



УНИВЕРСИТЕТ ИТМО

**Лекция 0**  
**Введение.**  
**Базовые концепции**

**Преподаватель:**

**Мусаев Андрей Александрович**  
**К.Т.Н.**  
**доцент**

**Обратная связь:**

- [amusayev1990@gmail.com](mailto:amusayev1990@gmail.com)
- [vk.com/triskaideka](https://vk.com/triskaideka)
- telegram



## **Предмет:**

***Алгоритмы и структуры данных на  python<sup>TM</sup>***

## **План работ на семестр:**

- **Лекции (очные)**
- **Лабораторные работы (домашнее задание) (~6 шт.)**
- **Защиты лабораторных работ (очные) (~6 шт.)**
- **Контрольные работы (2 шт.)**
- **Экзамен**



## **Лабораторные работы:**

**Задание рассылается группе после лекций.**

**Содержат задание по новому материалу, соответствующему теме лекций и могут содержать дополнительные задания для закрепления пройденного материала.**

**Выполняются дома.**

## **Требования к лабораторной работе:**

**Срок выполнения – 1,5 недели (до четверга следующей нечетной недели). Можно сдать отчет раньше, тогда будет время, чтобы переделать без снижения баллов. Присылать на почту.**

**Оформленный отчет: титульный лист, задание, решение (листинг), результат выполнения программы, выводы.**

**Оценивается в первую очередь оригинальность, читабельность, работоспособность кода и выводы.**

## **Защита лабораторной работы:**

**Защита происходит при  
условии принятого отчета.**

**Защита устная с вопросами по  
вашей лабораторной и по теме  
задания.**

Санкт-Петербургский Национальный Исследовательский Университет  
Информационных Технологий, Механики и Оптики

Кафедра Систем Управления и Информатики

Лабораторная работа №X

Вариант №Y

Выполнил(и):

Фамилия И.О.

Проверил

Мусаев А.А.

Санкт-Петербург,  
20XX



## **Контрольные работы:**

**Всего 9 вопросов по пройденному материалу за модуль.**

- ***8 вопросов – практические (что происходит при выполнении кода / найдите ошибку) и теоретические (дайте определение).***
- ***1 вопрос – практический (напишите программу / составьте блок-схему)***



## **Экзамен:**

***Всего 9 вопросов по пройденному материалу.***

- ***8 вопросов – практические (что происходит при выполнении кода / найдите ошибку) и теоретические (дайте определение).***
- ***1 вопрос – практические (напишите программу / составьте блок-схему)***



## **Экзамен:**

### ***Экзаменационный проект:***

***При получении 75 баллов (74.1+ округляется в пользу студента) появляется возможность написать экзаменационный проект – сложную программу на тематику, выбранную студентом, которая будет зачтена за сдачу экзамена.***

***Замена досрочного экзамена.***



**Лекции:**

	<b><i>ЧЕТНАЯ</i></b>	<b><i>НЕЧЕТНАЯ</i></b>
<b><i>8:20</i></b>	<b><i>K3121</i></b>	<b><i>K3120</i></b>
<b><i>11:40</i></b>	<b><i>K3123</i></b>	<b><i>K3122</i></b>



### **Список литературы:**

«Изучаем Python. Том 1»  
(Лутц М.)

«Структуры данных и алгоритмы»  
(А. В. Ахо, Д. Э. Хопкрофт, Д. Д. Ульман)


### **Среда разработки:**

PyCharm или на выбор студента

### **Операционная система:**

На выбор студента

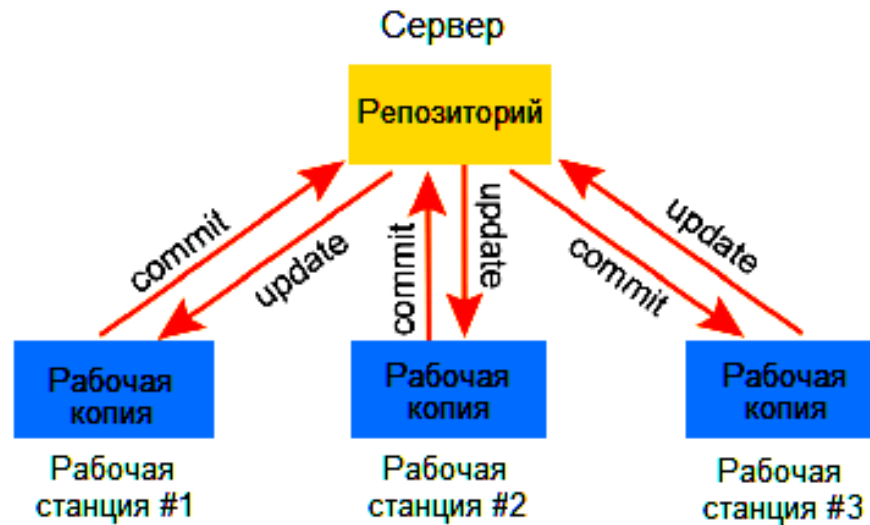
 **git** — распределённая система управления версиями.

 **git** — хранилище данных, содержащее различные версии созданного проекта.



- 1) Восстановление предыдущих решений
- 2) Сохранение полезных, но не актуальных решений
- 3) Обмен решениями при распределенной работе

## Централизованная система контроля версий



Локальная часть

## Установка Git

### Windows

- <https://git-scm.com/>

### Linux

- <https://git-scm.com>

### Terminal:

- apt-get install git
- другие установщики

### Mac

- <https://git-scm.com>

### Terminal:

- git

## Downloads



macOS



Windows



Linux/Unix



— крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

<https://github.com/>



Регистрация +  
Инструкции на сайте



`git add FILE/DIR`

`git commit -u "comment - будет виден на github"`

`git push` - отправит commit на github



## Права на файл

### Unix-системы:

u – владелец  
g – группа владельца  
o – остальные пользователи

r – чтение  
w – запись  
x – выполнение

двоичная	восьмеричная	символьная	права на файл	права на каталог
000	0	---	нет	нет
001	1	--x	выполнение	чтение свойств файлов
010	2	-w-	запись	нет
011	3	-wx	запись и выполнение	всё, кроме получения имени файлов
100	4	r--	чтение	чтение имён файлов
101	5	r-x	чтение и выполнение	доступ на чтение файлов/их свойств
110	6	rw-	чтение и запись	чтение имён файлов
111	7	rwx	все права	все права

### Git:

644 – нельзя выполнять  
755 – можно выполнять

### Windows:

Чтение = Выполнение



## Переменные.

### Числа

Числа в Python бывают трёх типов: целые, с плавающей точкой и комплексные.

- Примером целого числа может служить 2.
- Примерами чисел с плавающей точкой (или “плавающих” для краткости) могут быть 3.33 и 3.33E-4. Обозначение E показывает степени числа 10.
- Примеры комплексных чисел:  $(-5+4j)$  и  $(5.5-4.4j)$





## Строки

Строка – это *последовательность символов*. Чаще всего строки – это просто некоторые наборы слов. После создания строки её больше нельзя изменять. Слова могут быть как на английском языке, так и на любом другом, поддерживаемом стандартом Unicode, что означает почти на любом языке мира.

## Способы ввода строки:

**A='text'**

**A="text"**

**A=input()**

В первых двух случаях переменная A станет строкой, содержащей текст text, причем одинарные кавычки ничем не будут отличаться от двойных. Во втором случае текст надо будет вводить с клавиатуры, причем даже в случае ввода чисел, переменная по-прежнему будет являться строкой, а не числом



```
>>> input()
```

```
Hello World!
```

```
'Hello World!'
```

```
>>> input()
```

```
1234
```

```
'1234'
```

Чтобы строка '1234' стала числом, используйте команду `int`.



```
>>> A=int(input())
```

```
>>> A
```

```
1234
```

## Операторы

Большинство предложений (логических строк) в программах содержат *выражения*. Простой пример выражения: **2+3**. Выражение можно разделить на операторы и операнды.

*Операторы* – это некий функционал, производящий какие-либо действия, который может быть представлен в виде символов, как например +, или специальных зарезервированных слов. Операторы могут производить некоторые действия над данными, и эти данные называются *операндами*. В нашем случае 2 и 3 – это операнды.



## Операторы и их применение

Оператор	Название	Объяснение	Примеры
+	Сложение	Суммирует два объекта	3 + 5 даст 8; 'a' + 'b' даст 'ab'
-	Вычитание	Даёт разность двух чисел; если первый операнд отсутствует, он считается равным нулю	-5.2 даст отрицательное число, а 50 - 24 даст 26.
*	Умножение	Даёт произведение двух чисел или возвращает строку, повторённую заданное число раз.	2 * 3 даст 6. 'la' * 3 даст 'lalala'.
**	Возведение в степень	Возвращает число x, возведённое в степень y	3 ** 4 даст 81 (т.е. 3 * 3 * 3 * 3)
/	Деление	Возвращает частное от деления x на y	4 / 3 даст 1.3333333333333333.
//	Целочисленное деление	Возвращает неполное частное от деления	4 // 3 даст 1.
%	Деление по модулю	Возвращает остаток от деления	8 % 3 даст 2. -25.5 % 2.25 даст 1.5.
<<	Сдвиг влево	Сдвигает биты числа влево на заданное количество позиций. (Любое число в памяти компьютера представлено в виде битов - или двоичных чисел, т.е. 0 и 1)	2 << 2 даст 8. В двоичном виде 2 представляет собой 10. Сдвиг влево на 2 бита даёт 1000, что в десятичном виде означает 8.



>>	Сдвиг вправо	Сдвигает биты числа вправо на заданное число позиций.	11 >> 1 даст 5. В двоичном виде 11 представляется как 1011, что будучи смещённым на 1 бит вправо, даёт 101, а это, в свою очередь, ни что иное как десятичное 5
&	Побитовое И	Побитовая операция И над числами	5 & 3 даёт 1.
	Побитовое ИЛИ	Побитовая операция ИЛИ над числами	5   3 даёт 7
^	Побитовое ИСКЛЮЧИТЕЛЬНО ИЛИ	Побитовая операция ИСКЛЮЧИТЕЛЬНО ИЛИ	5 ^ 3 даёт 6
~	Побитовое НЕ	Побитовая операция НЕ для числа хсоответствует $-(x+1)$	~5 даёт -6.
<	Меньше	Определяет, верно ли, что x меньше y. Все операторы сравнения возвращают True или False. Обратите внимание на заглавные буквы в этих словах.	5 < 3 даст False, а 3 < 5 даст True.  Можно составлять произвольные цепочки сравнений: 3 < 5 < 7 даёт True.
>	Больше	Определяет, верно ли, что x больше y	5 > 3 даёт True. Если оба операнда - числа, то перед сравнением они оба преобразуются к одинаковому типу. В противном случае всегда возвращается False.
<=	Меньше или равно	Определяет, верно ли, что x меньше или равно y	x = 3; y = 6; x <= y даёт True.
>=	Больше или равно	Определяет, верно ли, что x больше или равно y	x = 4; y = 3; x >= 3 даёт True.
==	Равно	Проверяет, одинаковы ли объекты	x = 2; y = 2; x == y даёт True. x = 'str'; y = 'stR'; x == y даёт False. x = 'str'; y = 'str'; x == y даёт True.
!=	Не равно	Проверяет, верно ли, что объекты не равны	x = 2; y = 3; x != y даёт True.

# If-elif-else

**Условная инструкция if-elif-else** (её ещё иногда называют оператором ветвления) - основной инструмент выбора в Python.

**if test1:**

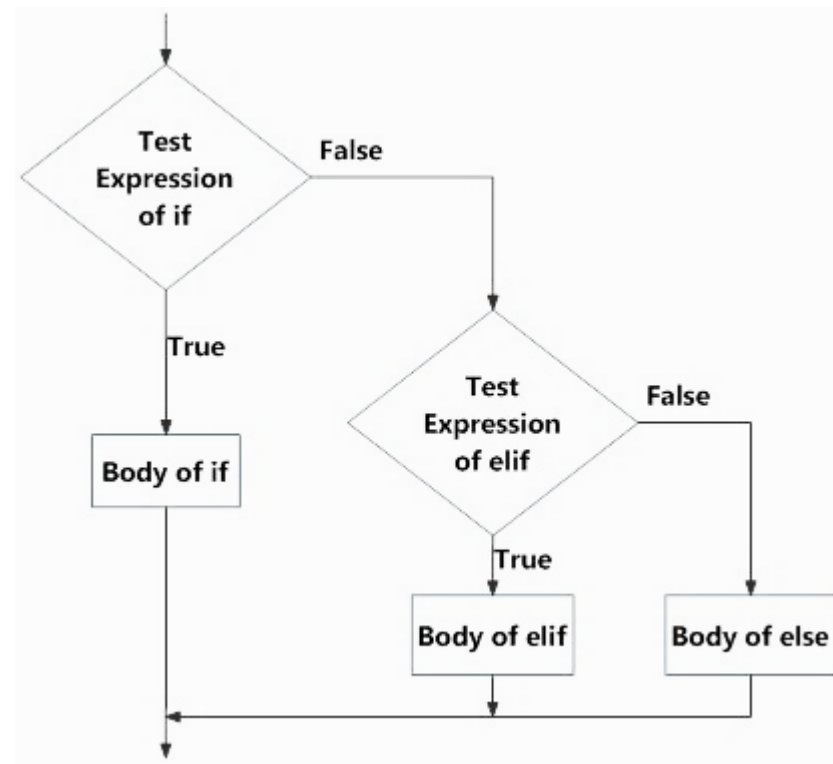
**state1**

**elif test2:**

**state2**

**else:**

**state3**





## If-elif-else



```
a = int(input())
```

```
if a < 18:
```

```
    print('Kid')
```

```
elif 18 <= a <= 21:
```

```
    print('Youth')
```

```
else:
```

```
    print('Adult')
```



# If-elif-else

## Проверка истинности в Python

Любое число, не равное 0, или непустой объект - истина.

Числа, равные 0, пустые объекты и значение None – ложь

Операции сравнения применяются к структурам данных рекурсивно

Операции сравнения возвращают True или False

Логические операторы and и or возвращают истинный или ложный объект-операнд



## If-elif-else

### Логические операторы:

#### **X and Y**

Истина, если оба значения X и Y истинны.

#### **X or Y**

Истина, если хотя бы одно из значений X или Y истинно.

#### **not X**

Истина, если X ложно.



## If-elif-else

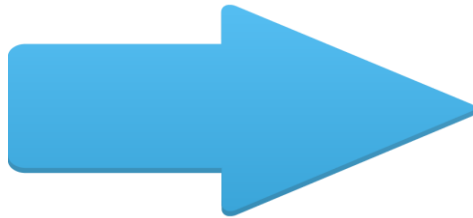
Трехместное выражение if/else

if X:

A = Y

else:

A = Z



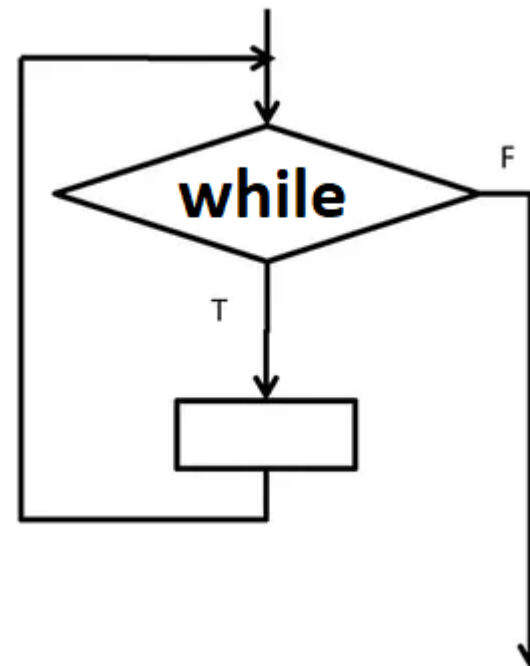
A = Y if X else Z

# While

**While** - один из самых универсальных циклов в Python, поэтому довольно медленный. Выполняет тело цикла до тех пор, пока условие цикла истинно.

**while test:**

**state(s)**



## For

Для повторения цикла некоторое заданное число раз  $n$  можно использовать цикл `for` вместе с функцией `range`:

```
for i in range(0, 4):           # равносильно инструкции for i in 0, 1, 2, 3
    print(i)
    print(i ** 2)
    print('Конец цикла')
```

Вызов `range(a, b)` означает, что индексная переменная будет принимать значения от  $a$  до  $b - 1$ .

# Break

Оператор `break` досрочно прерывает цикл.

```
for i in 'hello world':  
    if i == 'o':  
        break  
    print(i * 2, end="")
```

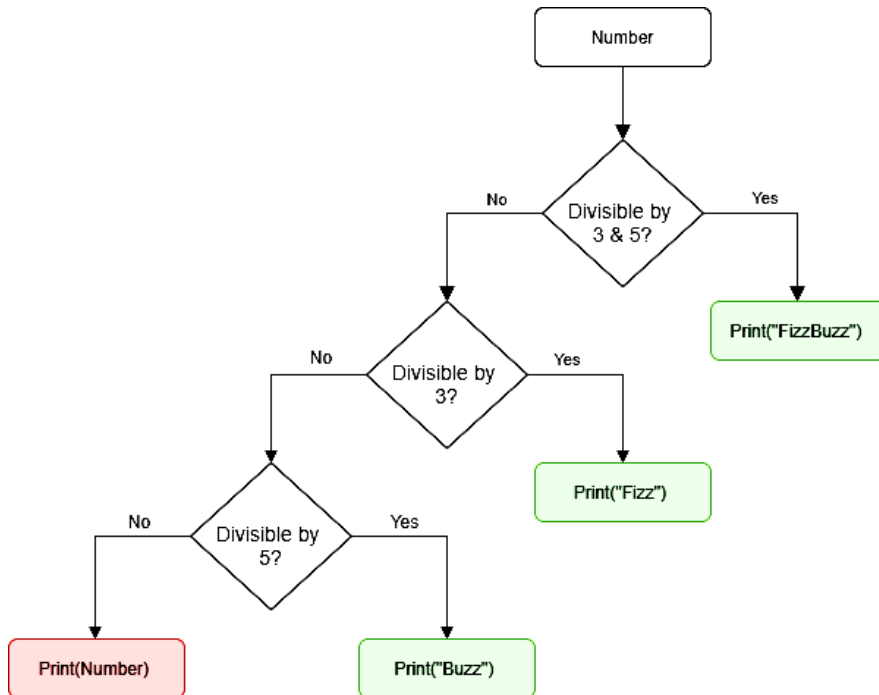


hheeIIII





# Задача!



**Fizz**  
**Buzz**

1	2	Fizz	4	Buzz	Fizz	7	8	Fizz	Buzz	11	Fizz	13	14	Fizz	Buzz
---	---	------	---	------	------	---	---	------	------	----	------	----	----	------	------