Лекция 1
Управляющие конструкции

Кисляков Иван

Кто я такой?

Иван Кисляков

лекции

Семинаристы

Канал и чат курса

GIT курса

Видеозаписи лекций

Как устроен курс

Лекция

Семинар

Домашнее задание

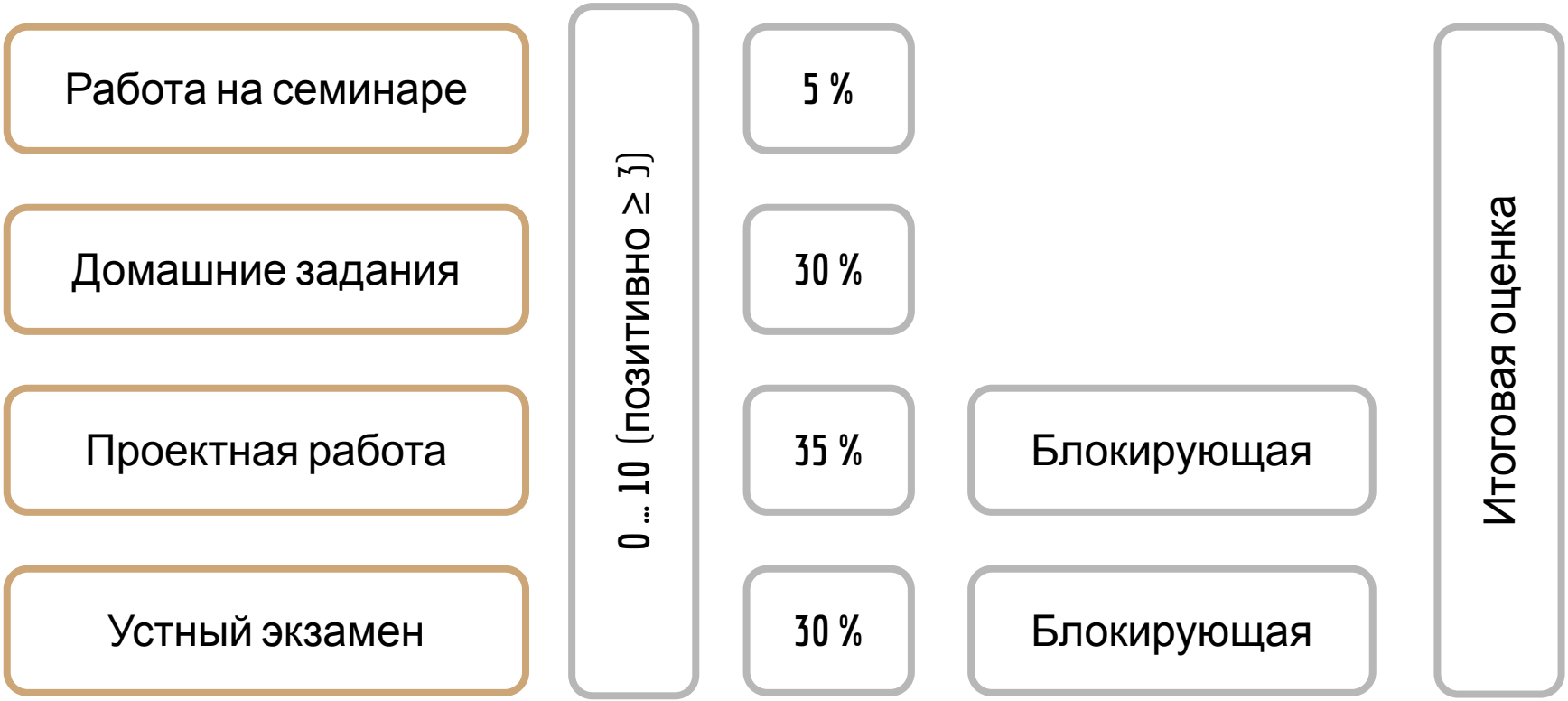
Итоговый проект

Домашнее задание

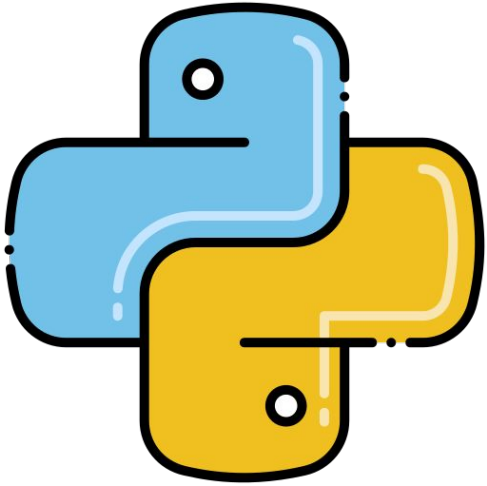
1. Еженедельное (на неделю)
2. Не успел - штраф 50%
3. Яндекс контест / jupyter notebook

4. Списал - получил проблемы
5. GPT - получил проблемы :(

Оценивание на курсе



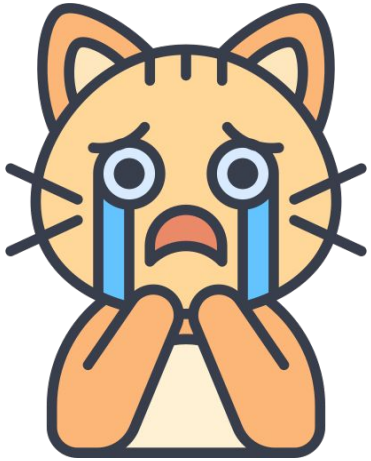
Что мы изучаем?



Python

- Скорость написания кода
- Скорость чтения кода
- Множество библиотек
- Легкость изучения

А минусы будут?



Python

Медленная скорость работы

Очень медленная...

+/- интерпретируемый

Заповеди Python

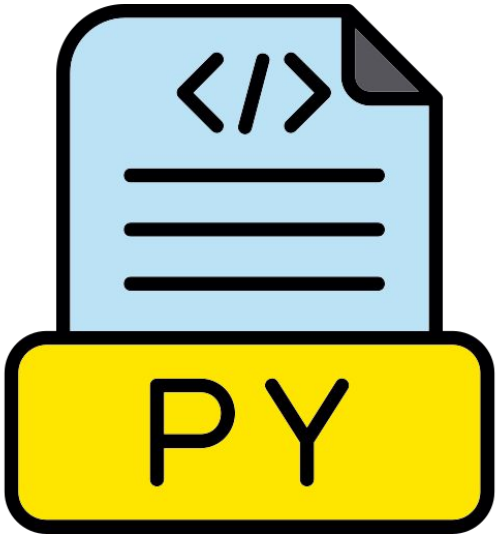
Заповеди

```
ivankisliakov — Python — 79×28

[ivankisliakov@MacBook-Air-103 ~ % python3                                    ]
Python 3.13.0 (v3.13.0:60403a5409f, Oct  7 2024, 00:37:40) [Clang 15.0.0 (clang
-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import this                                                              ]
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

PEP8



Стандарт написания красивого кода

Регламентирует:

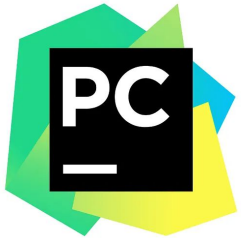
- отступы
- табуляцию
- пробелы
- нейминг

Среды разработки

PyCharm Community

Google Collab

Jupyter notebook



PyCharm

jupyter

Вывод данных

```
1  print("Hello, MIPT!")
2  print("Мы прогаем на Python")
3  print("А", "ты", 'на', 2, "курсе?")
```

```
Hello, MIPT!
Мы прогаем на Python
А ты на 2 курсе?
```

Комментарий

Комментарий - место в программе, которая видна разработчику, но игнорируется интерпретатором. (используется #)

```
print("Hello")  # программа это игнорирует
```

Переменные

```
1    a = 10
2    b = 15.4
3    c = a + b
4    d = "Переменные:"
5    print(d, a, b, c)
```

```
Переменные: 10 15.4 25.4
```

Типы данных

- Числовые
  - Целые (integer) — int
  - Дробные (floating) — float
- Строковый (string) — str
- Логический (boolean) — bool
- Нечто иное

Арифметика типов данных

Числовые

`+` `-`

`*` `/`

`**` `//` `%`

Строковый

`+` `'a' + 'b' == 'ab'`

`*` `'a' * 2 == 'aa'`

Преобразование типов

```
1  a = "12"
2  b = "17.5"
3  c = 186
4  d = "Строка "
5  e = int(a) + float(b)
6  f = d + str(c)
7  print(e)
8  print(f)
```

```
29.5
Строка 186
```

f - строка

```
1  a = 10
2  print("Переменная 'a' имеет значение", a)
3  b = 5.5
4  print(f"Переменная 'b' имеет значение {b}")
```

```
Переменная 'a' имеет значение 10
Переменная 'b' имеет значение 5.5
```

Продвинутый вывод данных

```
1  a = 10
2  b = 15
3  c = 20
4  print(a, b, c)
5  print(a, b, c, sep="-^-")
6  print(a, b, c, end="$$$")
7  print(" - это наши числа")
```

```
10 15 20
10-^-15-^-20
10 15 20$$$ - это наши числа
```

Ввод данных

```
1    a = input()
2    b = int(input())
3    print(a * 2)
4    print(b * 2)
```

```
hello
14
hellohello
28
```

Сравнение

Числовые

`>` `>=`

`<` `<=`

`==` `!=`

Строковый

`==` `!=` `<`

‘a’ < ‘b’

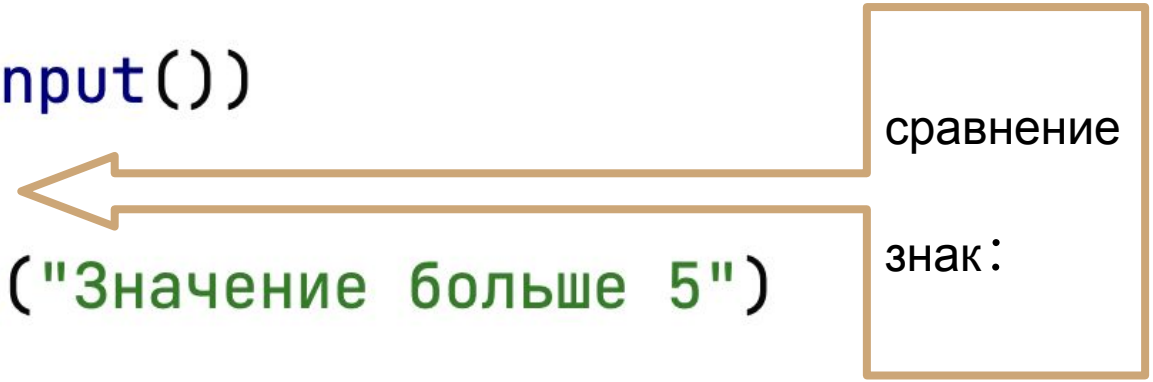
‘aoa’ < ‘aob’

in

‘a’ in ‘bac’

Условный оператор if

```
1    a = 5
2    b = int(input())
3    if b > a:
4        print("Значение больше 5")
```



Условный оператор else

```
1  a = 5
2  b = int(input())
3  if b > a:
4      print("Значение больше 5")
5  else:
6      print("Значение не больше 5")
```

4 пробела

Логические связки

```
if b > 5 or b < 0:
    print("(-inf, 0) U (5; +inf)")
if b > 0 and b < 5:
    print("(0; 5)")
if not (b > 5):
    print("(-inf; 5]")
```

or

and

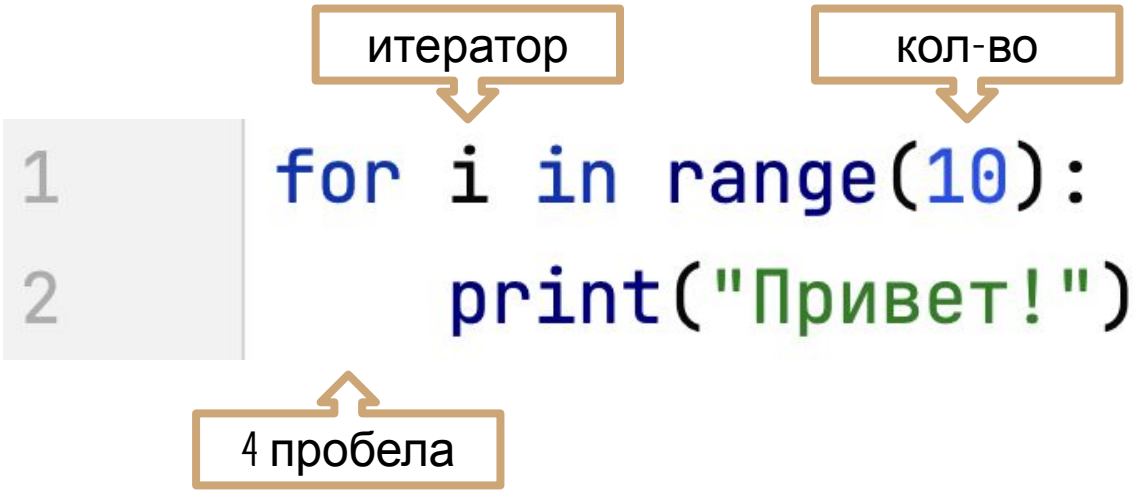
not

Сложное условие elif

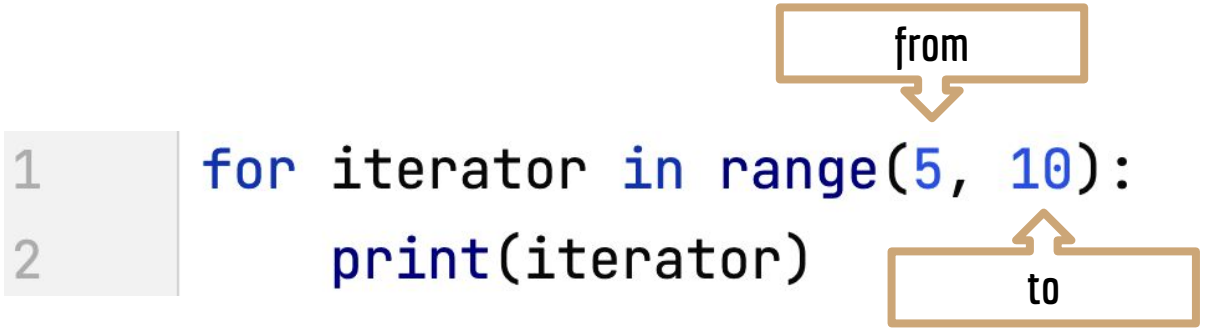
```
height = float(input())
if height < 120:
    print("Ребенок")
elif height < 150:
    print("Подросток")
elif height < 170:
    print("Юнец")
else:
    print("Взрослый")
```

115	Р
140	П
150	Ю
190	В

Цикл for



Диапазон значений




```
5
6
7
8
9
```

Менять значение итератора самостоятельно
не стоит

Вот так вот не надо

```
1  for iterator in range(5, 10):
2      print(iterator)
3      iterator = 1
```

```
5
6
7
8
9
```

Диапазон значений


```
for iterator in range(5, 13, 2):
    print(iterator, end=" ")
```

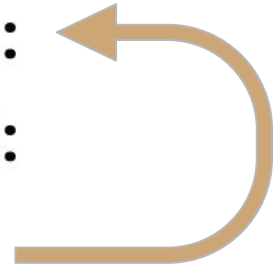
```
5 7 9 11
```

```
for iterator in range(10, 5, -1):
    print(iterator, end=" ")
```

```
10 9 8 7 6
```

Continue

```
1    for i in range(7):
2        if i % 2 == 0:
3            continue
4        print(f"Привет в {i} раз!")
```



```
Привет в 1 раз!
Привет в 3 раз!
Привет в 5 раз!
```

Break

```
1    for i in range(7):
2        if i == 3:
3            break
4        print(f"Привет в {i} раз!")
```

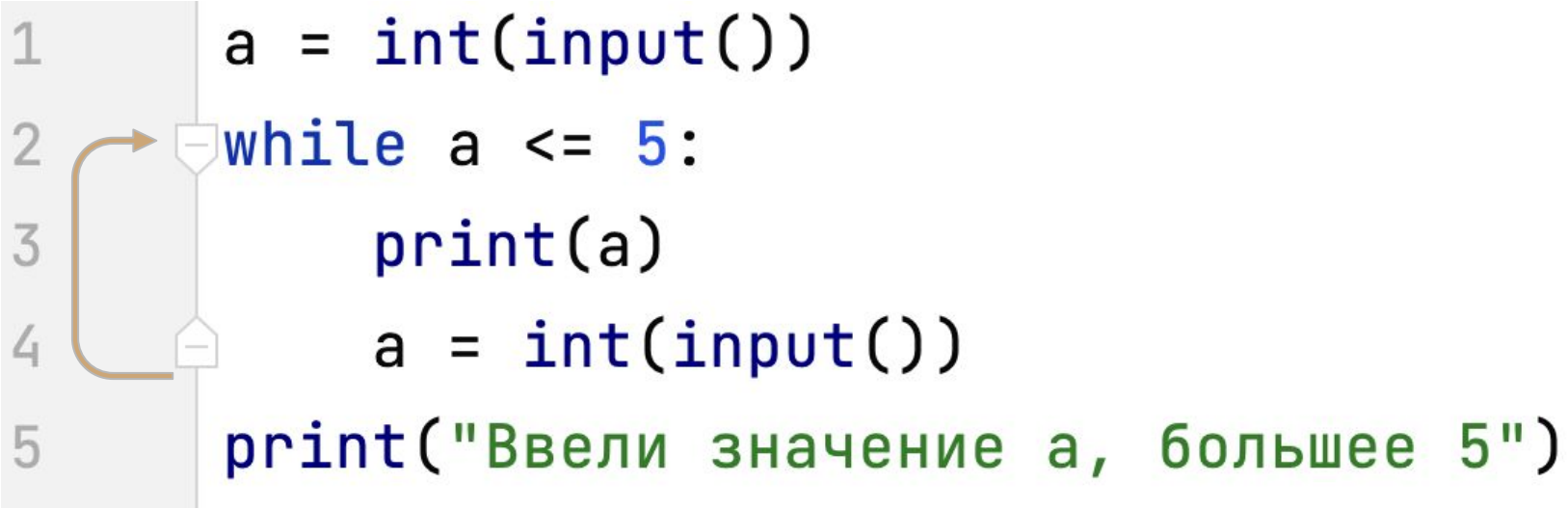
```
Привет в 0 раз!
Привет в 1 раз!
Привет в 2 раз!
```

Немного про красоту

```
for _ in range(5):
    print("Привет!")
```

Если итератор
не используется

Цикл while


1 a = int(input())
2 while a <= 5:
3 print(a)
4 a = int(input())
5 print("Ввели значение а, большее 5")

Цикл while True