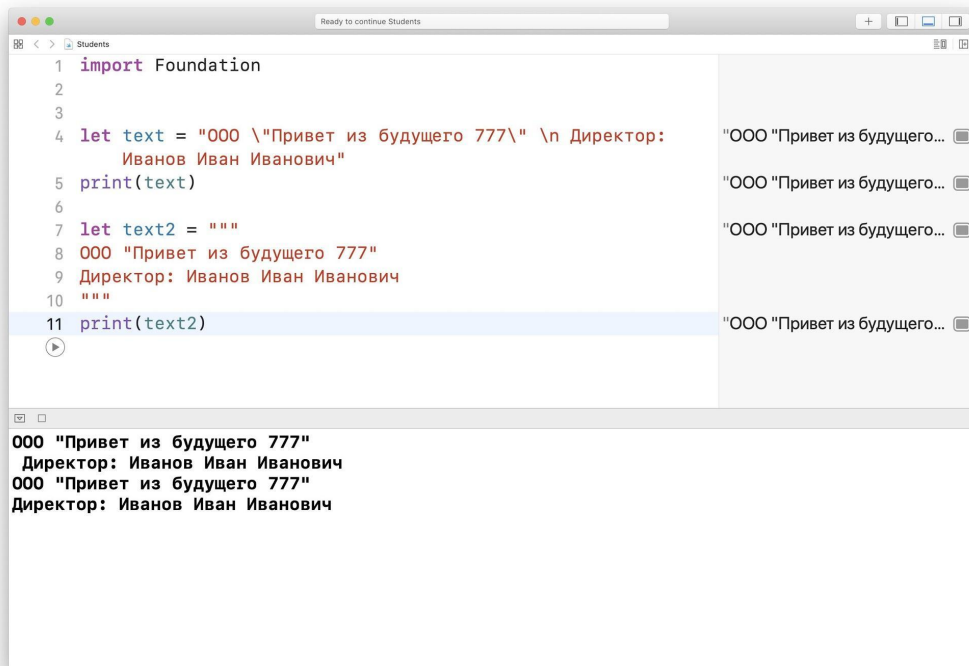
Получение Character из String:

```
let text = "test"  
let charIndex = text.index(text.startIndex, offsetBy: 1)  
let char = text[charIndex]  
print(char) // "e"
```

# Строковые типы данных

В строках могут присутствовать специальные символы. Давайте рассмотрим основные из них:

```
\n: перевод на новую строку  
\t: табуляция  
\: кавычка  
\\: обратный слеш
```



Начиная со Swift 4.0, многострочный текст обозначается `"""` и без использования переноса строки. Однако такая разметка текста часто используется.



## Итоги лекции:

- Создали первую программу на Swift;
- Узнали о структуре программы;
- Научились инициализировать переменные и константы;
- Получили знания об именовании переменных;
- Узнали, что такое числовые и строковые типы данных, и какие операции с ними можно совершить.



# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаём в чате Slack!
- Задачи можно сдавать по частям.
- Зачёт по домашней работе проставляется после того, как приняты **все задачи**.



**Задавайте вопросы и напишите отзыв о лекции!**

**НИКИТА КАЗАКОВ**

