

прога: 12 сентября

Задача на сегодня: вспомнить всё



План

огромен

1. написание кода: что такое программа и как она запускается
2. что есть в коде
 - a. переменные
 - b. ввод-вывод
 - c. логика:
 - i. приколюхи, специфичные для разных типов данных
 - ii. ветвления
 - iii. циклы
3. разбор теста

написание кода на
python

что такое программа и как её запустить

это текстовый файл в котором написан код
(~ список дел для компьютера)

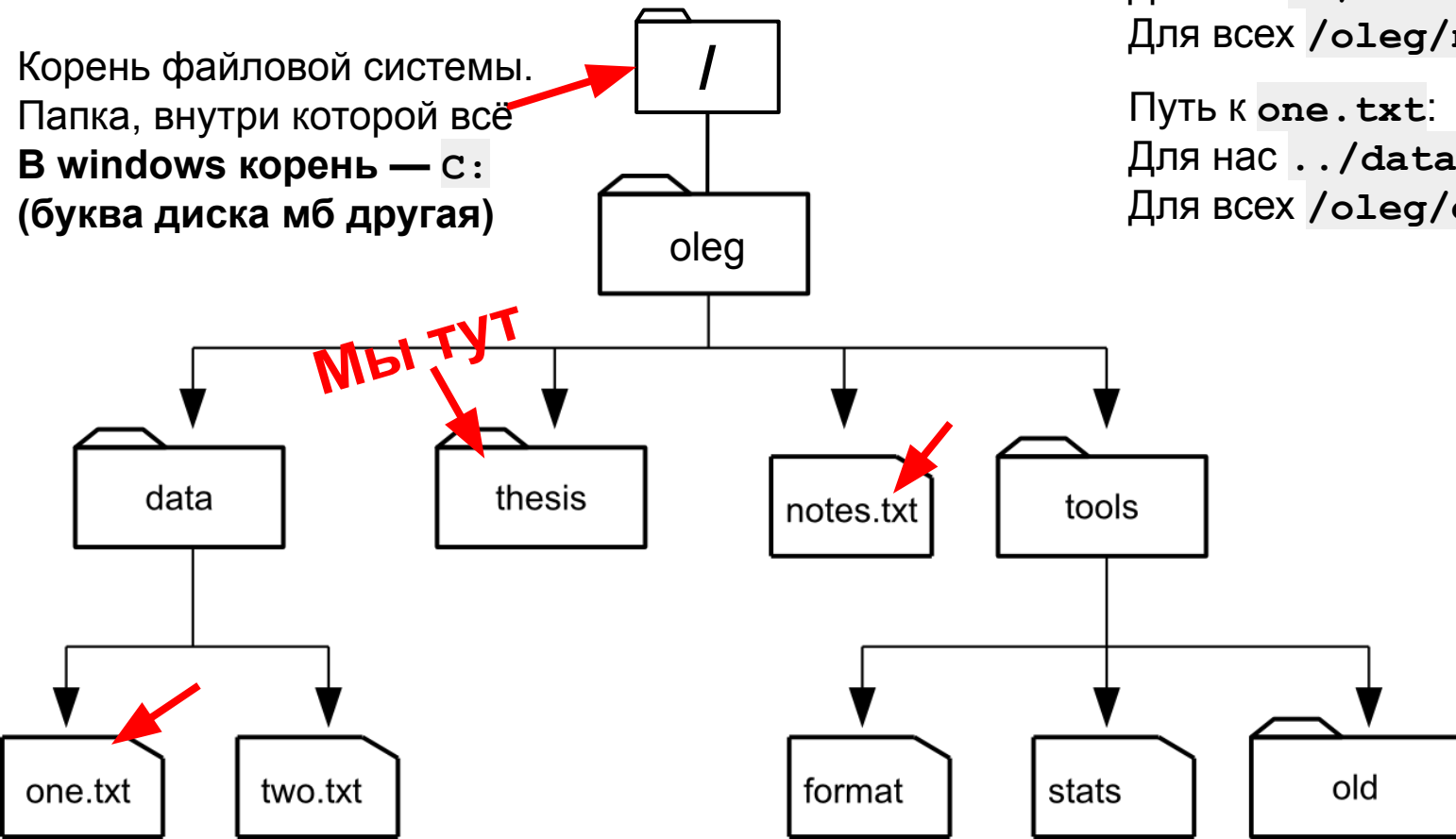
есть специальная программа "питон"
которая умеет заставлять компьютер
выполнять этот код

про файлы на компьютере

- их много, они сложены в папки
- некоторые файлы и папки необходимы чтобы комп нормально работал
- а некоторые — ваши
- файлы бывают текстовые и бинарные

Дерево файловой системы

Корень файловой системы.
Папка, внутри которой всё
В windows корень — C:
(буква диска мб другая)



Мы находимся в папке **thesis**

Путь к **notes.txt**:

Для нас **../notes.txt**

Для всех **/oleg/notes.txt**

Путь к **one.txt**:

Для нас **../data/one.txt**

Для всех **/oleg/data/notes.txt**

программа — текстовый или бинарный
файл?

что обычно есть в
программе

что обычно есть в программе

1. переменные

2. логика

а. какие-то операции

б. какое-то ветвление

с. какие-то циклы

что обычно есть в программе (в коде?)

переменные

логика

a. какие-то операции

например, сложение или сравнение
или функция "написать"

b. какое-то ветвление

ключевые слова: if .. elif .. else
можно и без elif и элс

c. какие-то циклы

ключевые слова: while или for .. in ..

```
a = 0
cnt = 2
condition_satisfied = True
while condition_satisfied:
    a = a+1
    print(a)
    cond = a < cnt
    if cond:
        condition_satisfied = True
    else:
        condition_satisfied = False
```

синтаксический сахар

- чтобы простые вещи записать хитрее но короче
- недостаток сахара: многие элементы кода как-то перемешаются друг с другом
- но получится прикольно! давайте менять наш код

```
a = 0
cnt = 2
c_satisfied = True
while c_satisfied:
    a = a+1
    print(a)
    cond = a < cnt
    if cond:
        c_satisfied = True
    else:
        c_satisfied = False
```

код пишется по правилам

например, в питоне очень любят двоеточия и отступы

важно! договор питонистов:

если вы отдаёте код кому-то, то отступ — это всегда **ЧЕТЫРЕ ПРОБЕЛА**

```
if today == 'Saturday':  
    print('Party!')  
elif today == 'Sunday':  
    if condition == 'Headache':  
        print('Recover, then rest.')  
    else:  
        print('Rest.')  
else:  
    print('Work, work, work.')
```

This single line of code is a suite.

These single lines of code are both suites.

These four lines of code are a suite

This single line of code is a suite.

Indentation level zero

Indentation level one

Indentation level two

программа которую мы обсуждали **странная!** почему?

```
a = 0
cnt = 2
c_satisfied = True
while c_satisfied:
    a = a+1
    print(a)
    cond = a < cnt
    if cond:
        c_satisfied = True
    else:
        c_satisfied = False
```



что обычно есть в программе

1. переменные
2. логика
 - а. какие-то операции
 - б. какое-то ветвление
 - с. какие-то циклы
3. интрига :) взаимодействие с миром
 - а. либо программа обрабатывает файлы, которых мы раньше не видели
 - б. либо работает с пользователем, и он непредсказуем

подробнее о всём
этом

что обычно есть в программе

1. переменные
2. логика
3. взаимодействие с миром

переменные

имеют имена

желательно понятные и ничего не
ломающие

примеры работы с переменными

переменные, бывает, возникают между делом
это было в тесте!

```
for bykovka in input():  
    print(bykovka)
```

логика: операции и функции

- в результате операции (и функции!) всегда получается то, что можно запомнить в переменную
- иногда в операции участвуют другие переменные

назовите операцию или функцию

логика:
условия



УСЛОВИЯ

Ветвление алгоритма в зависимости от условия, позволяющий получать разные расчеты исходя из условий

```
if True and count == 1 and count == 2:  
    print("if")  
elif count == 'count':  
    print("First elif")  
else:  
    print("Else")
```

- В базовом смысле ветвлению необходимо только наличие `if`, в отличие от некоторых языков, не нужно даже использовать `else`.
- `Elif` - дополнительная ветка внутри `else`, позволяет создать больше чем 1 выбор
- (фактически при создании ветвления `if - elif - else`, мы формируем дерево решений. Можно сказать, что мы с вами пишем Machine learning :))

Условия

```
if today == "Saturday":
```

```
    print("Partyyy 🎉🎊🎂")
```

```
if today == "Saturday":
```

```
    print("Partyyy 🎉🎊🎂")
```

```
else:
```

```
    print("Do some work 🧑💻")
```

```
if today == "Saturday":
```

```
    print("Partyyy 🎉🎊🎂")
```

```
elif today == "Sunday":
```

```
    print("Rest 😴")
```

```
elif today == "Tuesday":
```

```
    print("Workout day 🚴🏃")
```

```
else:
```

```
    print("Do some work 🧑💻")
```

Тернарные операторы

Конструкция позволяет в одну строку приравнять переменную в зависимости от значения другой переменной

Из этого

```
if X:
```

```
    A = Y
```

```
else:
```

```
    A = Z
```

Можно сделать это - `A = Y if X else Z`

Волшебный pass

```
if today == "Saturday":  
    print("Partyyy 🎉")  
elif today == "Sunday":  
    pass  
elif today == "Tuesday":  
    print("Workout day 🚴")  
else:  
    print("Do some work 🧑💻")
```



логика:
циклы



while

```
>>> i = 9
>>> while i >= 5:
...     print(i)
...     i = i-1
...
9
8
7
6
5
>>> print('Finally, i =', i)
Finally, i = 4
```

Похож на if синтаксисом:

```
ключевое_слово условие:
    тело
```

Раз за разом будет выполнять код из **тела** цикла пока истинно **условие** или пока мы принудительно этот процесс не остановим

break

Прерывает выполнение цикла

```
>>> cnt = 1
>>> while cnt < 100:
...     print(cnt)
...     cnt = cnt + 1
...
```

Напечатает числа от 1 до 99

```
>>> cnt = 1
>>> while cnt < 100:
...     print(cnt)
...     cnt = cnt + 1
...     if cnt % 7 == 0:
...         break
...
```

Вот он!



Напечатает числа от 1 до угадajte

break

Прерывает выполнение цикла

```
>>> cnt = 1
>>> while cnt < 100:
...     print(cnt)
...     cnt = cnt + 1
...
```

Напечатает числа от 1 до 99

```
>>> cnt = 1
>>> while cnt < 100:
...     print(cnt)
...     cnt = cnt + 1
...     if cnt % 7 == 0:
...         break
...
```

Вот он!



Напечатает числа от 1 до 6

continue

Старается сразу запустить следующую итерацию

```
>>> cnt = 1
>>> while cnt < 10:
...     print(cnt)
...     if cnt % 5 == 0:
...         cnt = cnt - 4
...     cnt = cnt + 1
...
```

Напечатает последовательность:

1, 2, 3, 4, 5, 2, 3, 4, 5, 2, 3, 4, 5 и т.д.

```
>>> cnt = 1
>>> while cnt < 10:
...     print(cnt)
...     if cnt % 5 == 0:
...         cnt = cnt - 4
...         continue
...     cnt = cnt + 1
...
```

Напечатает угадajte (проверьте)
что

continue

Старается сразу запустить следующую итерацию

```
>>> cnt = 1
>>> while cnt < 10:
...     print(cnt)
...     if cnt % 5 == 0:
...         cnt = cnt - 4
...     cnt = cnt + 1
...
```

Напечатает последовательность:

1, 2, 3, 4, 5, 2, 3, 4, 5, 2, 3, 4, 5 и т.д.

```
>>> cnt = 1
>>> while cnt < 10:
...     print(cnt)
...     if cnt % 5 == 0:
...         cnt = cnt - 4
...         continue
...     cnt = cnt + 1
...
```

Напечатает угадajte (проверьте)
что

For штука in штуки

```
>>> for letter in 'hello':  
...     print(letter)
```

Тут *штука* — letter, *штуки* — “hello”.
letter — это тоже переменная, вот так
причудливо заведённая.

Позволяет поэлементно перебирать
штуки. Например, побуквенно —
строки.

range()

Генерирует последовательность чисел. Можно задавать нижнюю и верхнюю границы и длину шага. Удобен в связке с for.

```
>>> for i in range(7):  
...     print(i)  
...  
0  
1  
2  
3  
4  
5  
6
```

```
>>> for i in range(2, 7):  
...     print(i)  
...  
2  
3  
4  
5  
6
```

```
>>> for i in range(2, 7, 2):  
...     print(i)  
...  
2  
4  
6
```

взаимодействие с внешним миром: ввод-вывод

Давайте напишем программу "Привет, мир!"

Давайте напишем программу "Привет, \$USERNAME"

взаимодействие с внешним миром: ввод-вывод


файлы

Как получить содержимое файла в питоне

Прочитать файл :)

Путь к `notes.txt` для нас `../notes.txt`

```
my_cool_f = open('../notes.txt', encoding='utf-8')  
for my_cool_line in my_cool_f:  
    print(my_cool_line)  
my_cool_f.close()
```



Лучше всегда
писать про
кодировку, но на
слайдах дальше не
буду тк места мало
и пара про другое

[illegible]

```
open(имя_файла, 'r', encoding="utf-8")
```

чтобы не сломалась кириллица, лучше явно просить
питон работать в utf-8

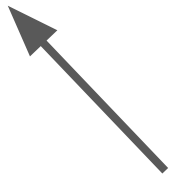
`open(filename, encoding="utf-8")` — это сокращение для

```
open(filename, 'r', encoding="utf-8")
```

```
open(имя_файла, 'r', encoding="utf-8")
```

`open(filename, encoding="utf-8")` — это сокращение для

```
open(filename, 'r', encoding="utf-8")
```



Можно ещё 'w'
и 'a'

`open(fn, 'w', ...`

'w' for Подготовиться писать в файл с чистого листа

```
>>> f = open("f1.txt", 'w', encoding="utf-8")
```

```
>>> print("hi", file=f, end='\n')
```

```
>>> f.close()    # закрывать файлы — ВАЖНАЯ традиция
```

```
>>> exit()
```

```
oleg@DESKTOP-8LKO59R:~$ cat f1.txt
```

```
hi
```

```
oleg@DESKTOP-8LKO59R:~$
```

`open(fn, 'w', ...`

Подготовиться писать в файл **с чистого листа**

Если в файле что-то было, всё сотрётся тревога
тревога

Читать буквой 'r', писать — 'w'

w всё сотрёт перед тем, как что-то записать в файл

open(fn, 'a', ...

'a' for Подготовиться дописывать в файл

```
>>> f = open("f1.txt", 'a', encoding="utf-8")
```

```
>>> print("hi", file=f, end='\n')
```

```
>>> f.close()
```

```
>>> exit()
```

```
oleg@DESKTOP-8LKO59R:~$ cat f1.txt
```

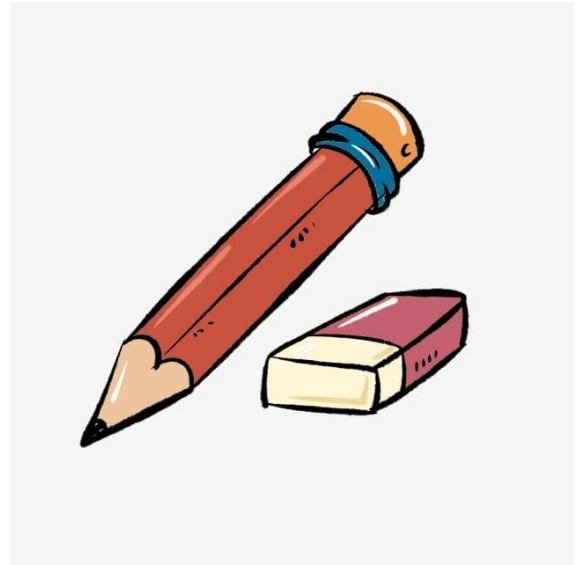
```
hi
```

```
hi
```

```
oleg@DESKTOP-8LKO59R:~$
```

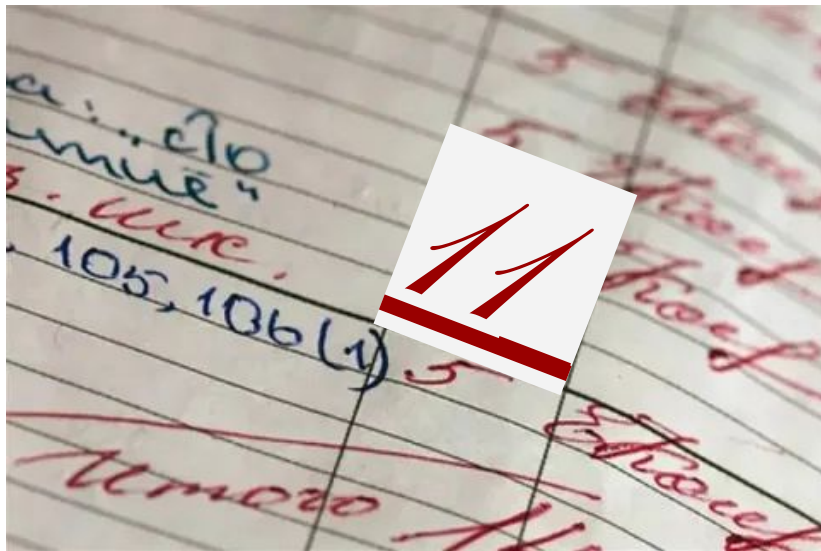
w — писать в файл вместо всего, что там было

a — писать в файл, продолжая.



строки

что такое типы данных



прикольные функции строк и их применения

int

lower upper

contains

startswith endswith

форматстроки

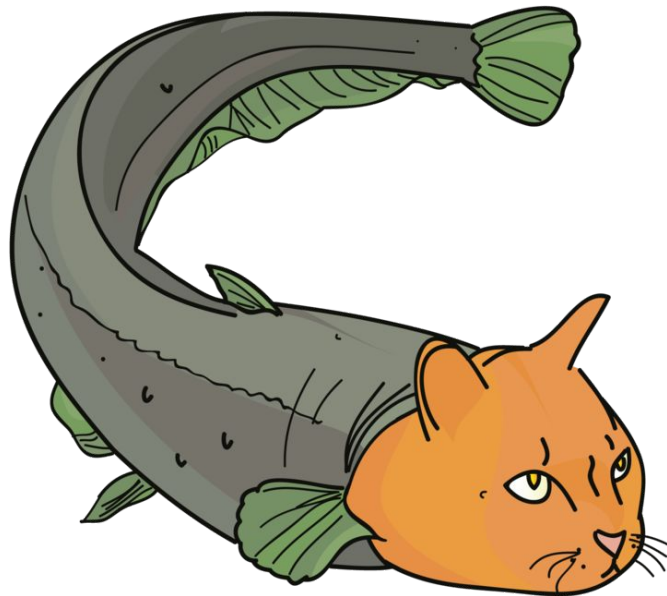
Некоторые операции со строками

- Взятие i -того элемента строки `"catfish"[1]` отдаст `"a"`
- Проверка, есть ли одна строка внутри другой — `in`

```
>>> "cat" in "catfish"  
True
```

- Узнать длину строки — `len()`

```
>>> len("catfish")  
7
```

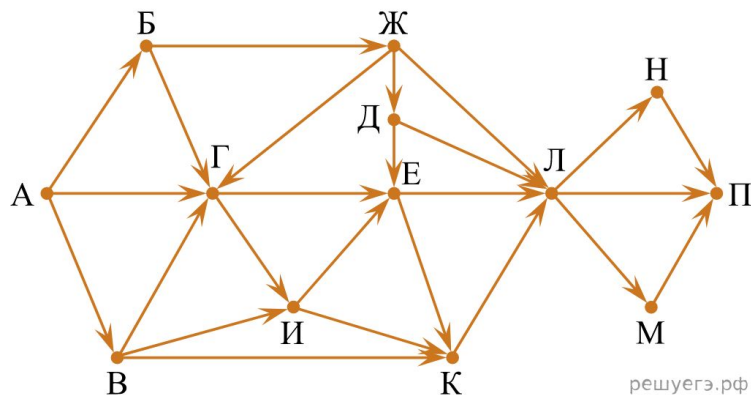


TECT!

1. 1.1 Лингвиста пригласили на национальный праздник индейцев ейцев. Чтобы не обидеть ейцев, ему нужно надеться в национальный костюм. В костюме много деталей, и важно не перепутать что на что надевается.

Из своих записок лингвист знает лишь то, какую вещь надевают перед какой. Помогите ему понять, в каком порядке теперь надевать вещи?

1.2 На рисунке — схема дорог, связывающих пункты А, Б, В, Г, Д, Е, Ж, И, К, Л, М, Н, П.



Сколько существует разл

сходящихся через пункт Е?

Расскажите, как вообще решать такие задачи? представьте, что вы меня учите.

2. 2.1| Что напечатается в результате работы кода

```
a = 8
```

```
b = print((int(a/input("введите число: "))) * "картошка"))
```

2.2| Что напечатается в результате работы кода

```
for height in range(0,5):
```

```
    width = height*4+1
```

```
    print('#'*width)
```

2.3| Что иллюстрирует код

```
from random import *
```

```
user_pass = input("Введите ваш пинкод: ")
```

```
password = [1,2,3,4,5,6,7,8,9,0]
```

```
guess = ""
```

```
while (guess != user_pass):
```

```
    guess = ""
```

```
    for letter in range(len(user_pass)):
```

```
        guess_letter = randint(0, 10)
```

```
        guess = str(guess_letter) + str(guess)
```

```
        print(guess)
```

```
print("Your password is",guess)
```

3. Подзадания под звёздочкой решайте после первого.

Ученики 8П класса школы №57 думают что никто не любит числа больше 100. Чтобы проверить свою гипотезу, они проводят опрос и спрашивают любимые числа своих одношкольников (кто-то даже отрицательные любит!).

Проверять результаты опроса вручную всем лень, поэтому ребятам хочется написать программу, которая делала бы все за них. Она должна уметь выводить:

1. **YES** если хоть кто-то любит числа больше 100 и **NO** в остальных случаях
- 2* среднее арифметическое всех любимых чисел;
- 3* самое большое любимое число

На вход сначала подается количество опрошенных, а потом любимое число каждого с новой строки. Программа должна вывести сначала **YES/NO**, потом среднее, а в конце максимум.

Пример ввода-вывода программы:

Ввод	Вывод
5	NO
-3	14
0	56
56	
10	
7	
4	YES
111	28.25
-19	111
13	
8	