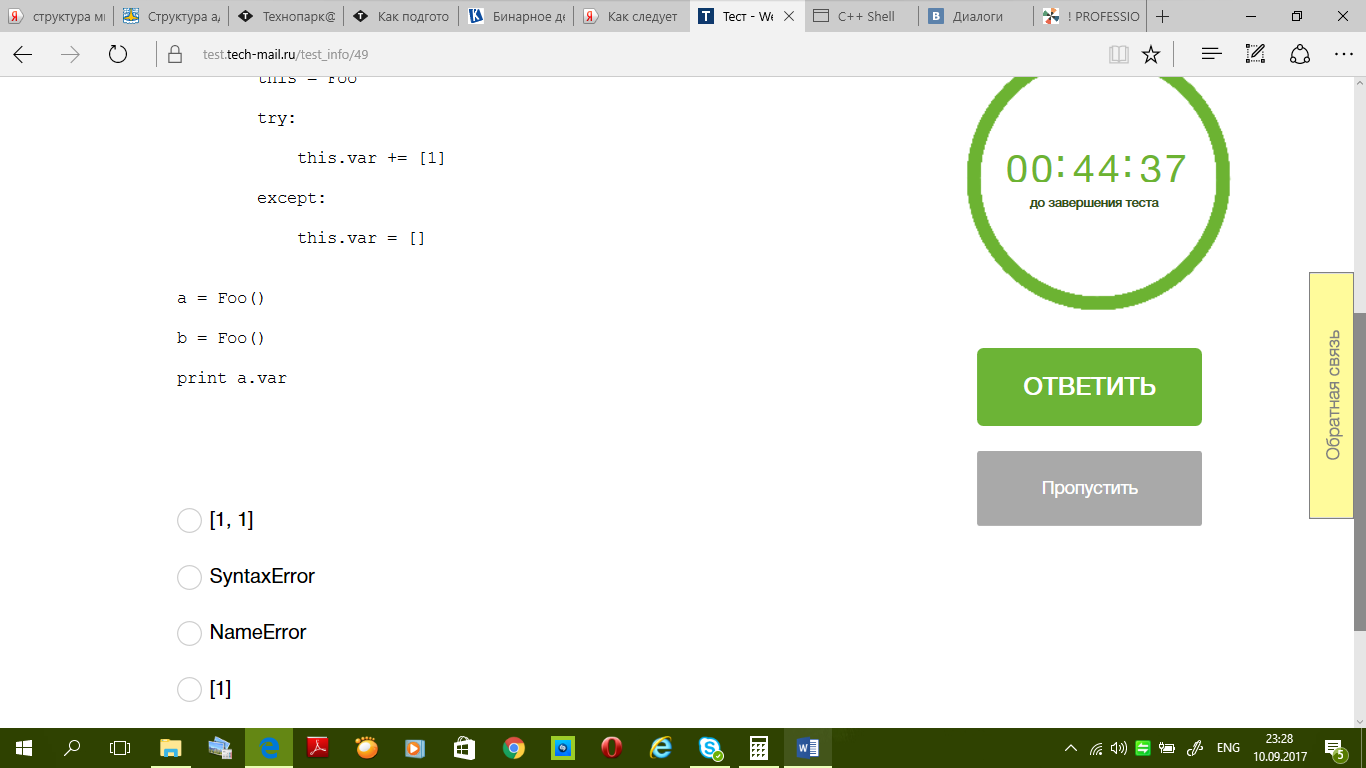
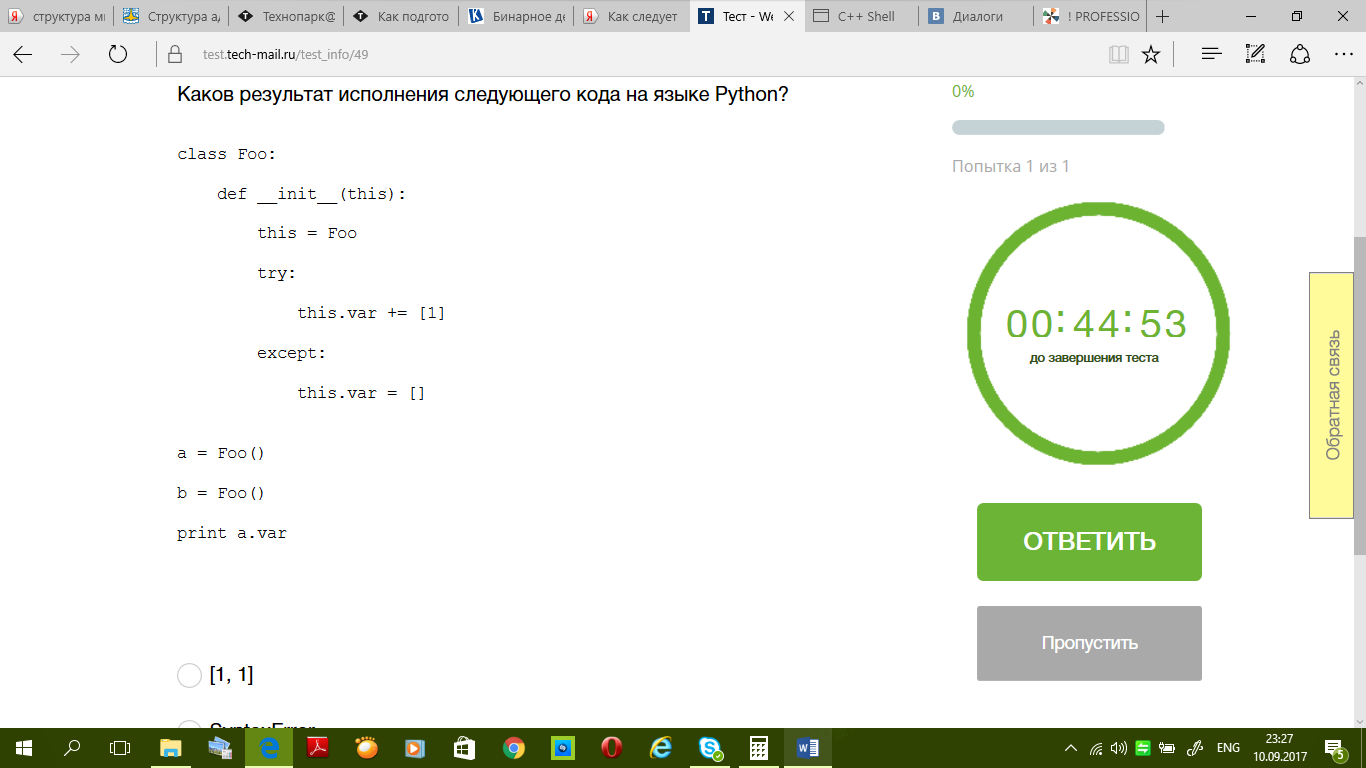
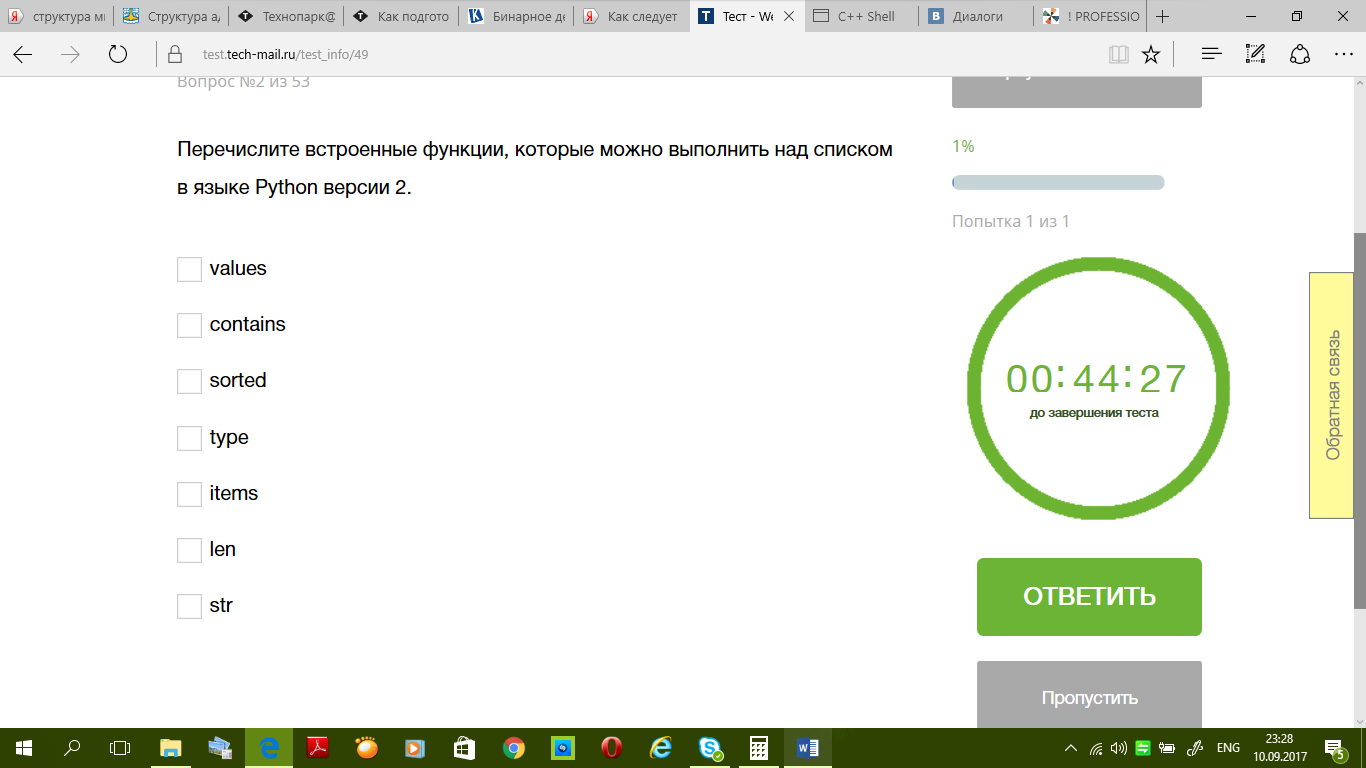
**Вопрос №1 из 53**



Правильный ответ [1]. Изначально в метод \_\_init\_\_ через аргумент this передается экземпляр класса Foo. Но строка this = Foo меняет значение переменной this с экземпляра класса Foo на сам класс Foo. Поэтому теперь this.var будет создавать переменную класса (статический член класса), а не переменную экземпляра. При первом выполнении строки this.var += [1] сработает исключение потому что переменная не проинициализированна и выполнится строка this.var = []. При втором исполнении строки this.var += [1] переменная уже будет проинициализированна значением [] поэтому инструкция отработает успешно выполнив вычисление *[] + [1] = [1]*.

**Вопрос №2 из 53**



Из вышеприведенного перечня ответов для списков определены только фукнции sorted, type, len, str:

l = [**'a'**, **'b'**, **'c'**]

**print** sorted(l) *# ['a', 'b', 'c']*

**print** type(l) *# <type 'list'>*

**print** len(l) *# 3*

**print** str(l) *# ['a', 'b', 'c']*

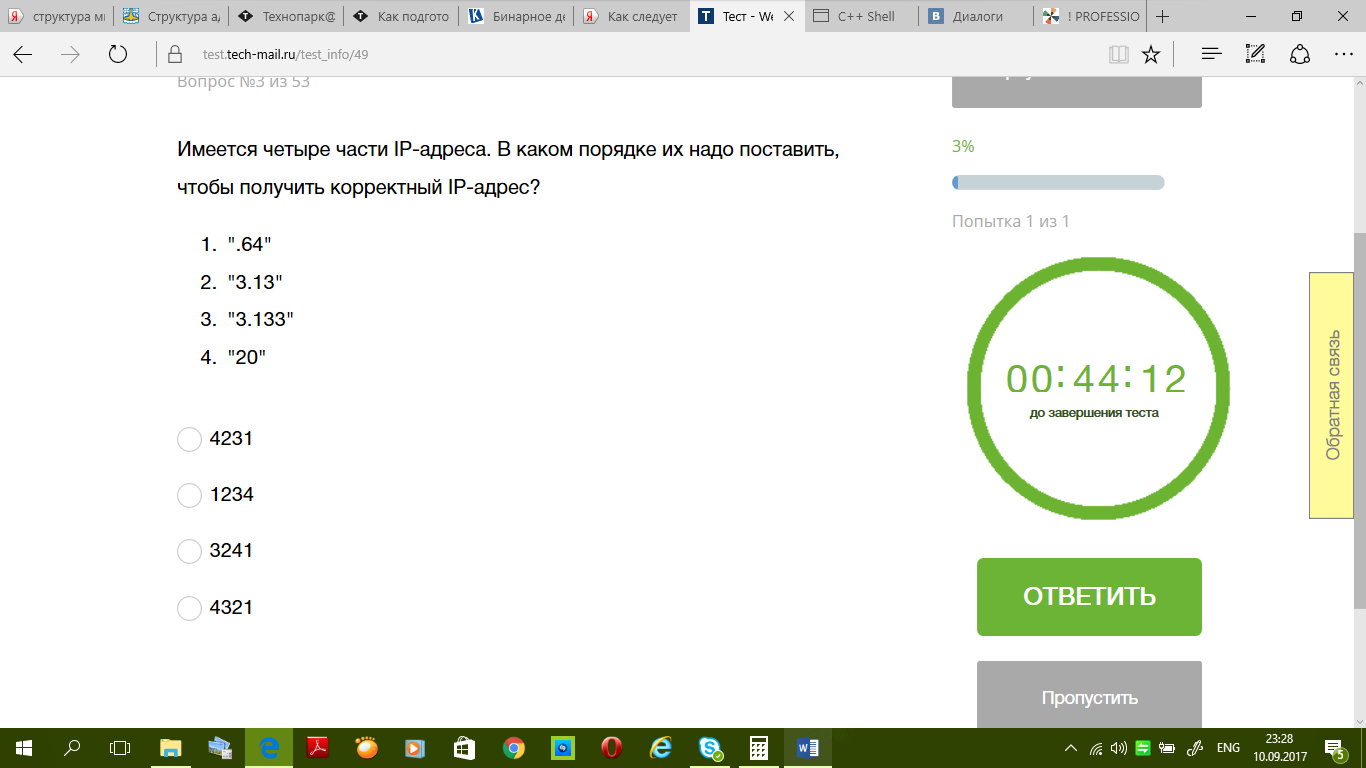
Оставшиеся функции values, contains и items не определены:

*# print values(l) # NameError: name 'values' is not defined*

*# print contains(l) # NameError: name 'contains' is not defined*

*# print items(l) # NameError: name 'items' is not defined*

**Вопрос №3 из 53**

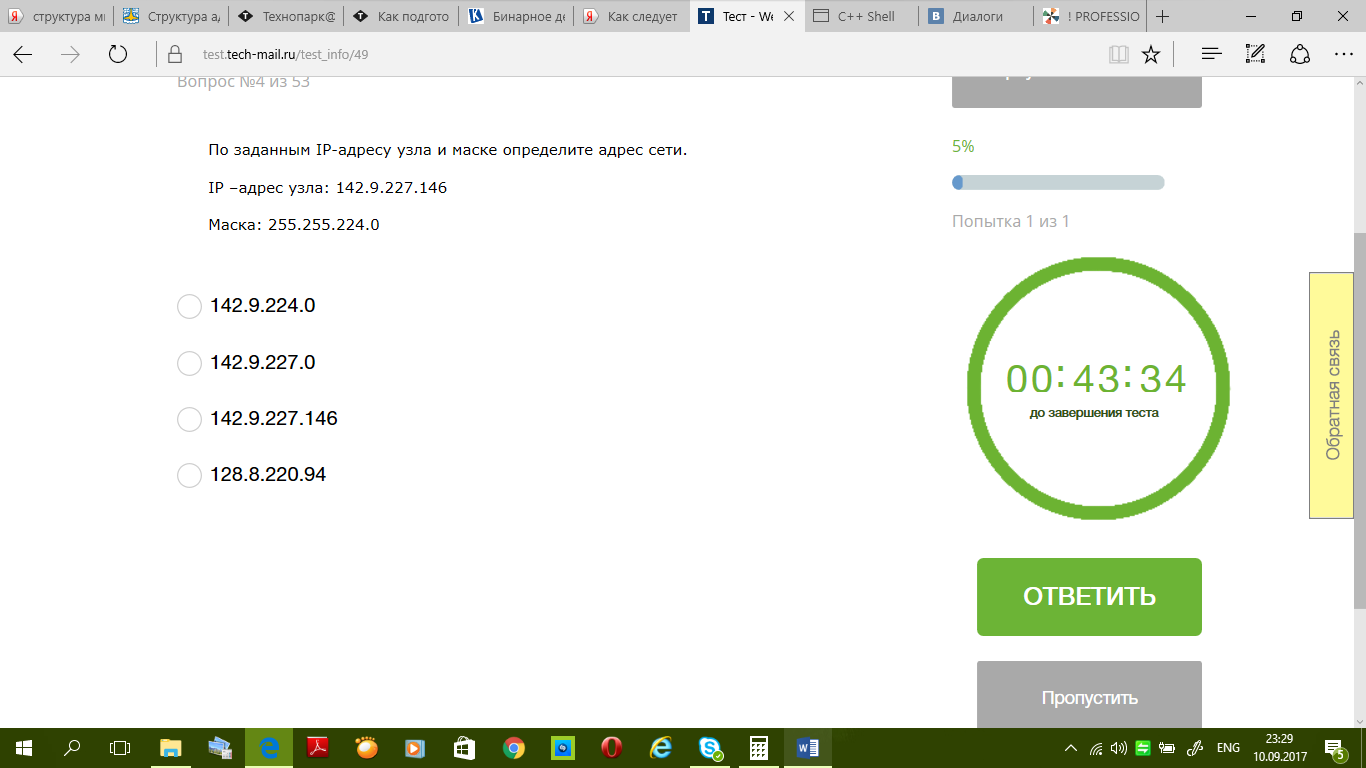


Правильный ответ: 4231

Формой записи IP-адреса (IPv4) является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками:

|  |  |
| --- | --- |
| 4231 | 203.133.133.64 |
| 1234 | .643.133.13320 |
| 3241 | 3.1333.1320.64 |
| 4321 | 203.1333.13.64 |

**Вопрос №4 из 53**



Правильный ответ: 142.9.224.0

IP-адрес состоит из двух частей: номера сети и номера узла.

Чтобы получить адрес сети, зная IP-адрес и маску подсети, необходимо применить к ним операцию поразрядной конъюнкции (логическое И). Например, в случае более сложной маски (битовые операции в IPv6 выглядят аналогично):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IP-адрес: | 10001110 | 00001001 | 11100011 | 10010010 | 142.9.227.146 |
| Маска подсети: | 11111111 | 11111111 | 11100000 | 00000000 | 255.255.224.0 |
| Адрес сети: | 10001110 | 00001001 | 11100000 | 00000000 | 142.9.224.0 |

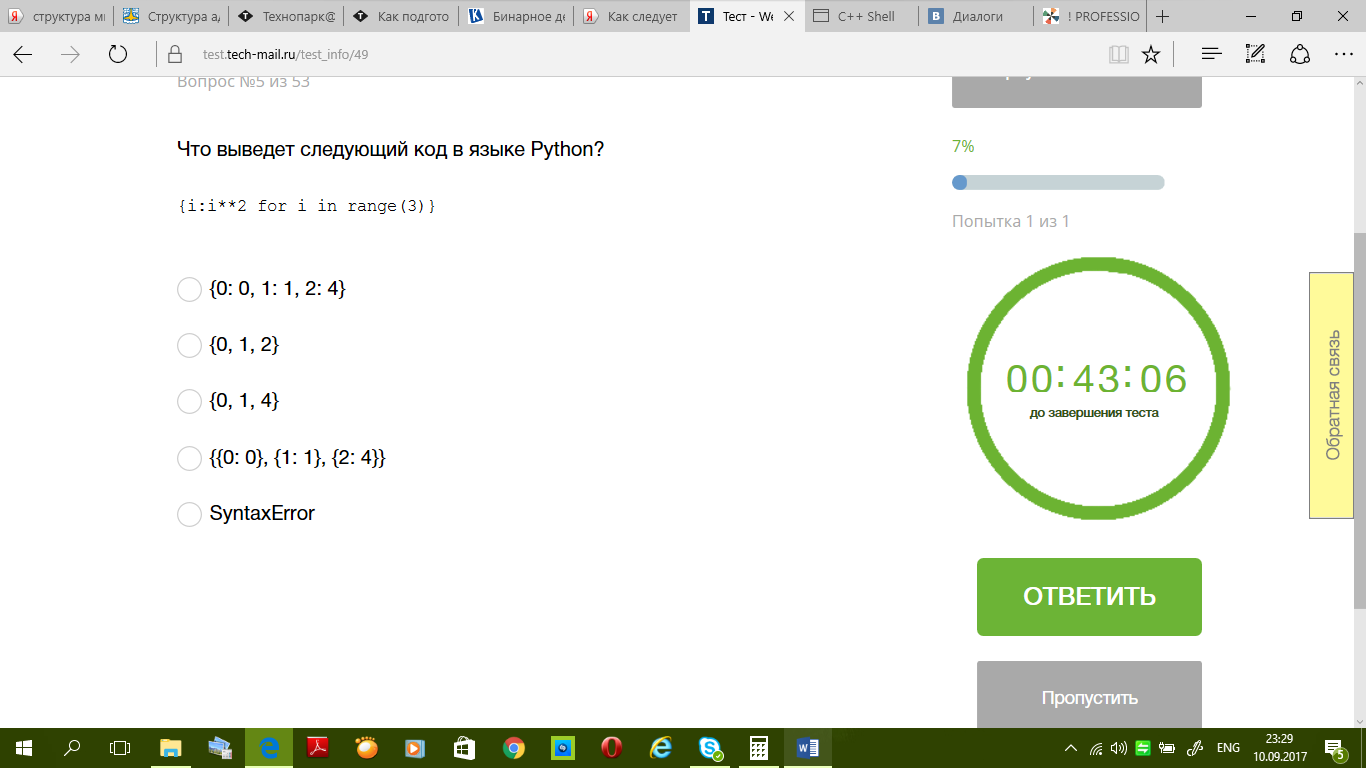
Легенда:

* Часть маски, определяющая адрес сети, состоящая из единиц.
* Адрес сети, который определяется маской подсети.
* Диапазон адресов устройств в этой сети.

Можно использовать какой-нибудь сетевой калькулятор:

<http://www.ip-ping.ru/netcalc/>

**Вопрос №5 из 53**

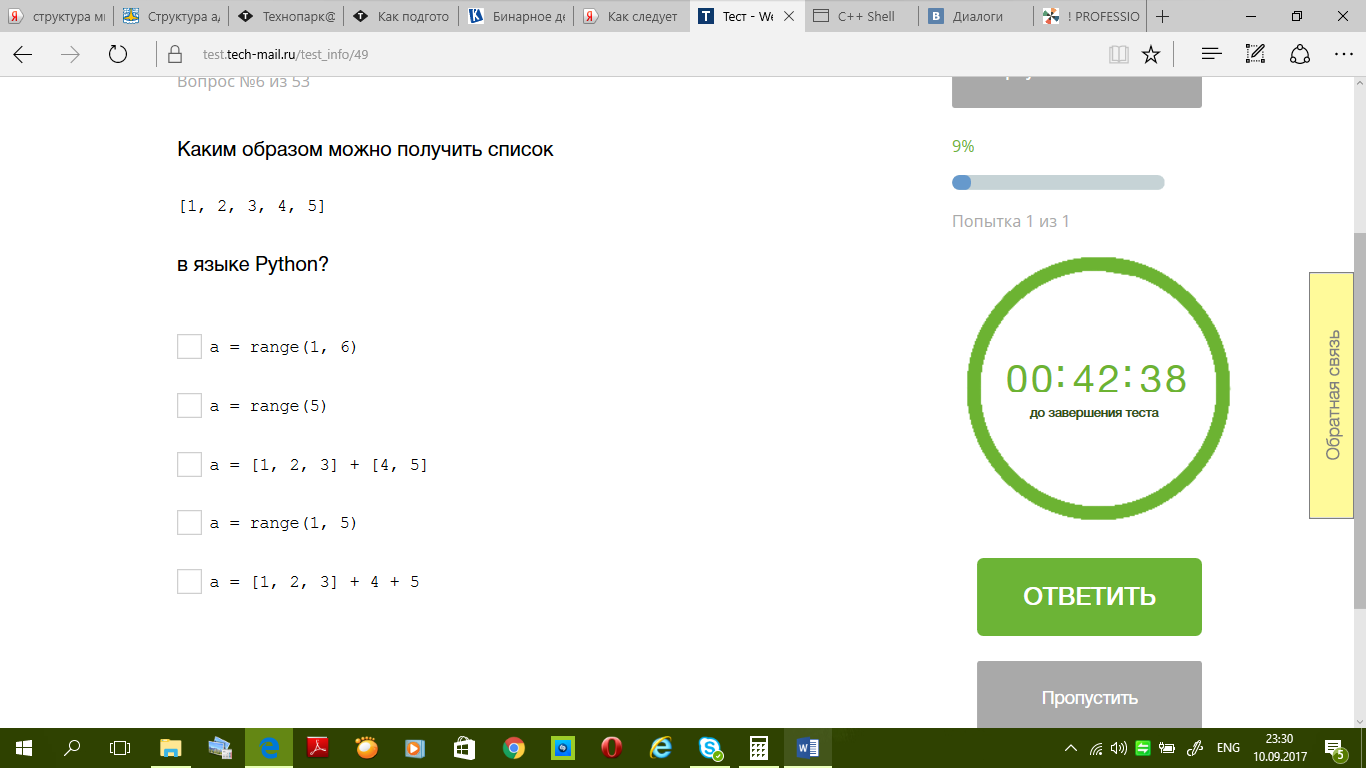


Правильный ответ: *{0: 0, 1: 2, 2: 4}*

Фигурные скобки {} обозначают создание словаря.

Конструкция i:i\*2 **for** i **in** range(3) позволяет циклическим способом задать набор ключей в диапазоне от 0 до 2 и соответственно значений для них в диапазоне от 02 до 22. Потому что для ключа используется просто i, а для значения, которое будет хранится по этому ключу используется i2.

**Вопрос №6 из 53**



Правильные варианты:

a = range(1, 6)

a = [1, 2, 3] + [4, 5]

Вызов функции **range(a, b)** генерирует диапазон от **a** до **b - 1**.

Списки можно складывать друг с другом: [a, b, c] + [d, e] = [a, b, c, d, e].

Нельзя складывать списки с отдельными целочисленными значениями.

Проверка:

a = range(1, 6)

**print** a *# [1, 2, 3, 4, 5]*

a = range(5)

**print** a *# [0, 1, 2, 3, 4]*

a = [1, 2, 3] + [4, 5]

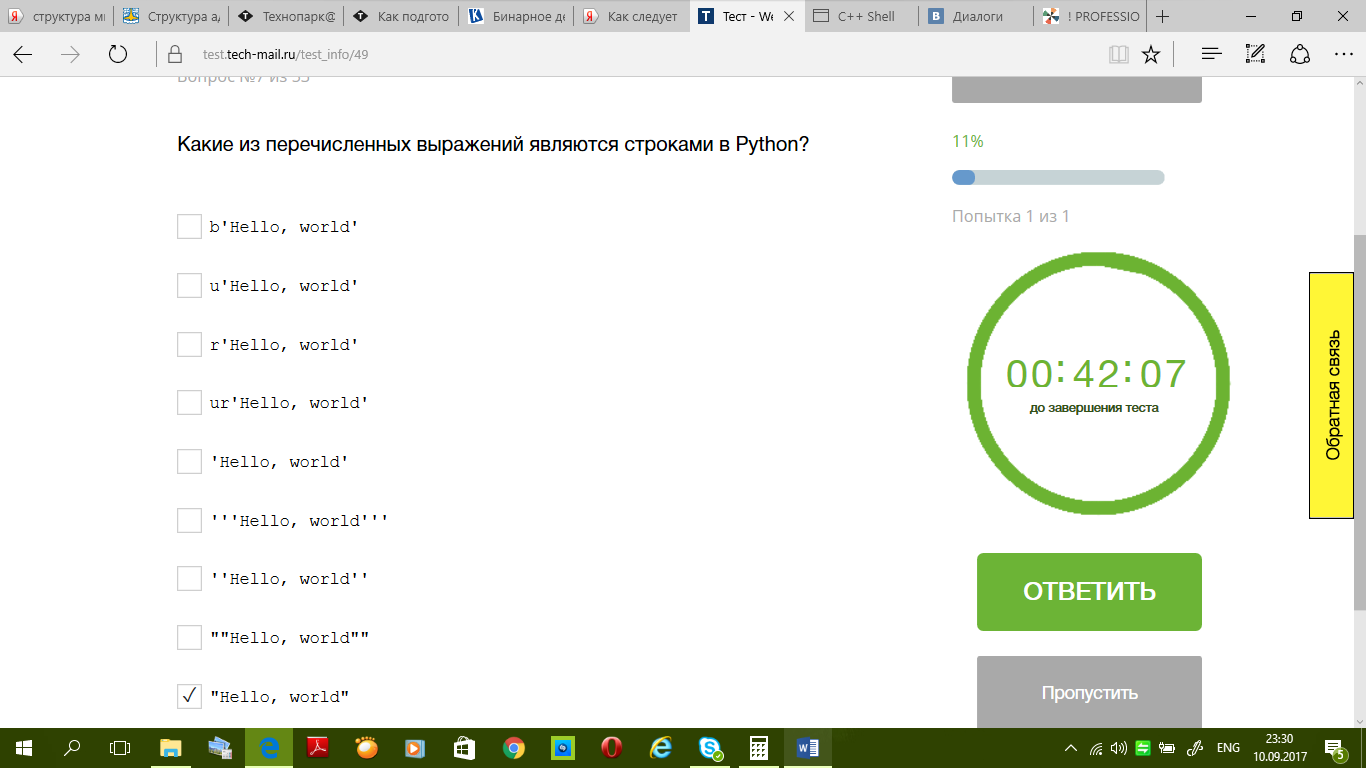
**print** a *# [1, 2, 3, 4, 5]*

a = range(1, 5)

**print** a *# [1, 2, 3, 4]*

*# a = [1, 2, 3] + 4 + 5 # TypeError: can only concatenate list (not "int") to list*

**Вопрос №7 из 53**



Все варианты кроме двух являются правильными.

Неправильные с точки зрения синтаксиса варианты:

*''Hello, world''*

*""Hello, world""*

Проверка:

s1 = **b'Hello, world'** *# байтовая строка*

*# Байтовые строки поддерживают практически все строковые методы.*

s2 = **u'Hello, world'** *# строка Unicode*

s3 = **r'Hello, world'** *# "сырая" строка (механизм экранирования отключается)*

s4 = **ur'Hello, world'** *# "сырая" строка Unicode*

s5 = **'Hello, world'** *# не нужно экранировать кавычки и нужно экранировать апострофы внутри, вот так \'*

*# Многострочный блок текста.*

*# Внутри такой строки возможно присутствие кавычек и апострофов, главное, чтобы не было трех кавычек подряд.*

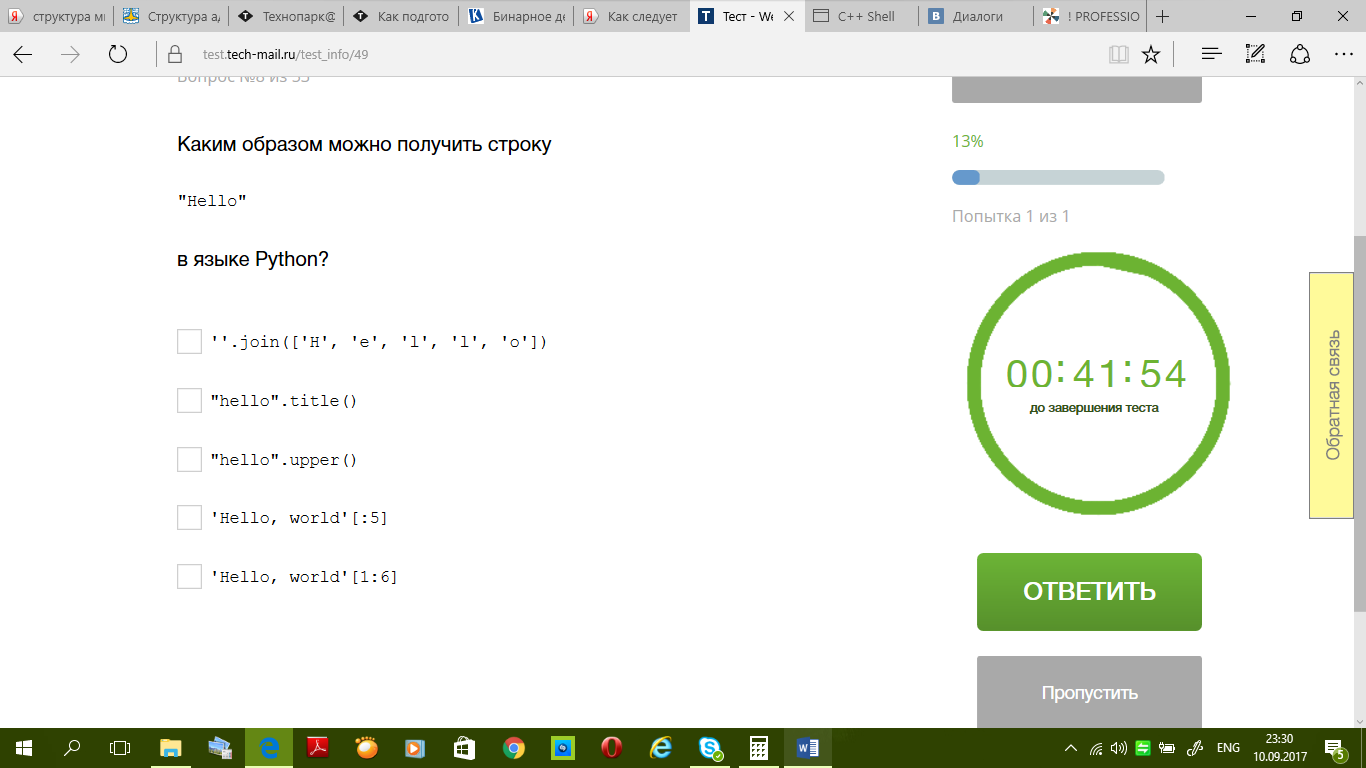
s6 = **'''Hello, world'''**

*# s7 = ''Hello, world'' # SyntaxError: invalid syntax*

*# s8 = ""Hello, world"" # SyntaxError: invalid syntax*

s9 = **"Hello, world"** *# не нужно экранировать апострофы и нужно экранировать кавычки внутри, вот так \"*

**Вопрос №8 из 53**



Правильными являются варианты:

* **''**.join([**'H'**, **'e'**, **'l'**, **'l'**, **'o'**])
* **"hello"**.title()
* **'Hello, world'**[:5]

Функция **join** соединяется символы в строку.

Функция **title** делает заглавной первую букву строки.

Конструкция **my\_collection[start:stop:step]** известка как слайсинг:

* 'Hello, world'[:] — элементы в диапазоне индексов [0;∞), т.е. копия списка;
* 'Hello, world'[:5] — элементы в диапазоне индексов [0;5), т.е. первые 5 элементов;
* 'Hello, world'[1:6] — элементы в диапазоне индексов [1;6), т.е. откидываем первый элемент.

Проверка:

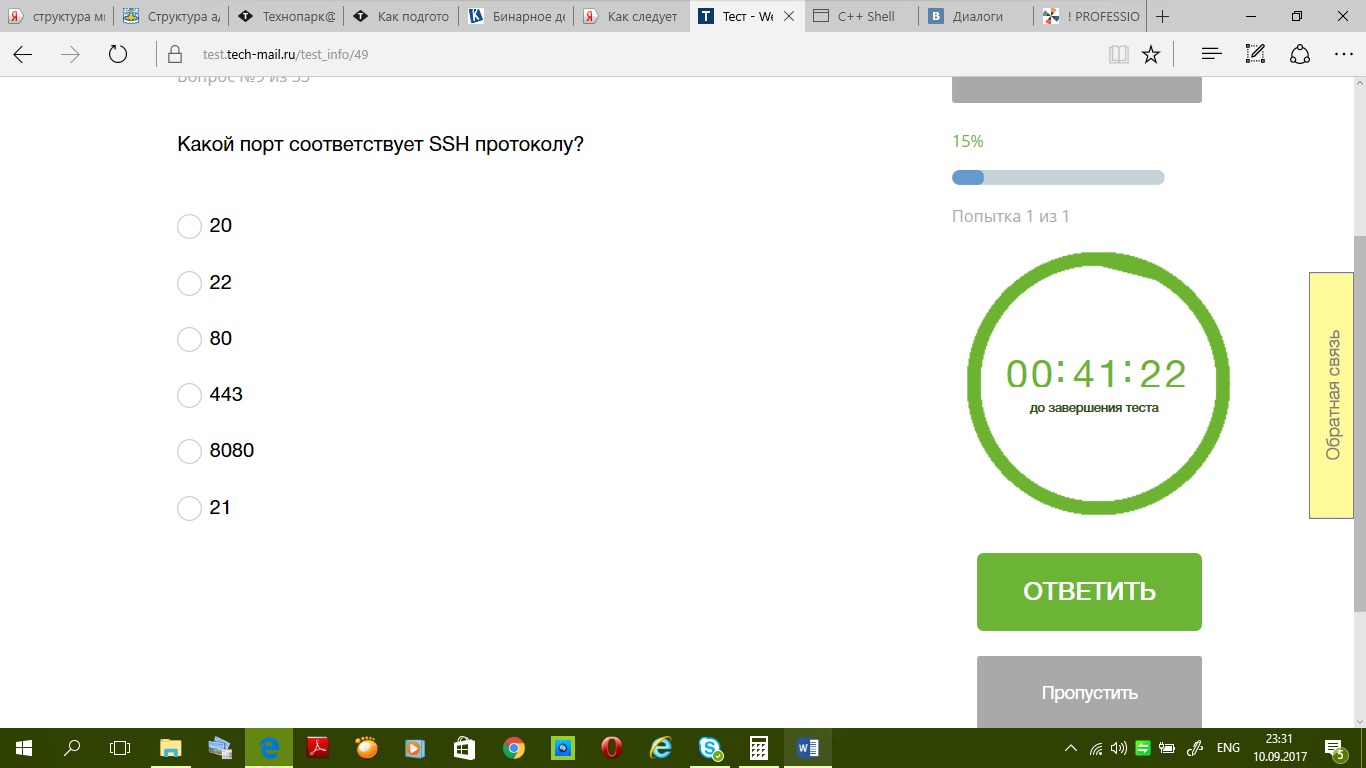
**print ''**.join([**'H'**, **'e'**, **'l'**, **'l'**, **'o'**]) *# Hello*

**print "hello"**.title() *# Hello*

**print "hello"**.upper() *# HELLO*

**print 'Hello, world'**[:5] *# Hello*

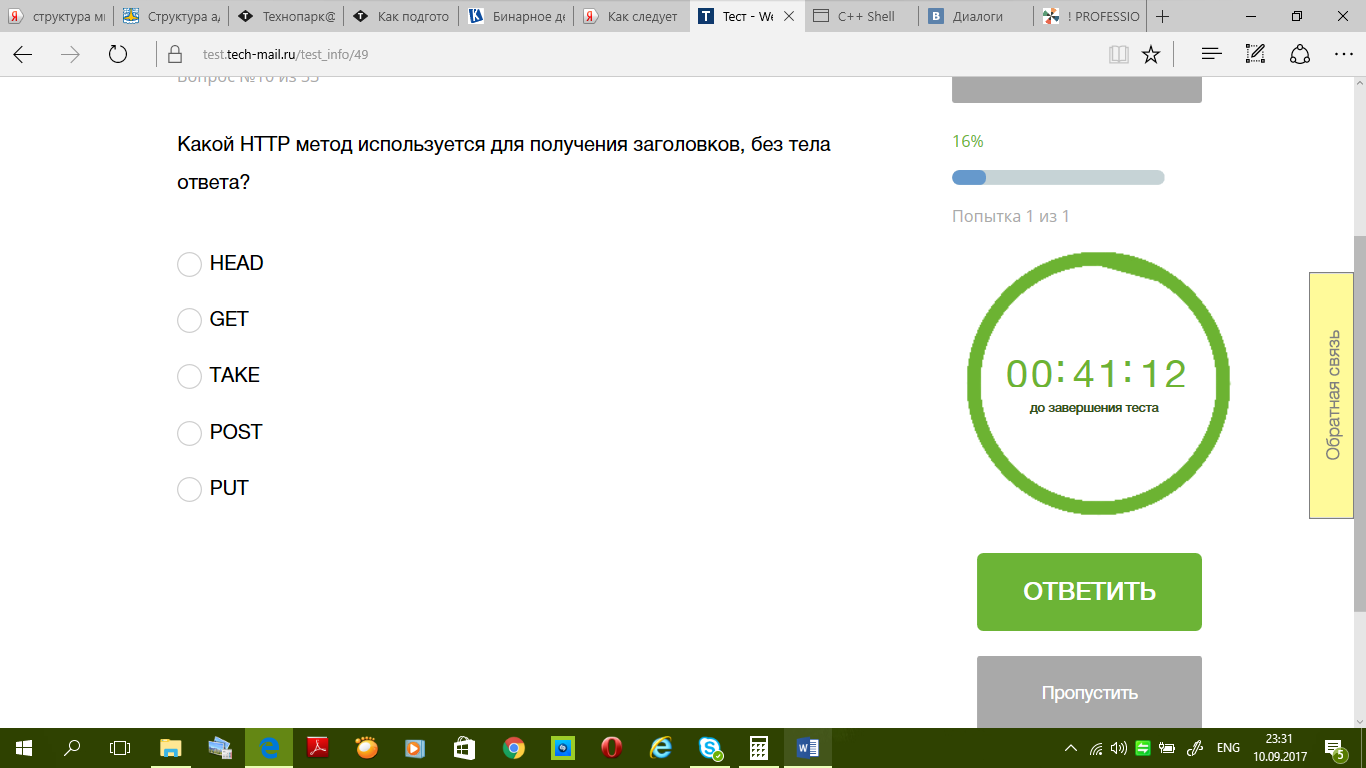
**print 'Hello, world'**[1:6] *# ello,*

**Вопрос №9 из 53**

Правильный ответ: 22

Общеизвестные порты:

* **20/TCP:** FTP-DATA — для передачи данных FTP
* **22/TCP,UDP:** SSH (Secure SHell) — криптографический сетевой протокол для безопасной передачи данных
* **80/TCP,UDP:** HTTP (HyperText Transfer Protocol); ранее — WWW
* **443/TCP,UDP:** HTTPS (HyperText Transfer Protocol Secure) — HTTP с шифрованием по SSL или TLS
* **8080/TCP:** HTTP alternate (http\_alt)—commonly used for Web proxy and caching server, or for running a Web server as a non-root user
* **21/TCP:** FTP — для передачи команд FTP

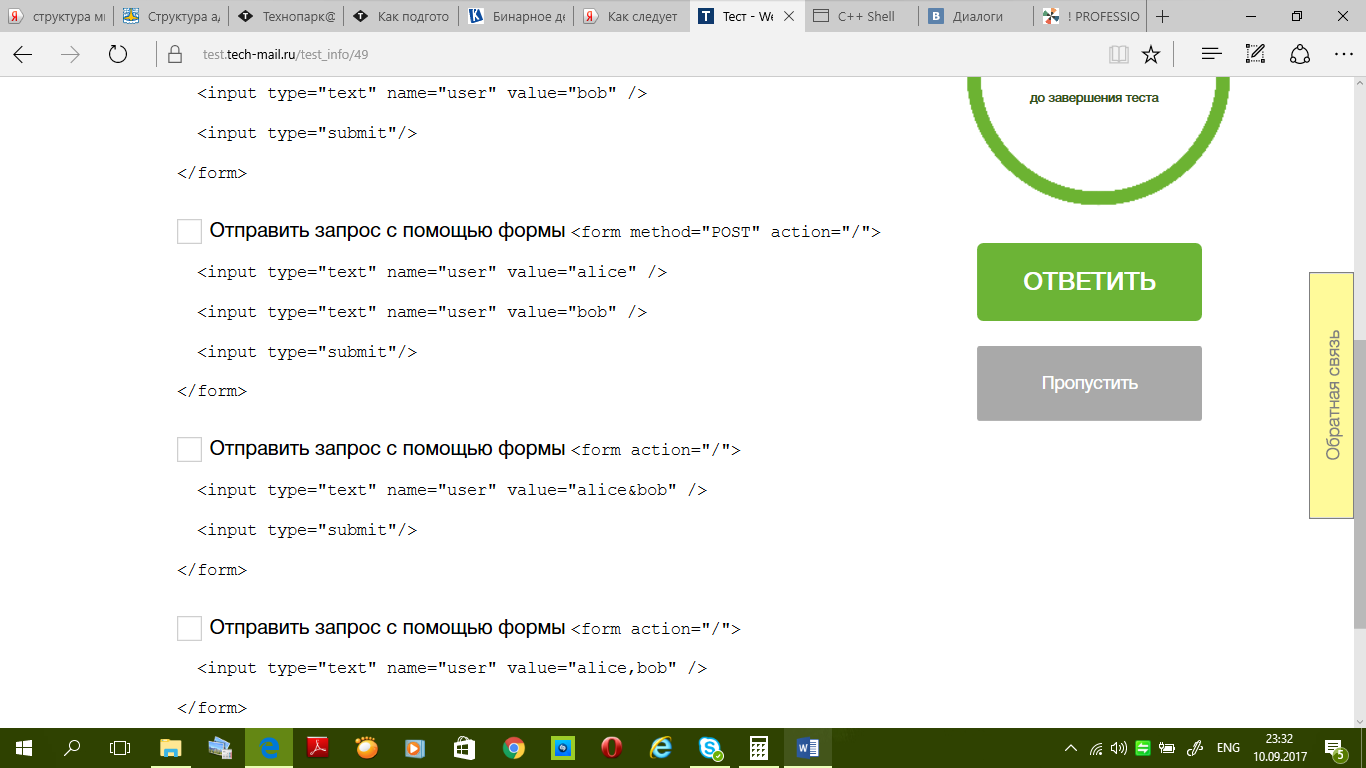
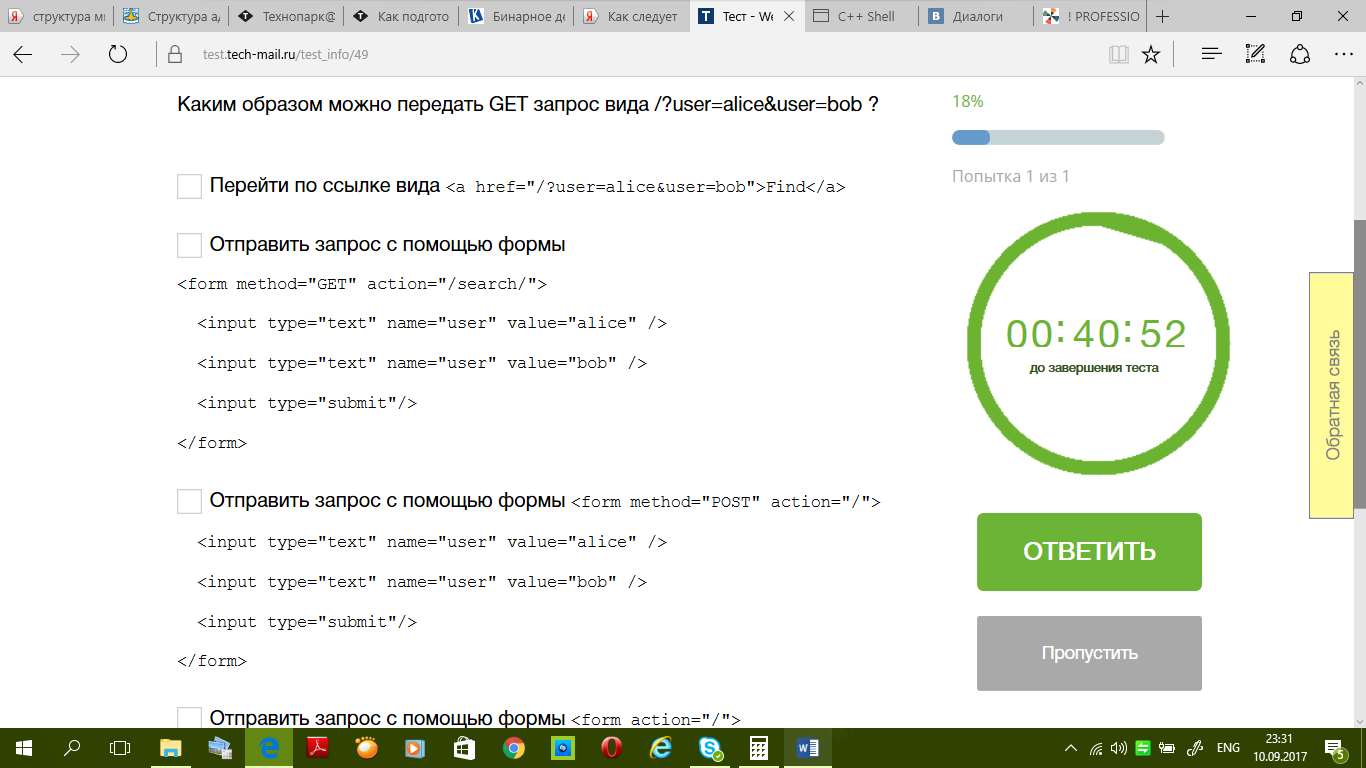
**Вопрос №10 из 53**

Правильный ответ: HEAD

Метода ~~TAKE~~ не существует.

Методы HTTP:

* HEAD - Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело.
* GET - Используется для запроса содержимого указанного ресурса.
* POST - Применяется для передачи пользовательских данных заданному ресурсу.
* PUT - Применяется для загрузки содержимого запроса на указанный в запросе URI.

**Вопрос №11 из 53**

Первый ответ правильный:

<a href="/?user=alice&user=bob">Find</a>

Все ответы:

1. GET-запрос вида example.com/?user=alice&user=bob
2. GET-запрос вида example.com/search/?user=alice&user=bob
3. POST-запрос на example.com
4. GET-запрос вида example.com/?user=alice%26bob (%26 вместо &)
5. GET-запрос вида example.com/?user=alice%2Cbob (%2C вместо ,)

Пример: <https://codepen.io/anon/pen/yvgGXJ>

<p>#1</p>

<!-- example.com/?user=alice&user=bob -->

<a href="/?user=alice&user=bob">Find</a>

<p>#2</p>

<!-- example.com/search/?user=alice&user=bob -->

<form method="GET" action="/search/">

<input type="text" name="user" value="alice" />

<input type="text" name="user" value="bob" />

<input type="submit">

</form>

<p>#3</>

<!-- example.com -->

<form method="POST" action="/">

<input type="text" name="user" value="alice" />

<input type="text" name="user" value="bob" />

<input type="submit">

</form>

<p>#4</p>

<!-- example.com/?user=alice%26bob -->

<form action="/">

<input type="text" name="user" value="alice&bob" />

<input type="submit">

</form>

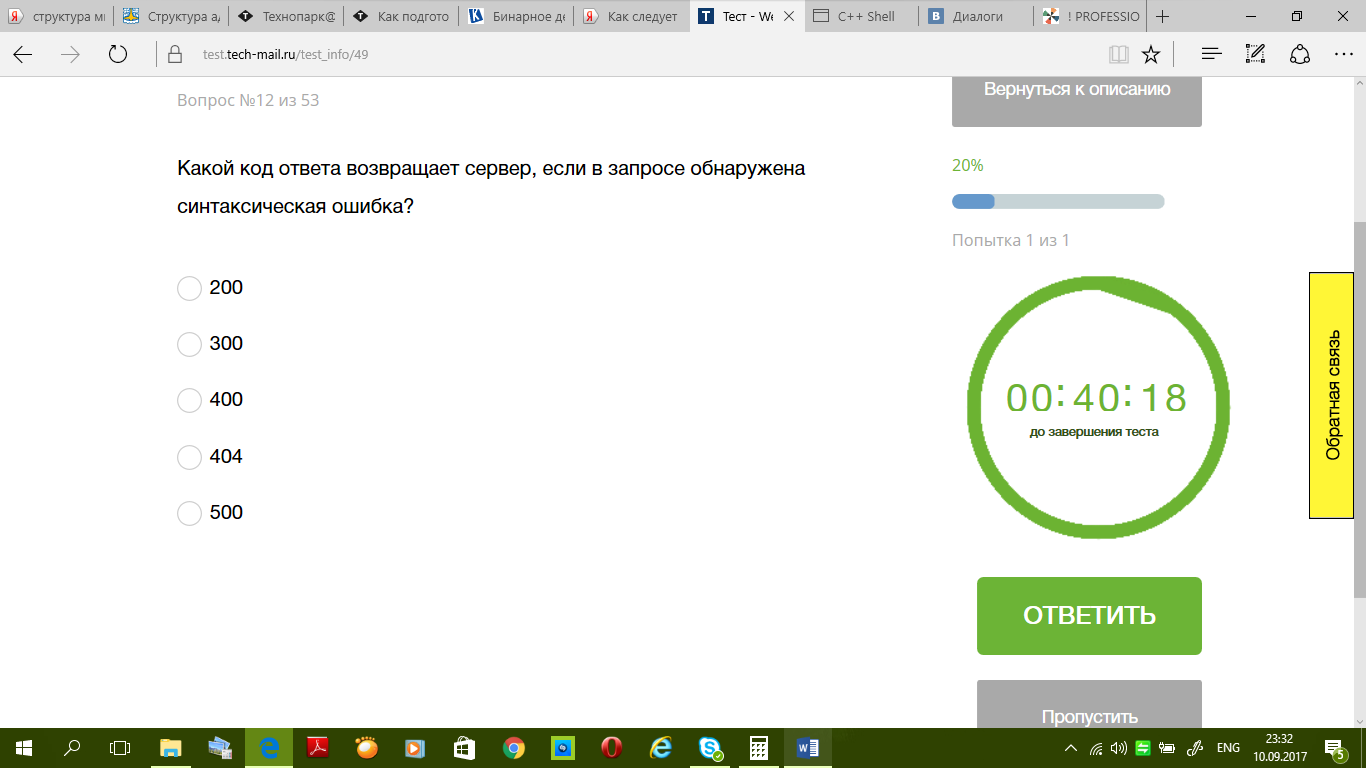
<p>#5</p>

<!-- example.com/?user=alice%2Cbob -->

<form action="/">

<input type="text" name="user" value="alice,bob" />

</form>

**Вопрос №12 из 53**

Правильный ответ: 400

400 Bad Request — сервер обнаружил в запросе клиента синтаксическую ошибку. Появился в HTTP/1.0.

Список кодов состояния HTTP:

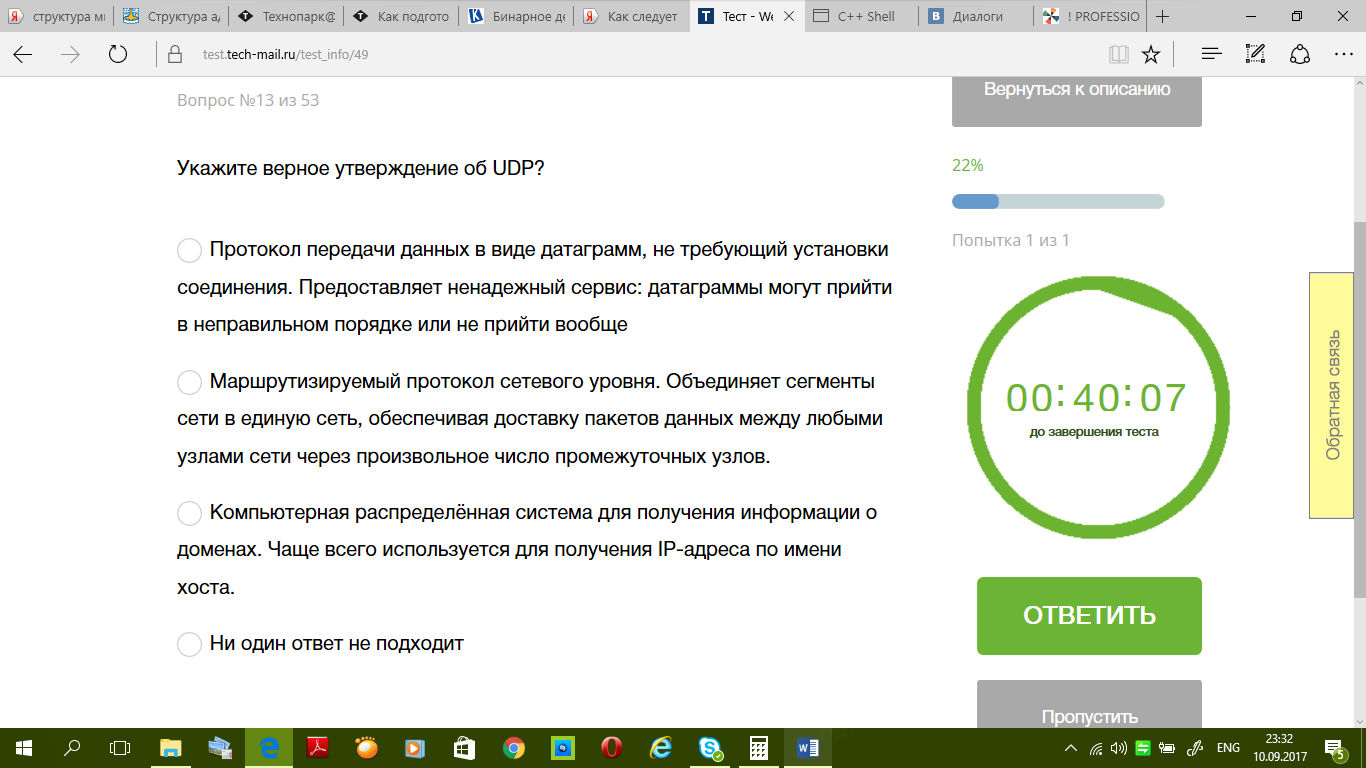
[200 OK](https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP" \l "200) («хорошо»)

[300 Multiple Choices](https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP" \l "300) («множество выборов»)[[](https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP" \l "cite_note-3digits-2)

[400 Bad Request](https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP" \l "400) («плохой, неверный запрос»)

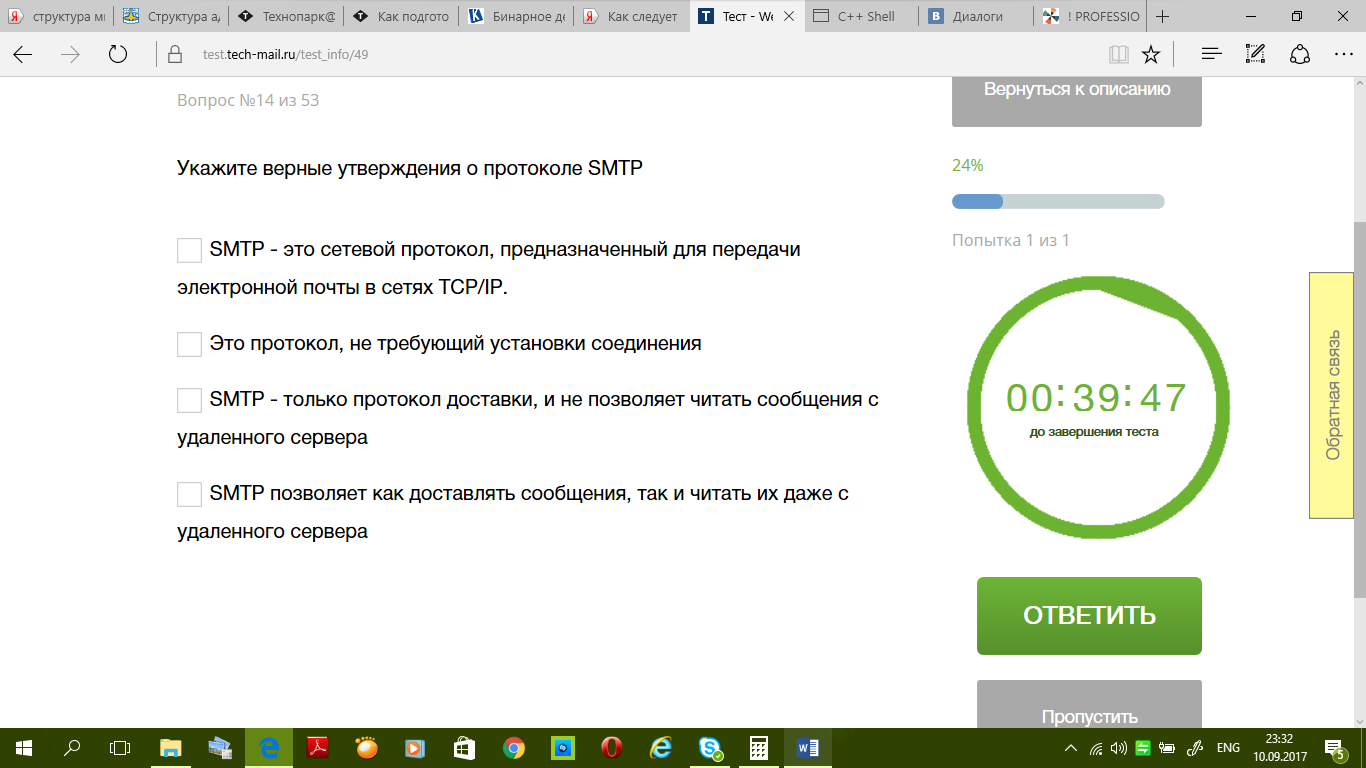
[404 Not Found](https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP" \l "404) («не найдено»)

[500 Internal Server Error](https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP" \l "500) («внутренняя ошибка сервера»)

**Вопрос №13 из 53**

Первый вариант правильный.

С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

**Вопрос №14 из 53**

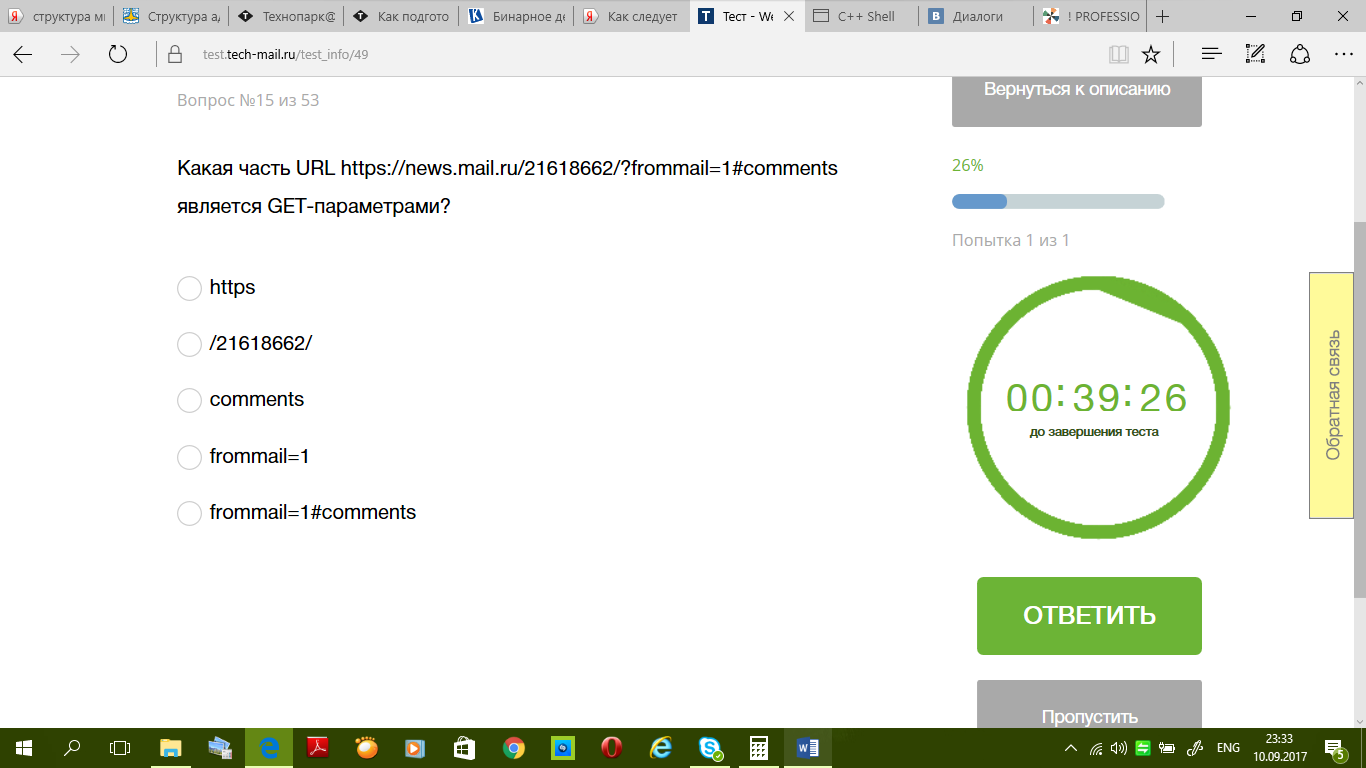
Правильные варианты:

* SMTP — это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.
* SMTP — только протокол доставки, и не позволяет читать сообщения с удаленного сервера.

**Протокол SMTP предназначен для передачи исходящей почты** с использованием порта TCP 25.

В то время, как электронные почтовые серверы и другие агенты пересылки сообщений используют SMTP для отправки и получения почтовых сообщений, **работающие на пользовательском уровне клиентские почтовые приложения обычно используют SMTP только для отправки сообщений на почтовый сервер для ретрансляции. Для получения сообщений клиентские приложения обычно используют либо POP** (англ. Post Office Protocol — протокол почтового отделения), либо IMAP (англ. Internet Message Access Protocol), либо патентованные системы (такие как Microsoft Exchange и Lotus Notes/Domino) для доступа к учётной записи своего почтового ящика на сервере.

**SMTP — требующий соединения текстовый протокол**, по которому отправитель сообщения связывается с получателем посредством выдачи командных строк и получения необходимых данных через надёжный канал, в роли которого обычно выступает TCP-соединение (Transmission Control Protocol — протокол управления передачей).

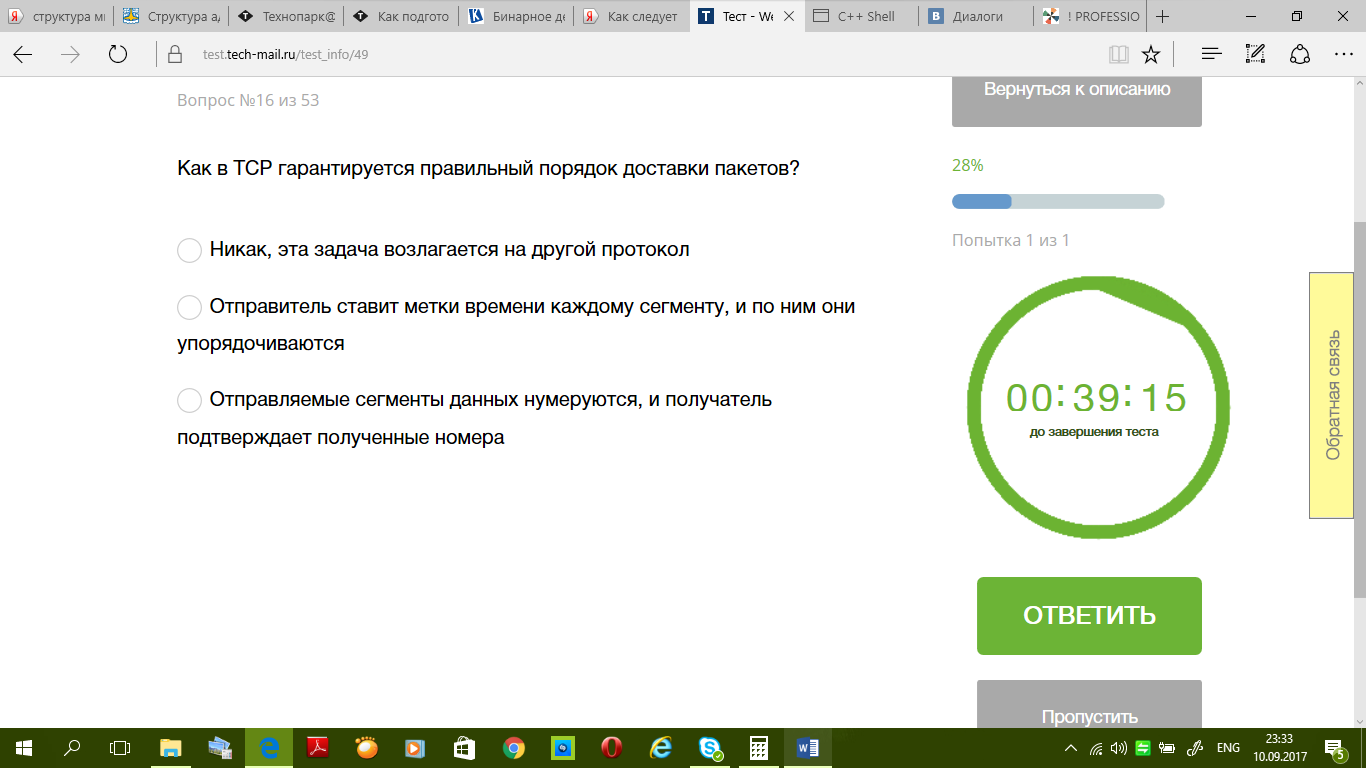
**Вопрос №15 из 53**

Правильный ответ: **frommail=1**

GET-параметры идут после вопросительного знака ? и до знака решетки #.

Другие ответы:

* **https** — это протокол
* **/21618662/** — это часть URL
* **comments** — это якорь (закладка с уникальным именем на определенном месте веб-страницы) на веб-странице

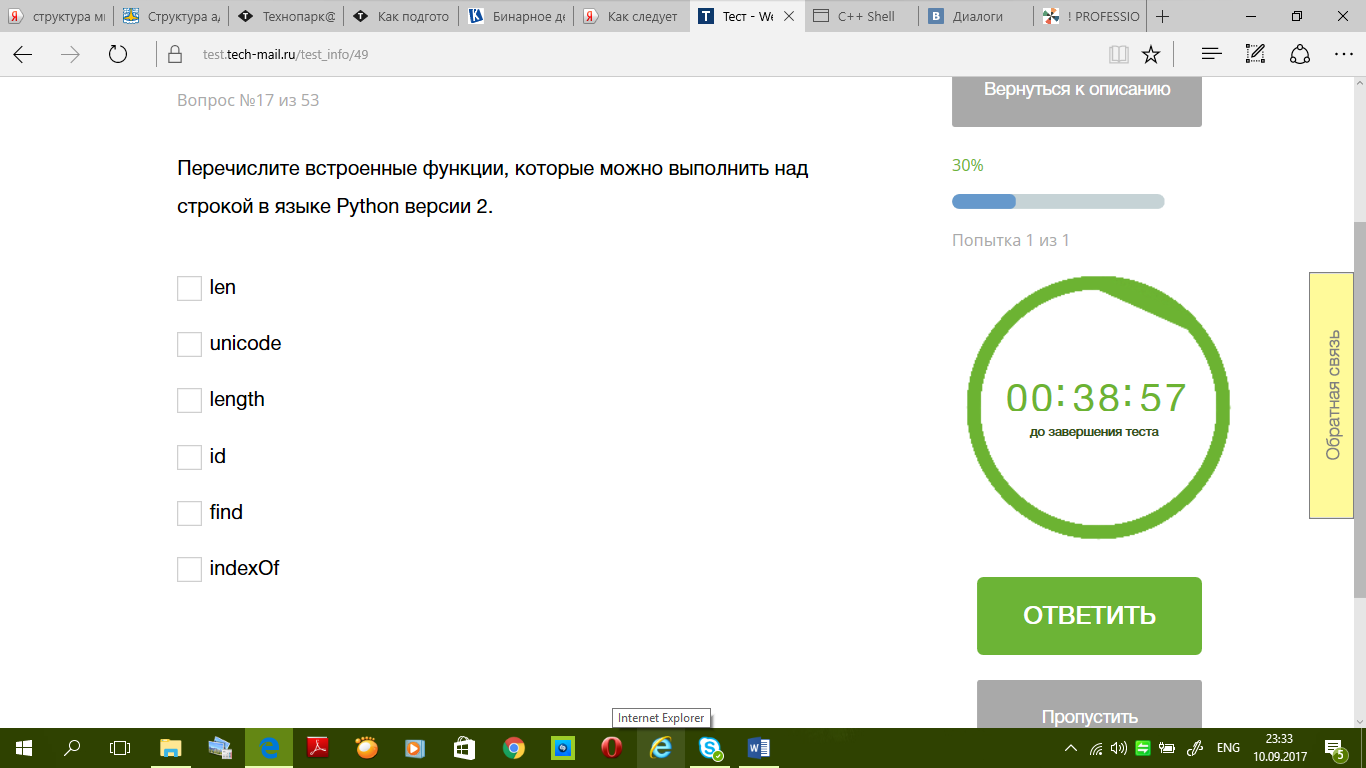
**Вопрос №16 из 53**

Правильный ответ:

Отправляемые сегменты данных нумеруются, и получатель подтверждает полученные номера.

Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым, в отличие от UDP, целостность передаваемых данных и уведомление отправителя о результатах передачи.

Одним из фундаментальных аспектов TCP является нумерация данных — каждый октет, передаваемый через соединение TCP имеет свой порядковый номер. Поскольку каждый октет пронумерован, для любого из октетов может быть передано подтверждение (acknowledgment). Механизм подтверждений является кумулятивным (накопительным), поэтому подтверждение для порядкового номера X показывает, что все октеты до X (но не включая сам октет с номером X) были получены. Этот механизм позволяет обнаруживать дубликаты данных при использовании повторной передачи. Нумерация октетов в сегменте начинается от заголовка, т. е. октет, следующий сразу после заголовка, имеет наименьший порядковый номер, а номера следующих октетов последовательно возрастают.

**Вопрос №17 из 53**

Надо строкой можно выполнять следующие методы:

len

unicode

id

Оставшиеся методы не определены:

*length*

*find*

*indexOf*

Проверка:

s = **'Hello, World!'**

**print** len(s) *# 13*

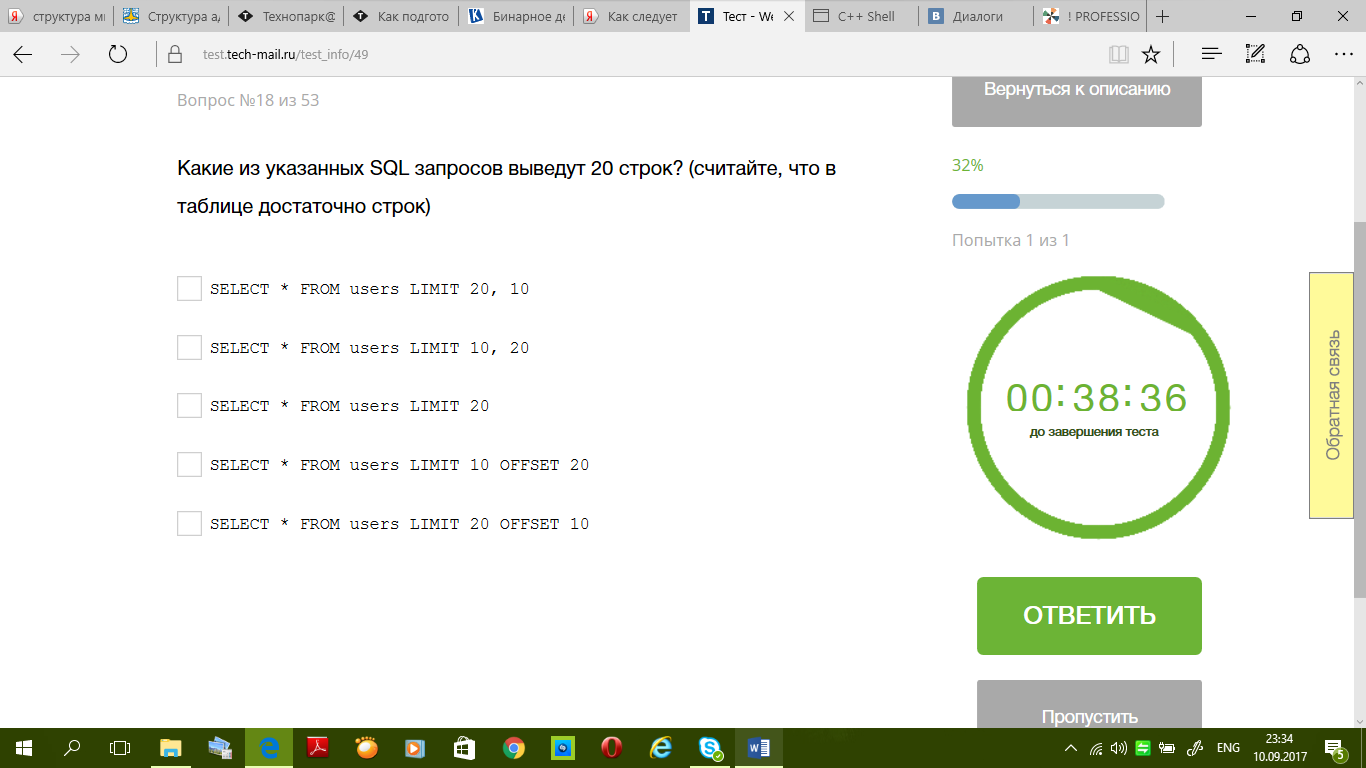
**print** unicode(s) *# Hello, World!*

**print** id(s) *# 140369511854520*

*# print length(s) # NameError: name 'length' is not defined*

*# print find(s) # NameError: name 'find' is not defined*

*# print indexOf(s) # NameError: name 'indexOf' is not defined*

**Вопрос №18 из 53**

Правильные ответы:

* SELECT \* FROM users LIMIT 10, 20
* SELECT \* FROM users LIMIT 20
* SELECT \* FROM users LIMIT 20 OFFSET 10

Другие ответы:

* SELECT \* FROM users LIMIT 20, 10 — выводит 10 строк в диапазоне [21;30]
* SELECT \* FROM users LIMIT 10, 20 — выводит 20 строк в диапазоне [11;30]
* SELECT \* FROM users LIMIT 20 - выводит 20 строк в диапазоне [1;20]
* SELECT \* FROM users LIMIT 10 OFFSET 20 — выводит 10 строк в диапазоне [21;30]
* SELECT \* FROM users LIMIT 20 OFFSET 10 — выводит 20 строк в диапазоне [11;30]

Cинтаксис: [LIMIT [offset,] rows]

Пример: [http://sqlfiddle.com/#!9/dc94c/5](http://sqlfiddle.com/" \l "!9/dc94c/5)

CREATE TABLE IF NOT EXISTS `users` (

`id` int(6) unsigned NOT NULL,

PRIMARY KEY (`id`)

) DEFAULT CHARSET=utf8;

INSERT INTO `users` (`id`) VALUES

('1'),('2'),('3'),('4'),('5'),('6'),('7'),('8'),('9'),('10'),('11'),('12'),('13'),('14'),('15'),('16'),('17'),('18'),('19'),('20'),('21'),('22'),('23'),('24'),('25'),('26'),('27'),('28'),('29'),('30'),('31'),('32'),('33'),('34'),('35'),('36'),('37'),('38'),('39'),('40'),('41'),('42'),('43'),('44'),('45'),('46'),('47'),('48'),('49'),('50');

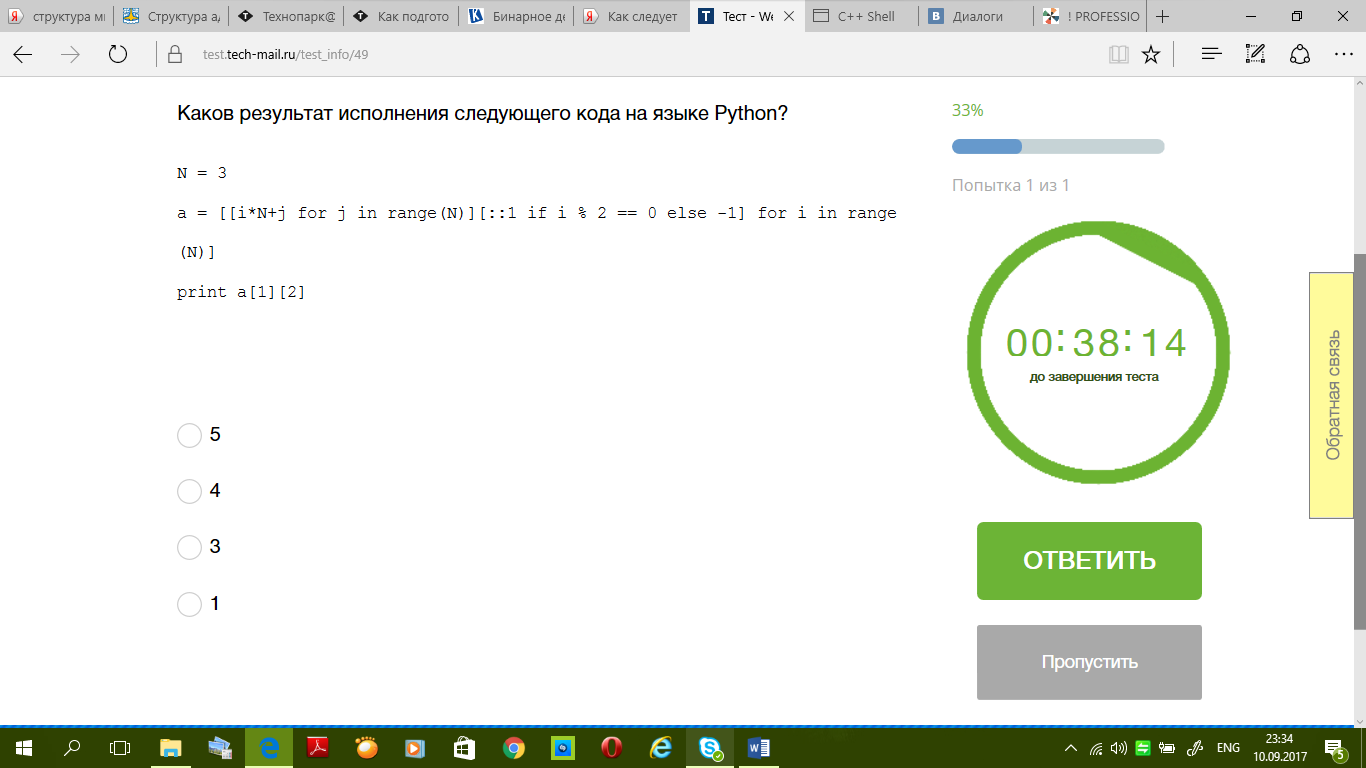
-- SELECT \* FROM users LIMIT 20, 10

-- SELECT \* FROM users LIMIT 10, 20

-- SELECT \* FROM users LIMIT 20

-- SELECT \* FROM users LIMIT 10 OFFSET 20

SELECT \* FROM users LIMIT 20 OFFSET 10

**Вопрос №19 из 53**

Правильный ответ: 3

Квадратные скобки a = [] обозначают создание списка a.

Конструкция внутри квадратных скобок

[i\*N+j **for** j **in** range(N)][::1 **if** i % 2 == 0 **else** -1] **for** i **in** range(N)

обозначает создание N строк: **for** i **in** range(N)

|  |
| --- |
| row |
| 0 |
| 1 |
| 2 |

по N столбцов в каждой: [i\*N+j **for** j **in** range(N)]

|  |  |  |  |
| --- | --- | --- | --- |
| row \ col | j=0 | j=1 | j=2 |
| i=0 | i\*N\*+j=0\*3+0=**0** | i\*N\*+j=0\*3+1=**1** | i\*N\*+j=0\*3+2=**2** |
| i=1 | i\*N\*+j=1\*3+0=**3** | i\*N\*+j=1\*3+1=**4** | i\*N\*+j=1\*3+2=**5** |
| i=2 | i\*N\*+j=2\*3+0=**6** | i\*N\*+j=2\*3+1=**7** | i\*N\*+j=2\*3+2=**8** |

Конструкция:

[::1 **if** i % 2 == 0 **else** -1]

является конструкцией слайсинга, синтаксис: my\_collection[старт:стоп:шаг]

В данном случае это означает, что каждая строка в зависимости от нечетности/четности будет модифицирована следующим образом:

[0,1,2][::1 if 0 % 2 == 0 else -1] ⟶ [0,1,2][::1] ⟶ [0, 1, 2]

[3,4,5][::1 if 1 % 2 == 0 else -1] ⟶ [3,4,5][::-1] ⟶ [5, 4, **3**]

[6,7,8][::1 if 2 % 2 == 0 else -1] ⟶ [6,7,8][::1] ⟶ [6, 7, 8]

Отрицательное значение шага — перебор ведём в обратном порядке справа налево.

**Вопрос №20 из 53**

Над списком можно выполнить следующие методы:

* **list**
* **max**
* **min**
* **len**
* **sorted**

Методы списка:

* index
* count
* remove
* append

Проверка:

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

***# Надо списком:***

**print** list(l) *# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]*

**print** max(l) *# 9*

**print** min(l) *# 0*

*# index(l) # NameError: name 'index' is not defined*

*# count(l) # NameError: name 'count' is not defined*

**print** len(l) *# 10*

**print** sorted(l) *# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]*

*# remove(l) # NameError: name 'remove' is not defined*

*# append(l) # NameError: name 'append' is not defined*

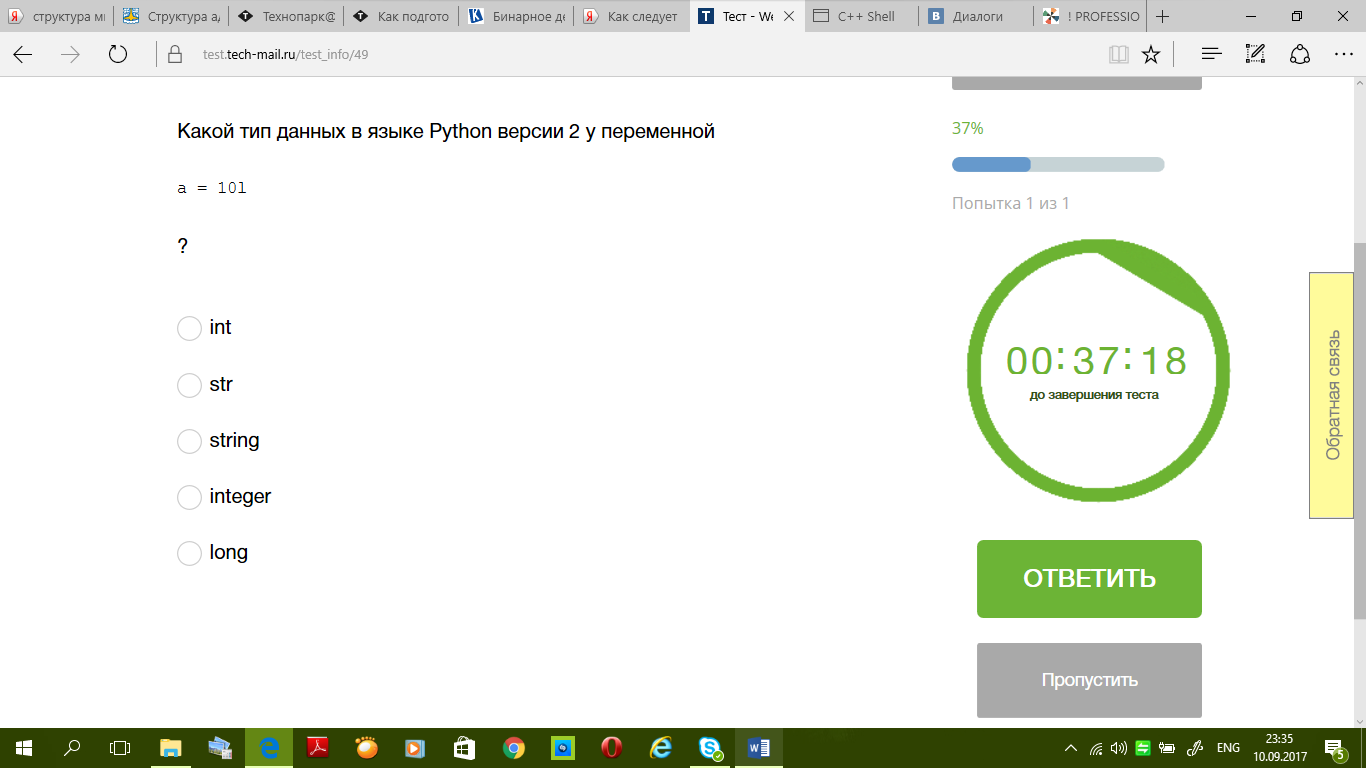
***# У списка:***

**print** l.index(5) *# 5*

**print** l.count(6) *# 1*

**print** l.remove(5) *# None*

**print** l.append(5) *# None*

**Вопрос №21 из 53**

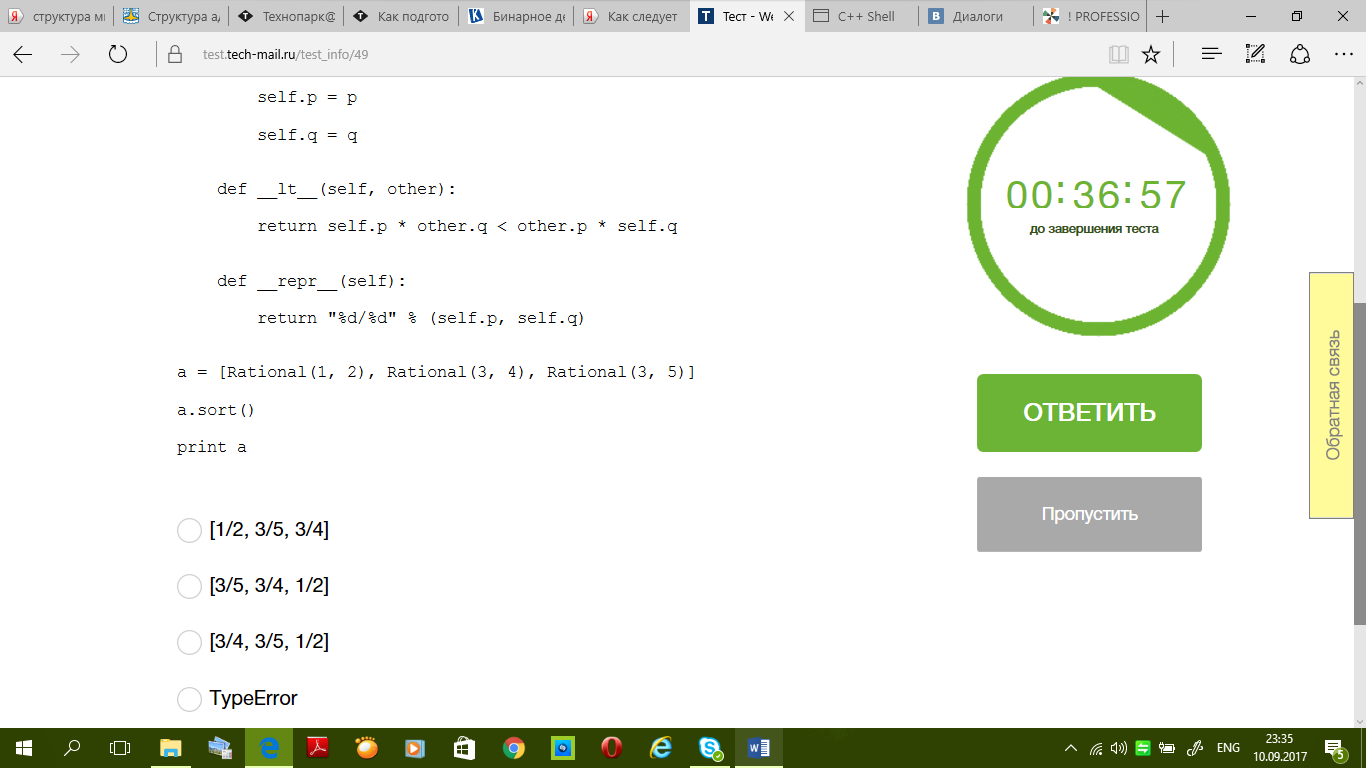
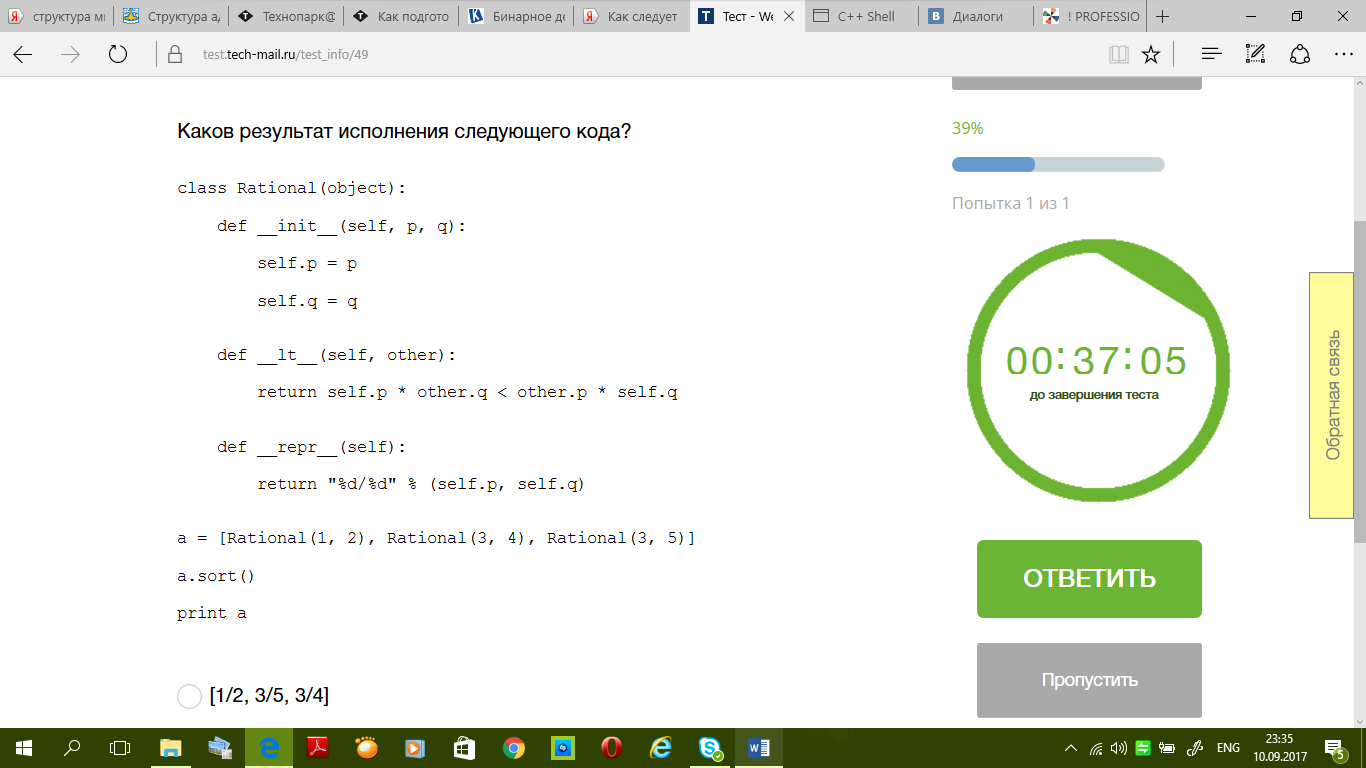
Правильный ответ: int

101 — это целочисленное значение, поэтому тип будет int.

Проверка:

a = 101

**print** type(a) *# <type 'int'>*

**Вопрос №22 из 53**

Правильный ответ: [1/2, 3/5, 3/4]

Класс Rational представляет собой рациональное число. Т.е. отношение целого числа p на q. Сначала создается список из 3 дробей: ½, ¾, 3/5. Потом вызывается функция sort над список, которая в свою очередь использует метод \_\_lt\_\_ у класса Rational для сравнения дробей. Метод \_\_repr\_\_ же отвечает за вывод дроби в строков виде при использовании функции **print**.

Проверка:

**class** Rational(object):

**def** \_\_init\_\_(self, p, q):

self.p = p

self.q = q

**def** \_\_lt\_\_(self, other):

**return** self.p \* other.q < other.p \* self.q

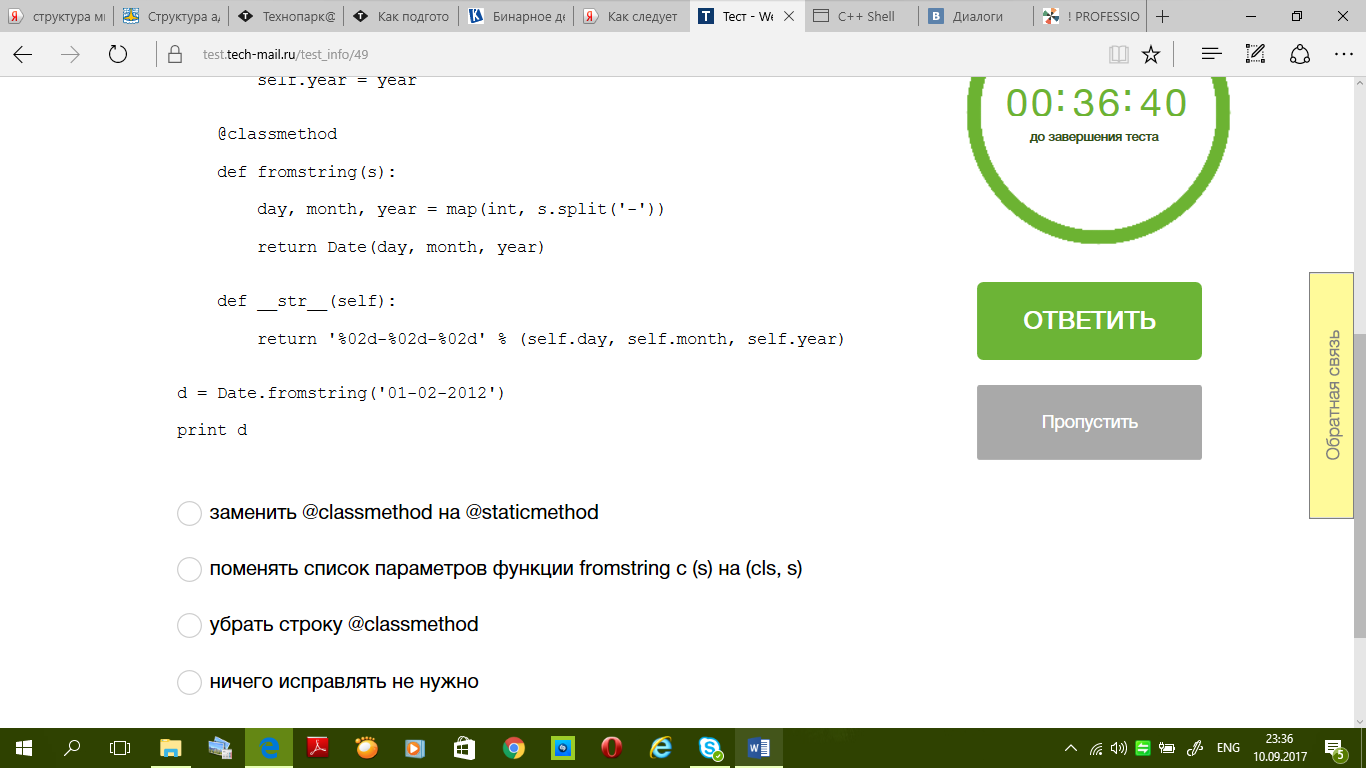
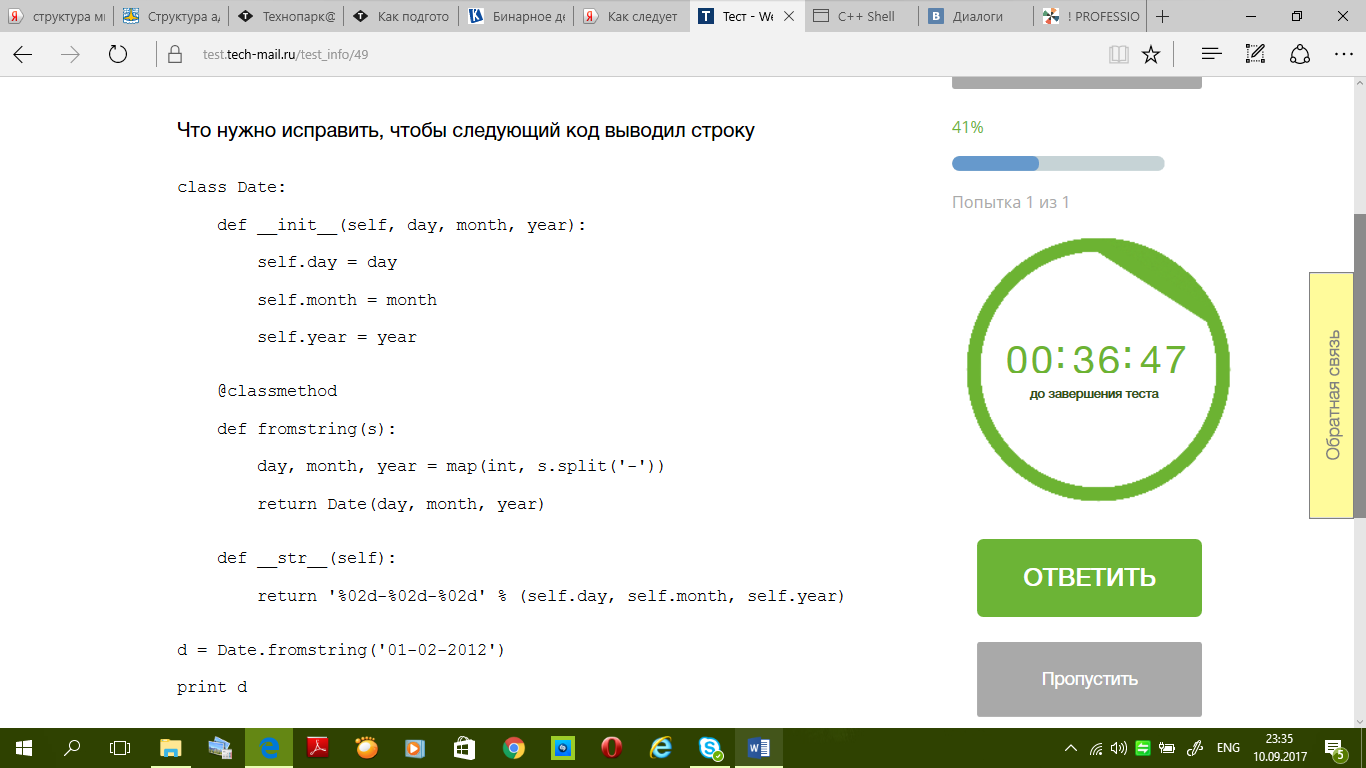
**def** \_\_repr\_\_(self):

**return "%d/%d"** % (self.p, self.q)

a = [Rational(1, 2), Rational(3, 4), Rational(3, 5)]

a.sort()

**print** a *# [1/2, 3/5, 3/4]*

**Вопрос №23 из 53**

Правильные ответы:

* либо **поменять список параметров функции fromstring с (s) на (cls, s).** Предпочтительнее выбрать именно этот вариант потому что в этом примере fromstring используется как как альтернативный конструктор для создания экземпляров класса Date по строке вида DD-MM-YYYY.
* либо **заменить @classmethod на @staticmethod;**

При использовании classmethod первый аргумент всегда будет ссылкой на класс, поэтому его нужно добавить в сигнатуру метода. Для staticmethod напротив не нужно ничего дополнительно передавать.

Хорошая информация на StackOverflow об использовании **@classmethod** и **@staticmethod**: <https://stackoverflow.com/questions/12179271/meaning-of-classmethod-and-staticmethod-for-beginner>

Проверка:

*# -\*- coding: utf-8 -\*-*

**class** Date:

**def** \_\_init\_\_(self, day, month, year):

self.day = day

self.month = month

self.year = year

@classmethod

**def** fromstring(cls, s):

day, month, year = map(int, s.split(**'-'**))

**return** Date(day, month, year)

*# @classmethod*

*# @staticmethod*

*# def fromstring(s):*

*# day, month, year = map(int, s.split('-'))*

*# return Date(day, month, year)*

**def** \_\_str\_\_(self):

**return '%02d-%02d-%02d'** % (self.day, self.month, self.year)

*# with @classmethod and cls*

d = Date.fromstring(**'01-02-2012'**)

**print** d *# 01-02-2012*

*# with @classmethod only*

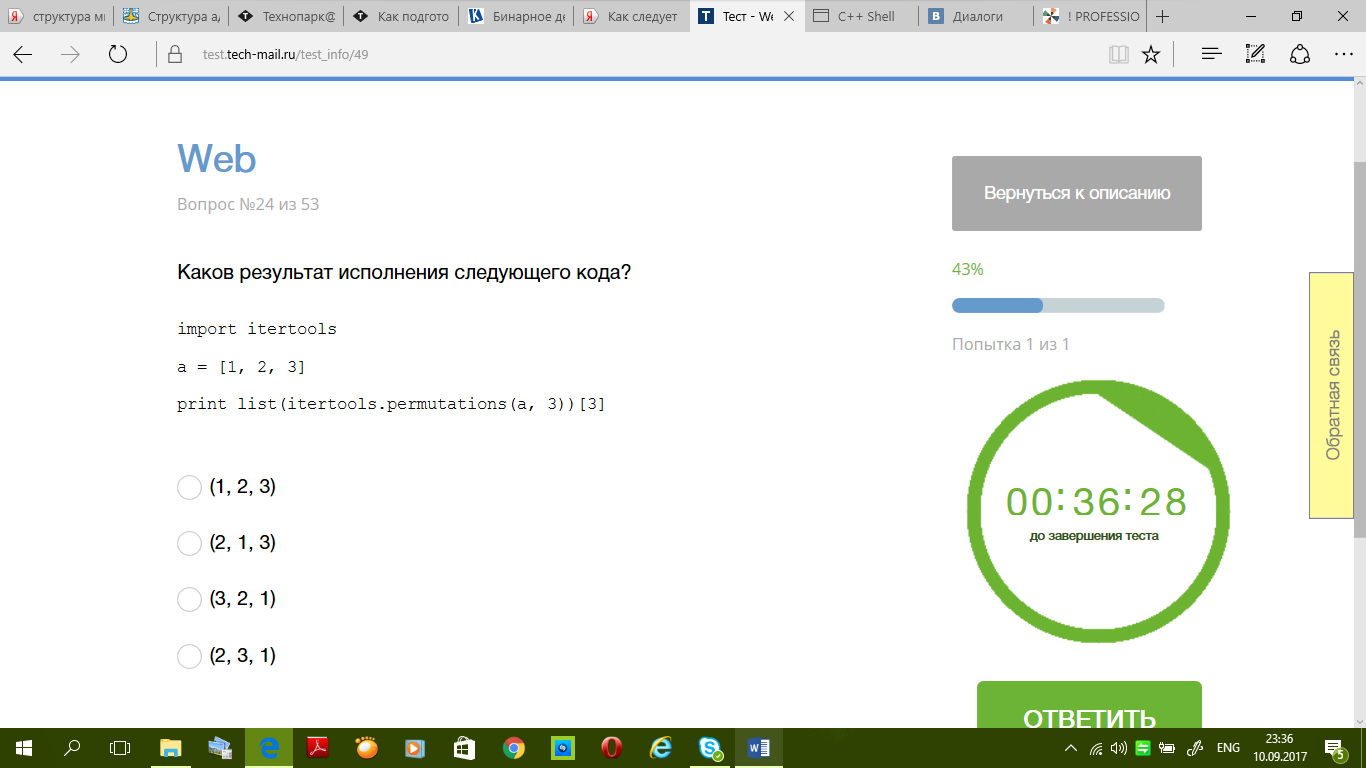
*# d = Date.fromstring('01-02-2012')*

*# print d # TypeError: fromstring() takes exactly 1 argument (2 given)*

*# with @staticmethod*

*# d = Date.fromstring('01-02-2012')*

*# print d # 01-02-2012*

**Вопрос №24 из 53**

Правильный ответ: **(2, 3, 1)**

Вызов функции itertools.permutations(a, 3) сгенерирует следующие перестановки:

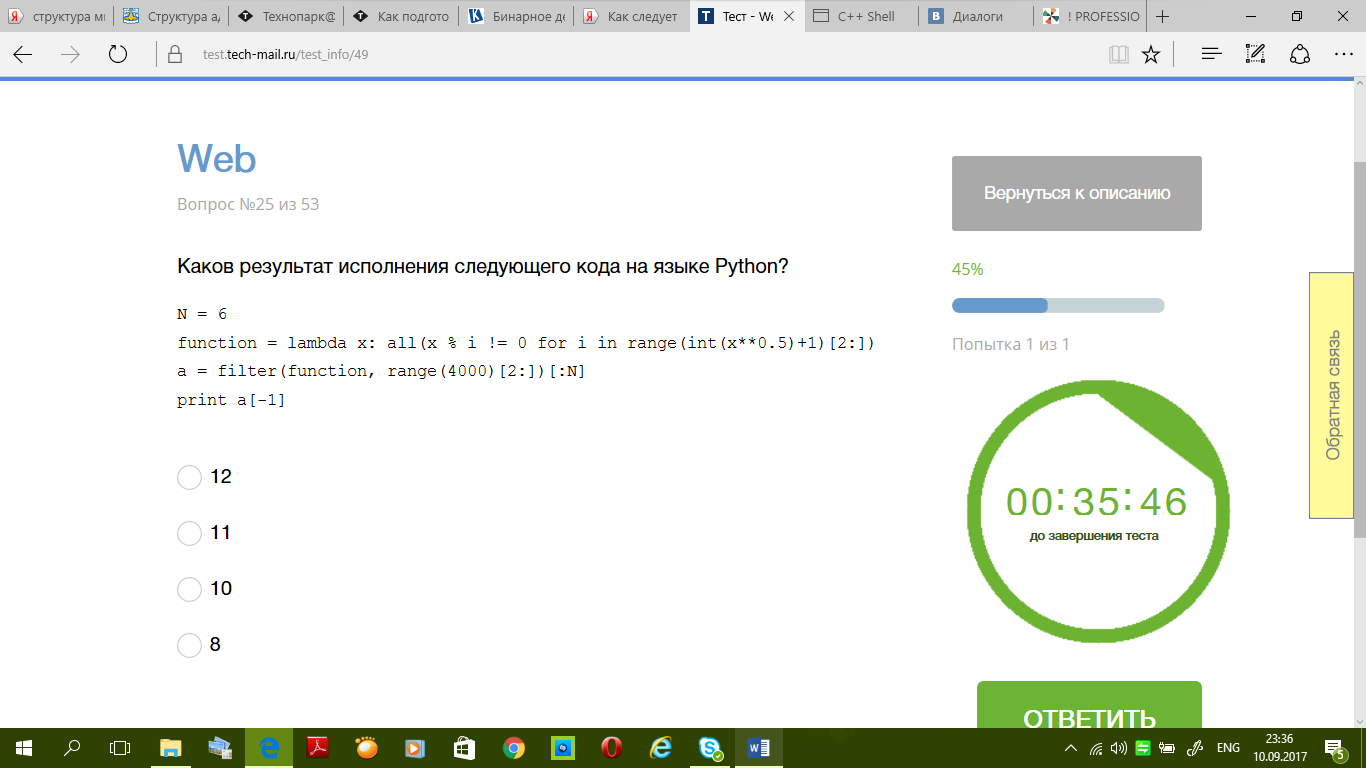
* *(1, 2, 3)*
* *(1, 3, 2)*
* *(2, 1, 3)*
* *(2, 3, 1)*
* *(3, 1, 2)*
* *(3, 2, 1)*

Если перевести их в список с помощью функции list, то 3 по индексу элементом как раз будет перестановка *(2, 3, 1)*.

Проверка:

*a = [1, 2, 3]*

**print** list(itertools.permutations(a, 3))[3] *# (2, 3, 1)*

**Вопрос №25 из 53**

На самом деле здесь правильный ответ 13, но в перечне вариантов такого нет.

В Python, кроме индексов, существуют ещё и срезы. item[START:STOP:STEP] - берёт срез от номера START, до STOP (не включая его), с шагом STEP. По умолчанию START = 0, STOP = длине объекта, STEP = 1.

Функция filter возвращает список элементов, для которых заданная функция возвращает True.

Встроенная функция all - идет по списку True/False (выражения которые сводятся к этим значениям) и прекращает проверку после первого элемента, не удовлетворяющего условию.

Проверка:

N = 6

function = **lambda** x: all(x % i != 0 **for** i **in** range(int(x\*\*0.5)+1)[2:])

a = filter(function, range(4000)[2:])[:N]

**print** a[-1] *# 13*

Чтобы было понятнее можно переписать код следующим образом:

*# print range(4000) # [0, 1, 2, ..., 3997, 3998, 3999]*

*# print range(4000)[2:] # [2, 3, 4, ..., 3997, 3998, 3999]*

result = []

**for** x **in** range(4000)[2:]:

*# print range(int(x\*\*0.5)+1);*

*# [0, 1] [0, 1]*

*# [0, 1, 2] [0, 1, 2] [0, 1, 2] [0, 1, 2] [0, 1, 2]*

*# [0, 1, 2, 3] [0, 1, 2, 3] [0, 1, 2, 3] [0, 1, 2, 3] [0, 1, 2, 3] [0, 1, 2, 3] [0, 1, 2, 3]*

*# print range(int(x \*\* 0.5) + 1)[2:]*

*# [] []*

*# [2] [2] [2] [2] [2]*

*# [2, 3] [2, 3] [2, 3] [2, 3] [2, 3] [2, 3] [2, 3]*

take = True

**for** i **in** range(int(x \*\* 0.5) + 1)[2:]:

**if** (x % i != 0) == False:

take = False

**break**

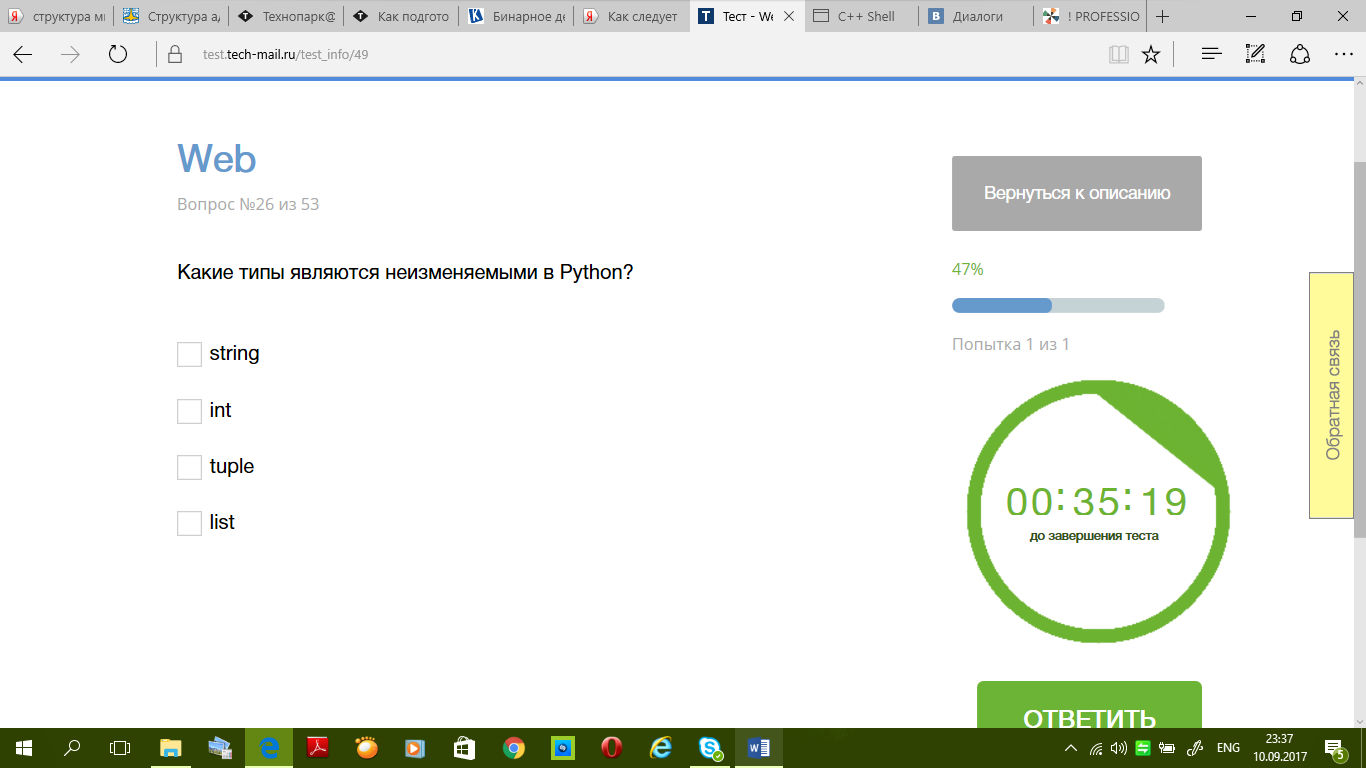
**if** take:

result.append(x)

*# print result # [2, 3, 5, ..., 3947, 3967, 3989]*

*# print result[:6] # [2, 3, 5, 7, 11, 13]*

**print** result[:6][-1] *# 13*

**Вопрос №26 из 53**

Правильный ответ: string, int, tuple. Т.е. все кроме ~~list~~.

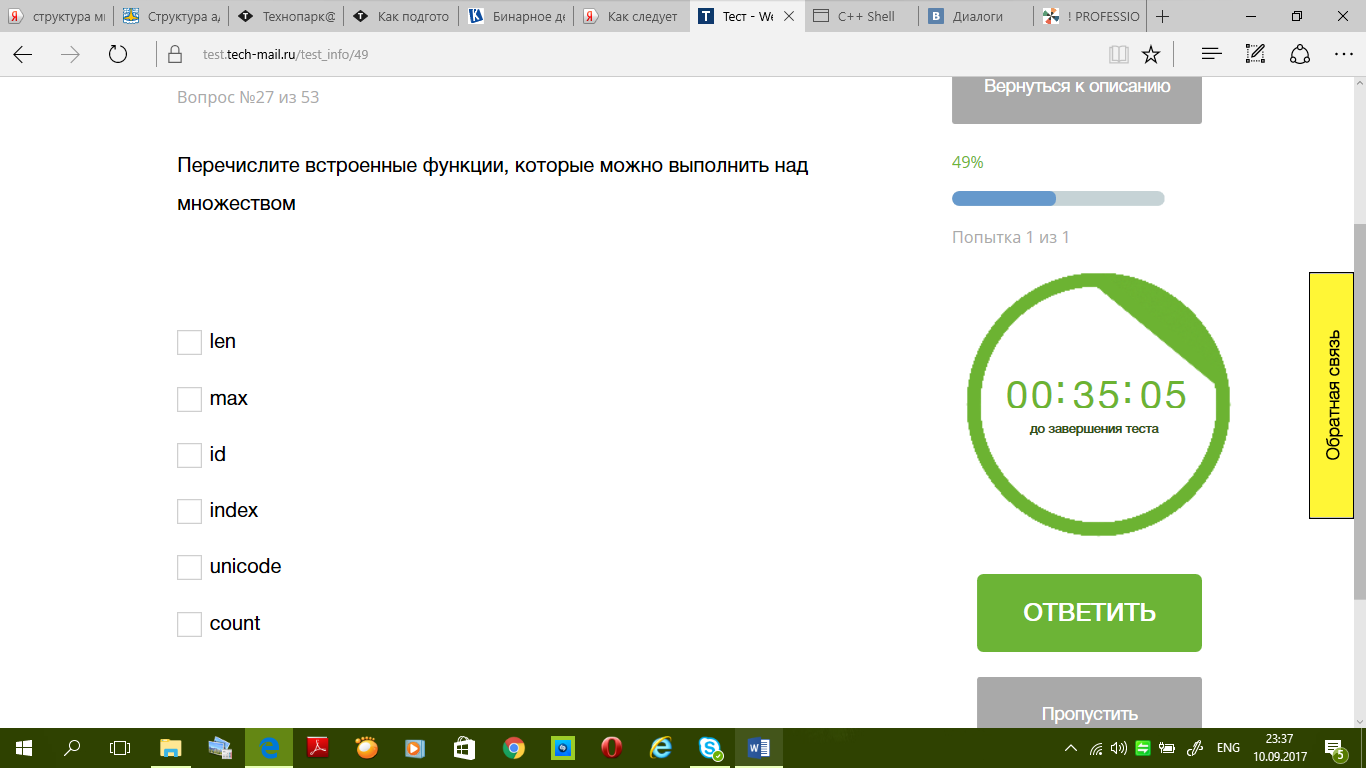
Неизменяемые объекты:

1. int
2. float
3. decimal
4. complex
5. bool
6. string
7. tuple
8. range
9. frozenset
10. bytes

Изменяемые объекты:

1. list
2. Dictionary
3. Set
4. bytearray
5. user defined classes

**Вопрос №27 из 53**



Правильные ответы:

len

max

id

unicode

Проверка:

s = {**'a'**, **'b'**, **'c'**, **'d'**}

**print** len(s) *# 4*

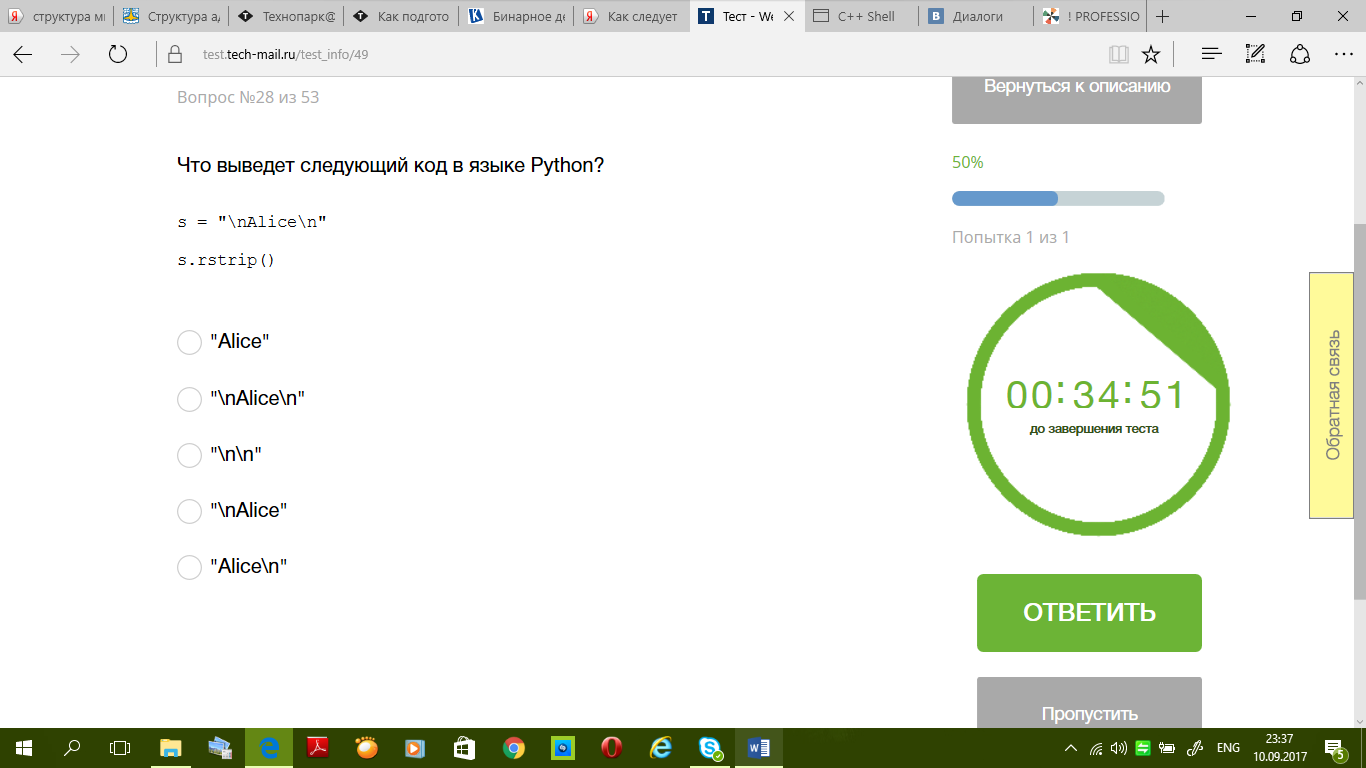
**print** max(s) *# d*

**print** id(s) *# 139639003450624*

*# print index(s) # NameError: name 'index' is not defined*

**print** unicode(s) *# set(['a', 'c', 'b', 'd'])*

*# print count(s) # name 'count' is not defined*

**Вопрос №28 из 53**

Правильный ответ: **"\nAlice\n"**

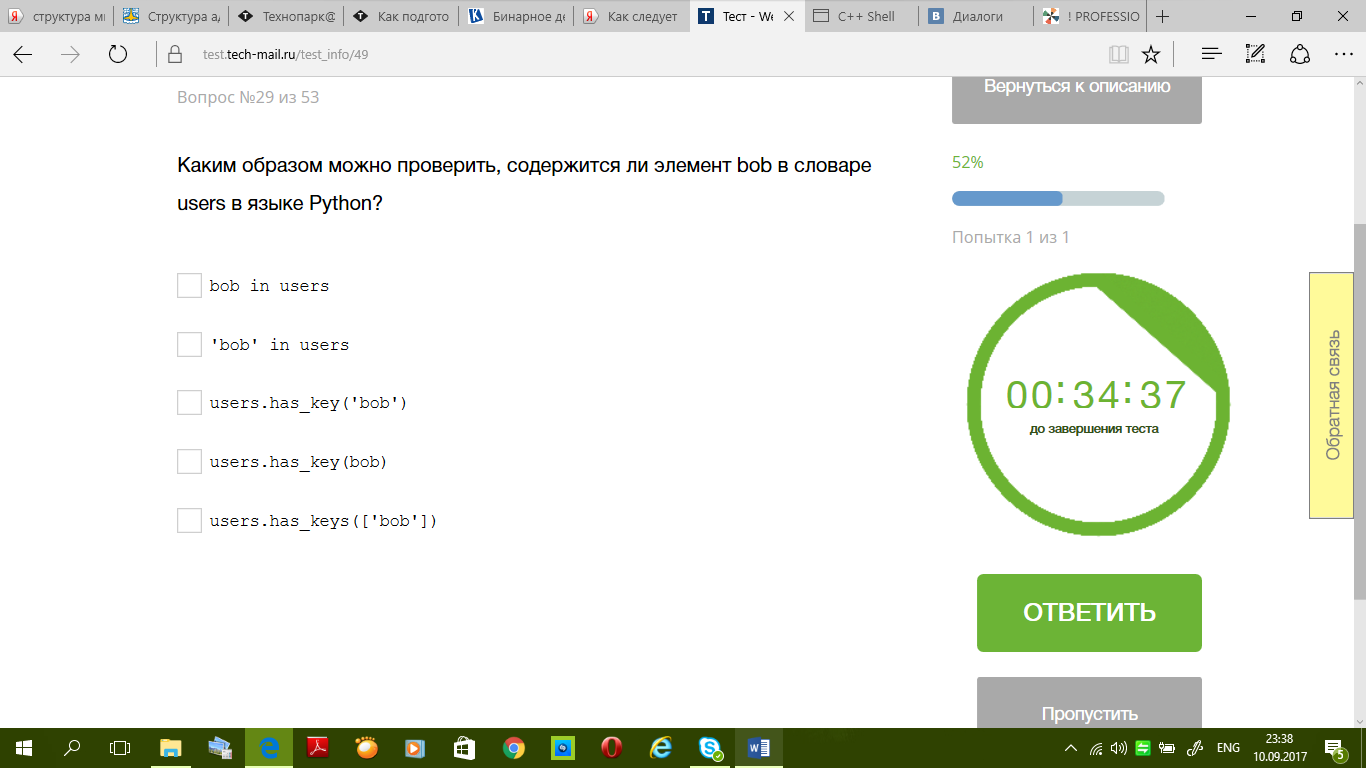
Данный код выведет **"\nAlice\n"** потому что функция rstrip() возвращает результат, который здесь ничему не присваивается. Т.е. переменная s не меняет своего значения. Если бы например был s = rstrip() то вывод был бы **"\nAlice"**.

Проверка:

s = **"\nAlice\n"**

s.rstrip()

**print** s *# \nAlice\n*

**Вопрос №29 из 53**

Правильный ответ: **'bob' in** users

Просто bob — это переменная, которой здесь нет. Нужно использовать строковое значение **'bob'**, которое представляет ключ в словаре. У словаря нет методов *has\_key* и *has\_keys*.

Проверка:

users = {**'bob'**}

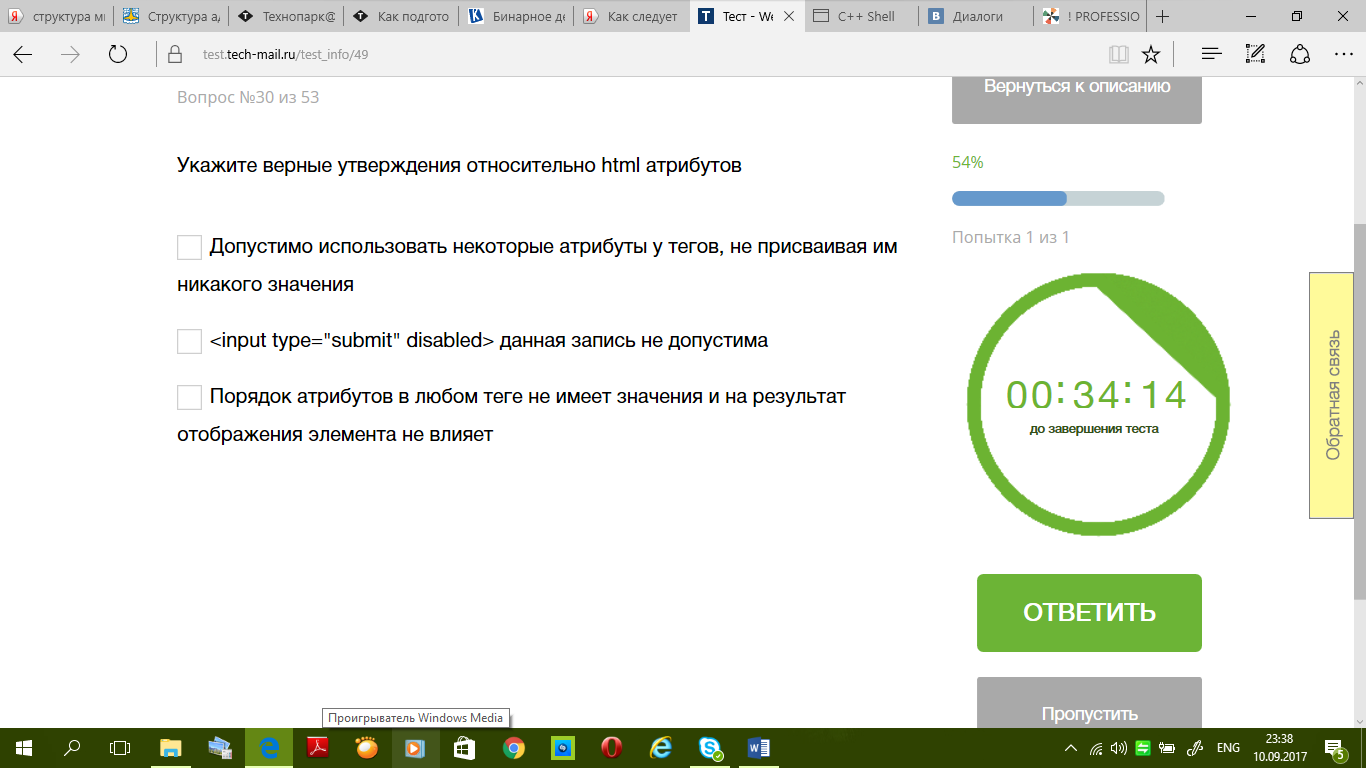
*# print bob in users # NameError: name 'bob' is not defined*

**print 'bob' in** users *# True*

*# print users.has\_key('bob') # AttributeError: 'set' object has no attribute 'has\_key'*

*# print users.has\_key(bob) # AttributeError: 'set' object has no attribute 'has\_key'*

*# print users.has\_keys('bob') # AttributeError: 'set' object has no attribute 'has\_key'*

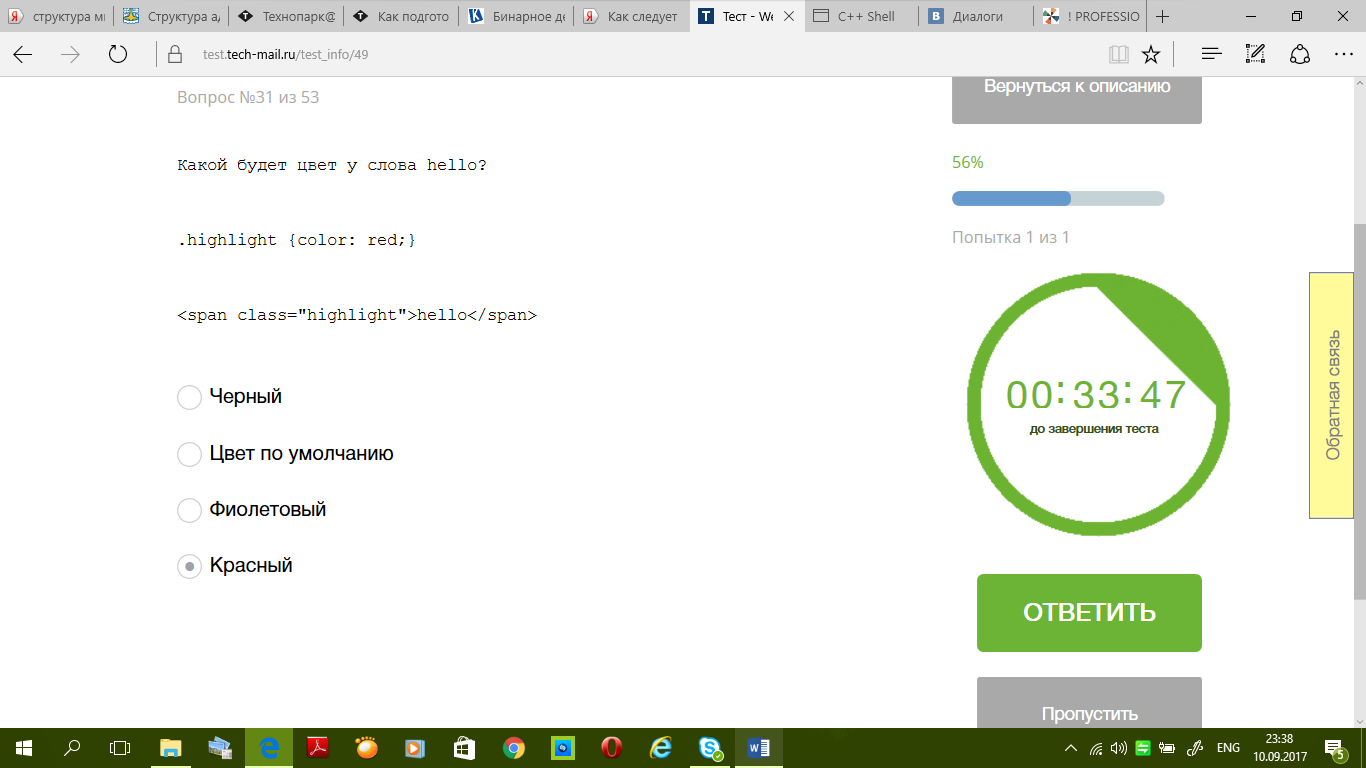
**Вопрос №30 из 53**

Правильные ответы:

* Допустимо использовать некоторые атрибуты у тегов, не присваивая им никакого значения.
* Порядок атрибутов в любом теге не имеет значения и на результат отображения элемента не влияет

Такая запись допустима:

**<p><input**type="submit"name="submit"value="Отправить"disabled**></p>**

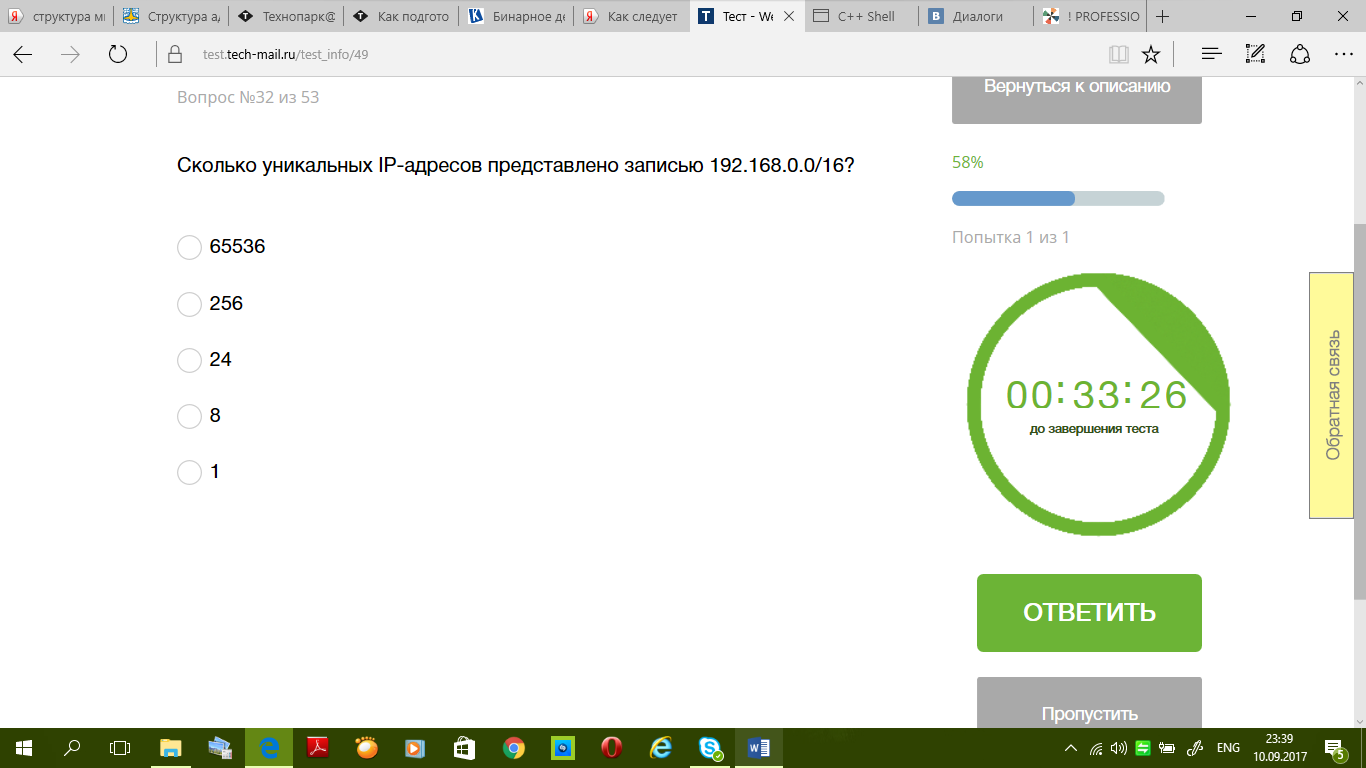
**Вопрос №31 из 53**

Правильный ответ: Красный

В CSS записи **.наименование\_класса** означает применение свойств ко всем тегам принадлежащим к этому классу. По сути это селектор.

Пример:

<https://codepen.io/anon/pen/aqpVdO>

**Вопрос №32 из 53**

Правильный ответ: 65536

/x это просто "сколько бит используется в маске" всего в IPv4 - 32 bit, поэтому, например, маска /16 указывает на первые 16 бит.

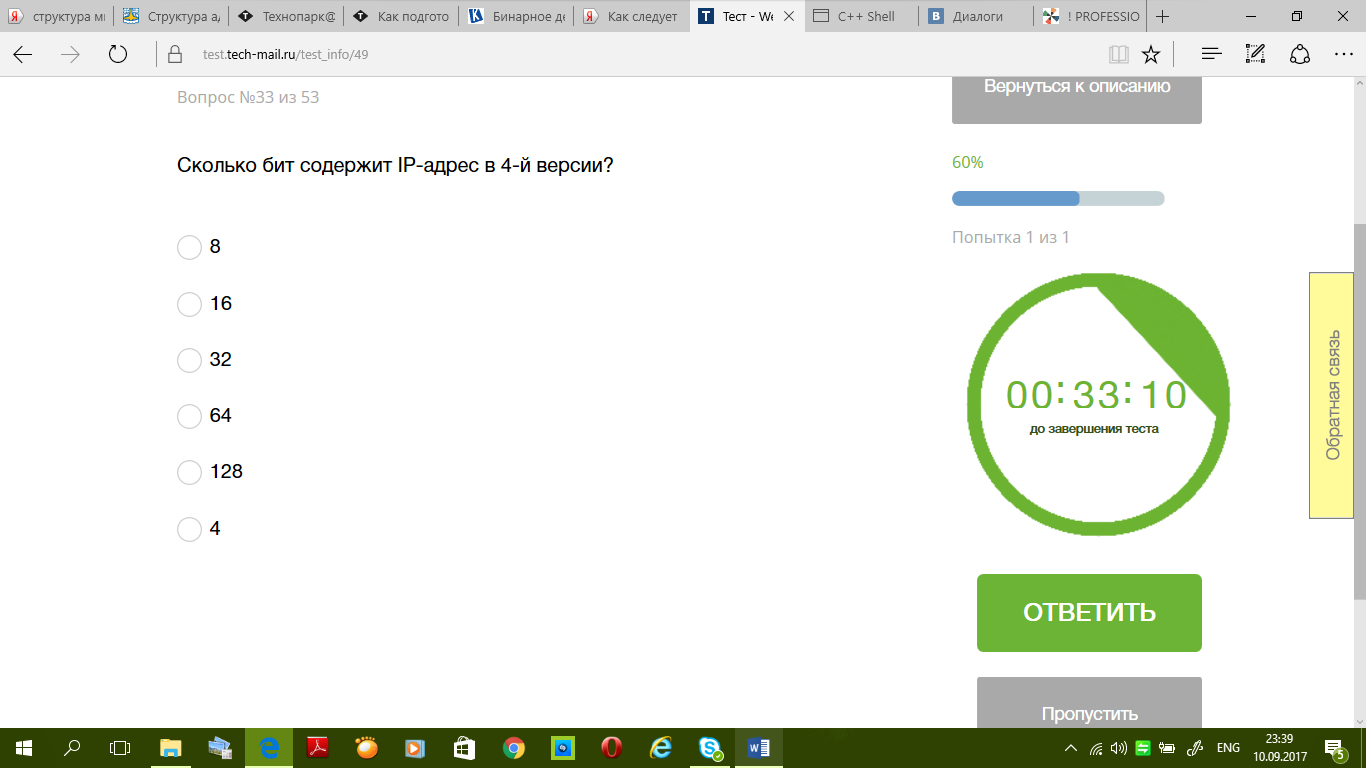
Данный вид записи заменяет собой указание диапазона IP-адресов. Число после косой черты означает количество единичных разрядов в маске подсети. Для приведённого примера маска подсети будет иметь двоичный вид

11111111.11111111.00000000.00000000

или то же самое в десятичном виде: 255.255.0.0

16 разрядов IP-адреса отводятся под номер сети, а остальные 32 − 16 = 16 разрядов полного адреса — под локальный адрес в этой сети.

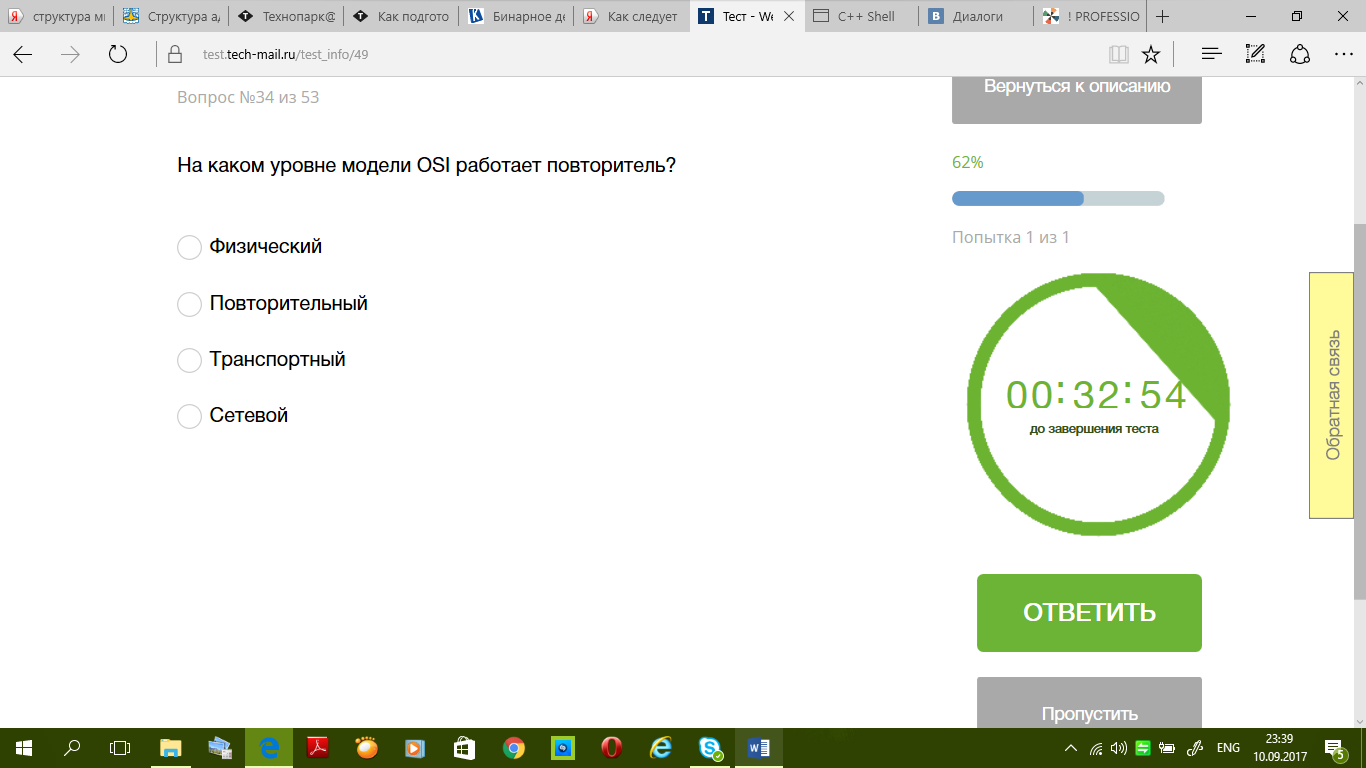
Количество уникальных IP-адресов подсети определяется как 232-N, где N — длина маски. Чем длиннее маска, тем меньше в ней уникальных IP-адресов.

**Вопрос №33 из 53**

Правильный ответ: 32

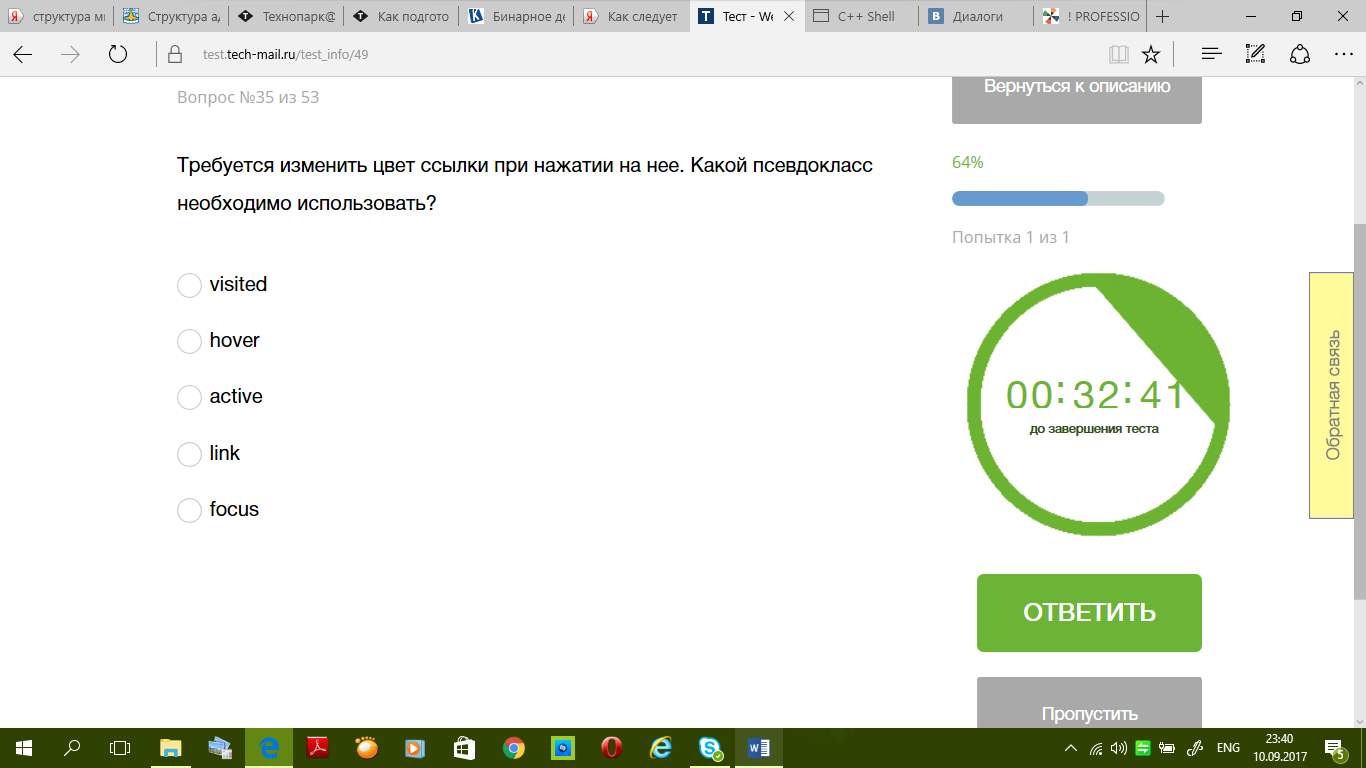
В одном байте 8 бит. Соответственно в 4 байтах 4\*8=32 бита.

В версии протокола IPv4 IP-адрес имеет длину 4 байта, а в версии протокола IPv6 IP-адрес имеет длину 16 байт.

**Вопрос №34 из 53**

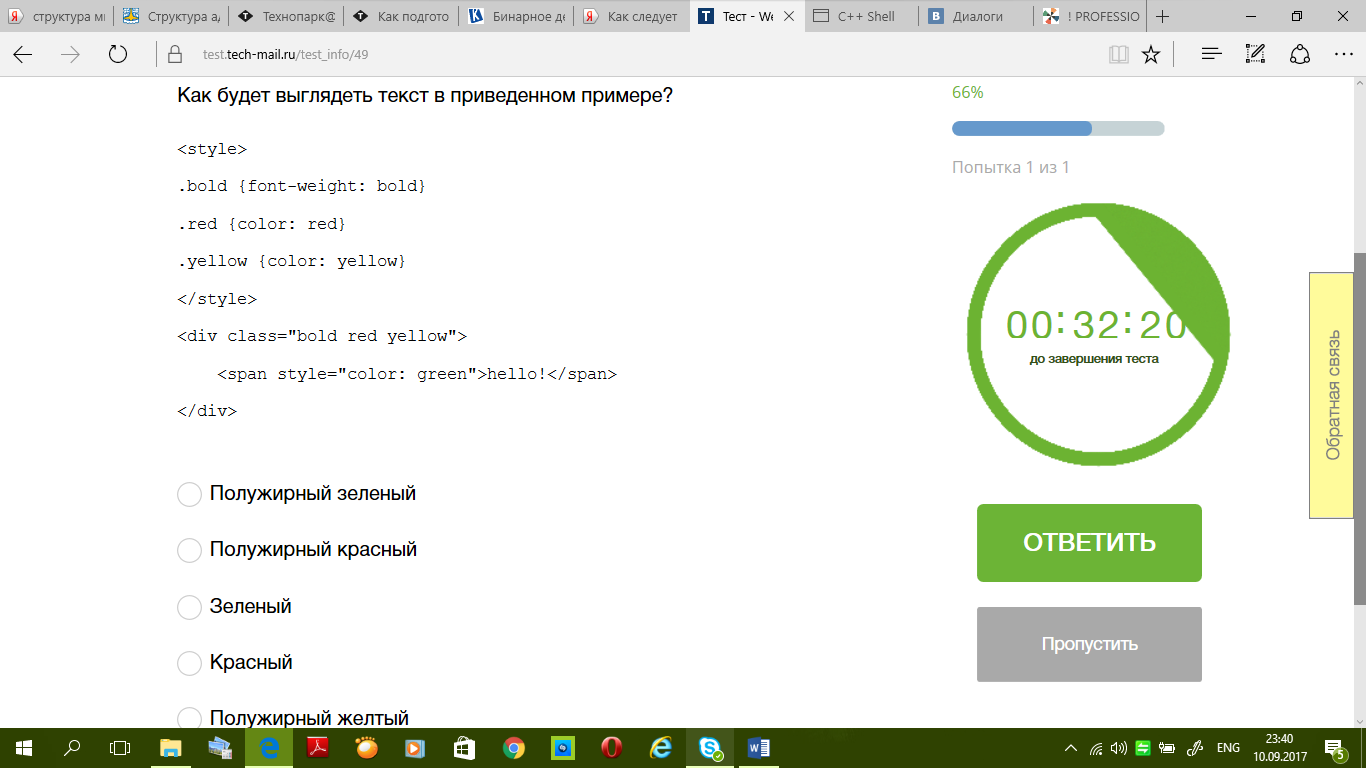
Правильный ответ: Физический

Повторитель (репи́тер, от англ. repeater) — сетевое оборудование, предназначенное для увеличения расстояния сетевого соединения и его расширения за пределы одного сегмента или для организации двух ветвей, путём повторения электрического сигнала «один в один». Бывают однопортовые повторители и многопортовые. В терминах модели OSI работает на физическом уровне.

**Вопрос №35 из 53**

Правильный ответ: active

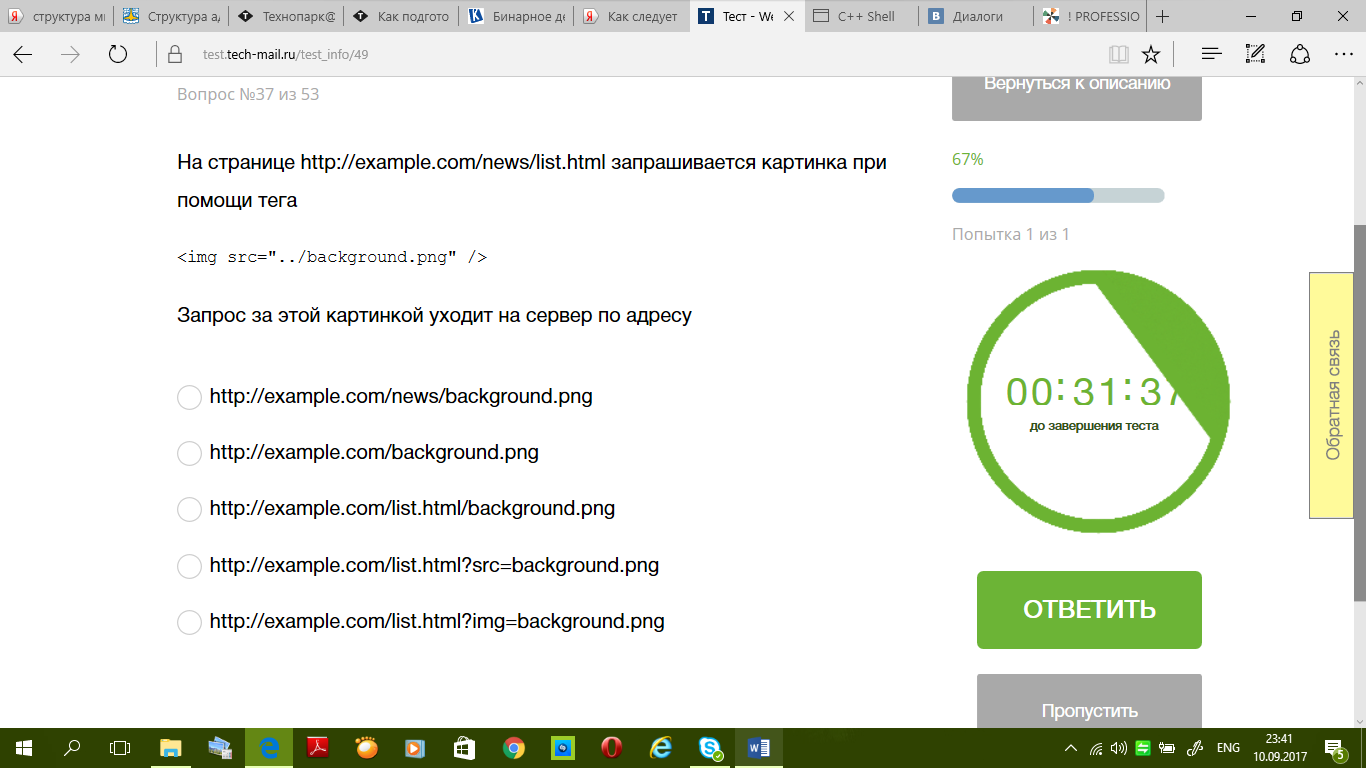
* Псевдокласс :visited применяется к ссылкам, уже посещённым пользователем, и задаёт для них стилевое оформление.
* hover Определяет стиль элемента при наведении на него курсора мыши, но при этом элемент еще не активирован, иными словами кнопка мыши не нажата.
* Псевдокласс :active определяет стиль для активной ссылки. Активной ссылка становится при нажатии на нее.
* Псевдокласс :link применяется к ссылкам, которые еще не посещались пользователем, и задает для них стилевое оформление.
* Псевдокласс :focus определяет стиль для элемента получающего фокус. Например, им может быть текстовое поле формы, в которое устанавливается курсор.

**Вопрос №36 из 53**

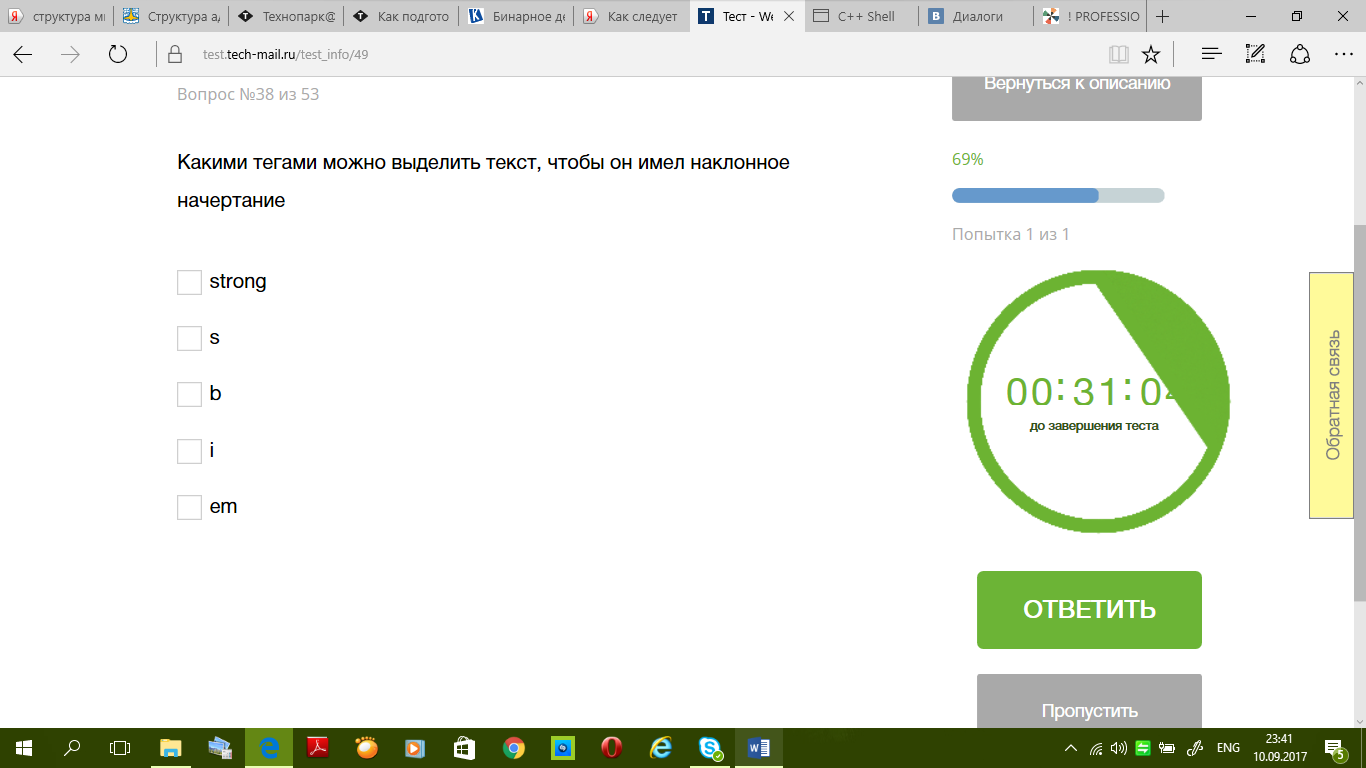
Правильный ответ: Полужирный зеленый

Потому что <span style="color: green"> перекрывает стиль yellow, который в свою очередь перекрывает стиль red.

Пример: <https://codepen.io/anon/pen/EQZbQK>

**Вопрос №37 из 53**

Так как файл list.html находится в папке news, а картинка запрашивается из папки по уровню выше ../backgorund.png, то запрос будет уходить по адресу http://example.com/background.png.

**Вопрос №38 из 53**

Правильный ответ: i и em

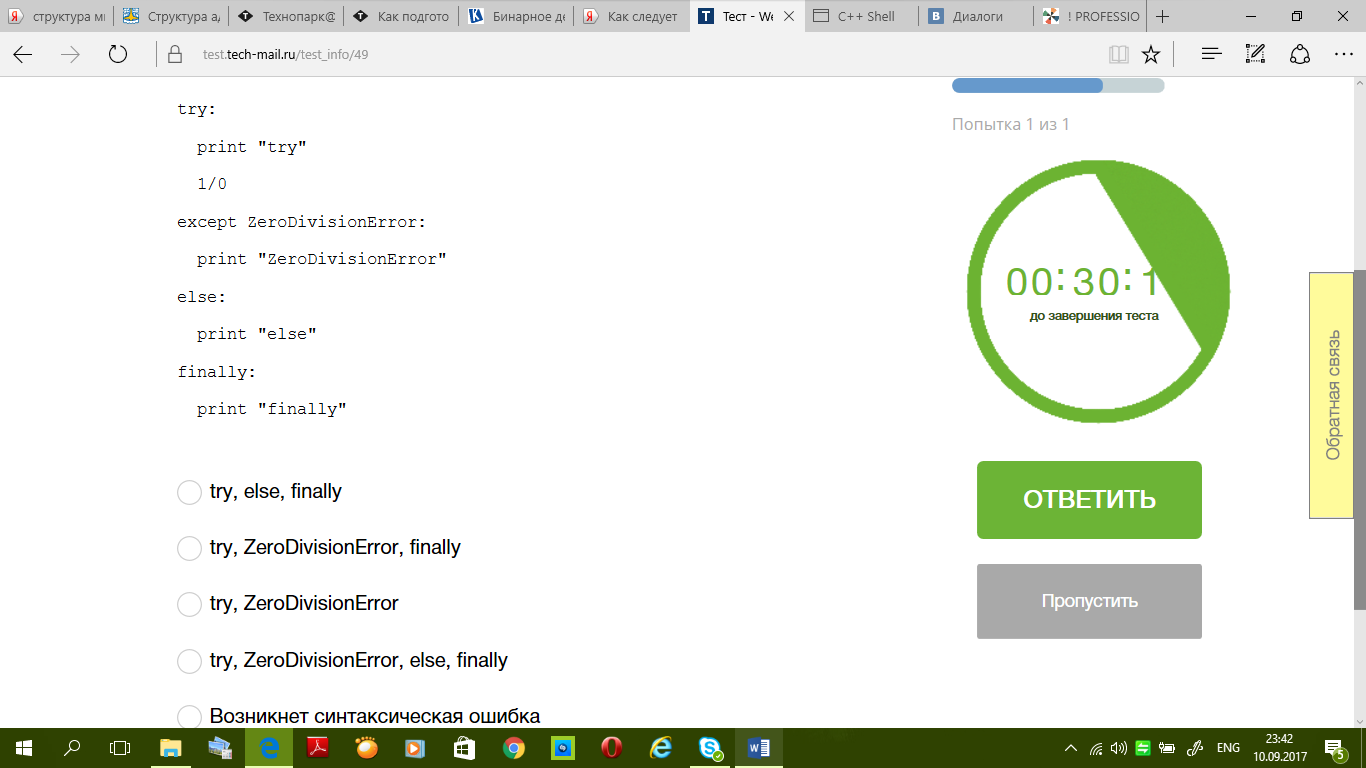
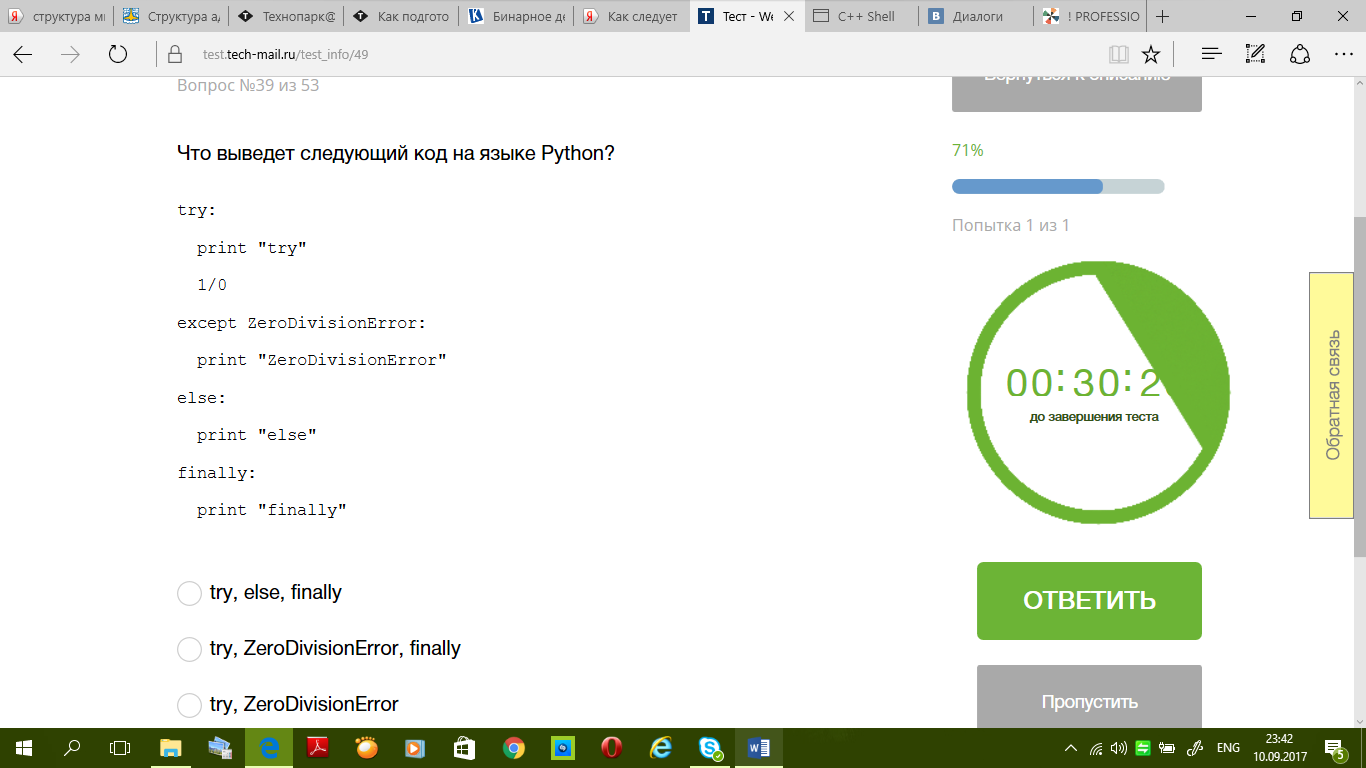
Теги b и strong используются для полужирного начертания.

Теги i и em используются для наклонного начертания.

Тег s — это перечеркнутый текст.

Пример:

<https://codepen.io/anon/pen/rJjYoe>

**Вопрос №39 из 53**

Данный код выдает: try, ZeroDivisionError, finally

**try** и **finally** исполняются всегда. **except** выполняется при исключении, в данном случае при делении 0. **else** выполняется, когда нет исключений. Но в данном случае оно есть из-за деления на 0.

Проверка:

**try**:

**print "try"**

1/0

**except** ZeroDivisionError:

**print "ZeroDivisionError"**

**else**:

**print "else"**

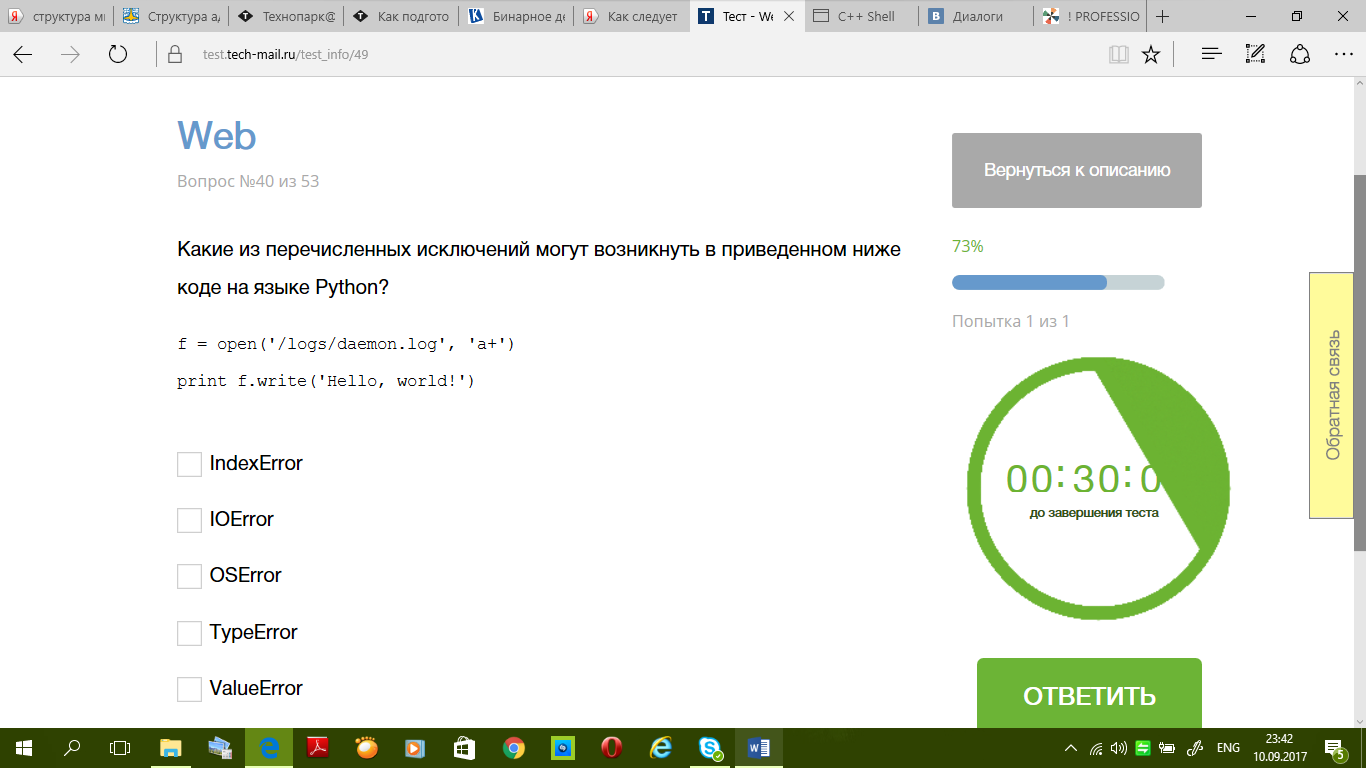
**finally**:

**print "finally"**

*# try*

*# ZeroDivisionError*

*# finally*

**Вопрос №40 из 53**

Правильные ответы:

**IOError** если нет папки /logs например.

**OSError** в принципе может быть потому что в Python 3.3 исключения EnvironmentError, **IOError**, WindowsError, VMSError, socket.error, select.error and mmap.error были объеденины в один **OSError** (см. [https://docs.python.org/3.3/library/exceptions.html#IOError](https://docs.python.org/3.3/library/exceptions.html" \l "IOError)).

Но для Python 2 можно ограничиться одним лишь **IOError** (см. [https://docs.python.org/2/library/functions.html#open](https://docs.python.org/2/library/functions.html" \l "open)).

Неправильные ответы:

**IndexError** не может быть потому что здесь нет нигде выхода за границы списков как при слайсинге например.

**TypeError** не может быть потому что это исключение возникает при неподходящем типе данных.

**ValueError** не может быть потому что тут нет конвертации данных например. Это исключение возникает например когда тип данных правильный, но значение самих данных не подходящее.

**Вопрос №41 из 53**

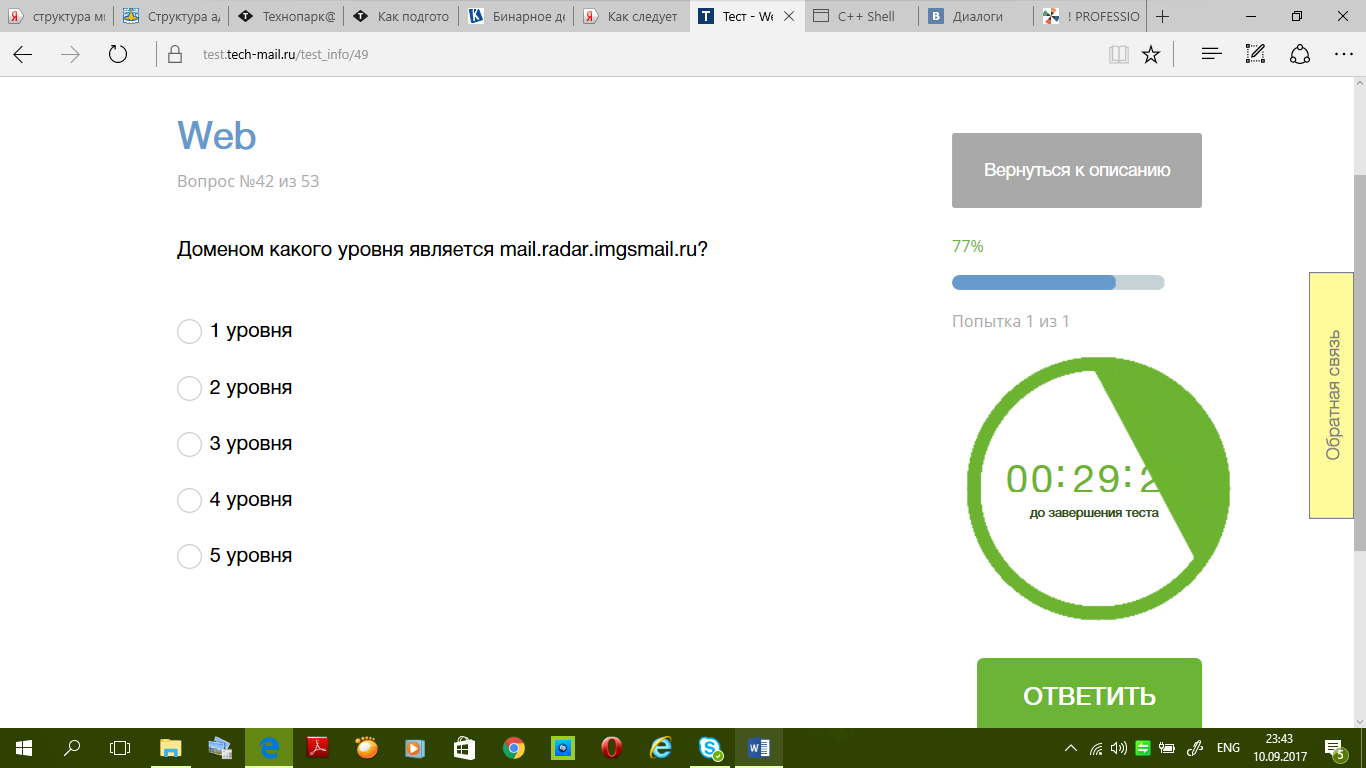
Правильный ответ:

Значение cookie передается с помощью заголовка Cookie

Ку́ки (англ. cookie, буквально — печенье) — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя.

Пример:

|  |  |  |
| --- | --- | --- |
| **браузер** → **сервер** | **браузер** ← **сервер** | **браузер** → **сервер** |
| GET /index.html HTTP/1.1 Host: www.example.org | HTTP/1.1 200 OK Content-type: text/html **Set-Cookie: name=value**  (содержимое страницы) | GET /spec.html HTTP/1.1 Host: www.example.org **Cookie: name=value** Accept: \*/\* |

**Вопрос №42 из 53**

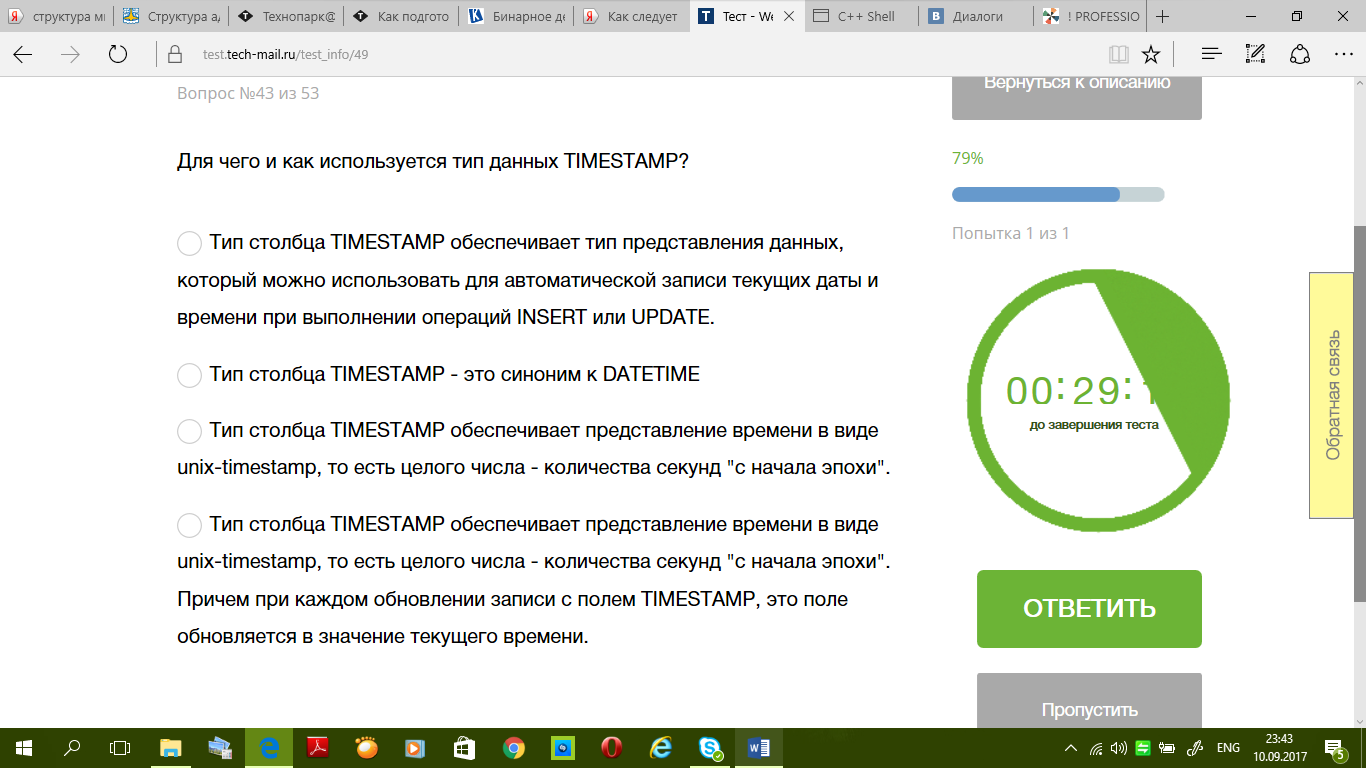
Правильный ответ: 4 уровня

.ru – указывает на принадлежность домена России.

Домен imgsmail.ru является доменом второго уровня.

Домен radar.imgsmail.ru является доменом 3 уровня.

Домен mail.radar.imgsmail.ru является доменом 4 уровня.

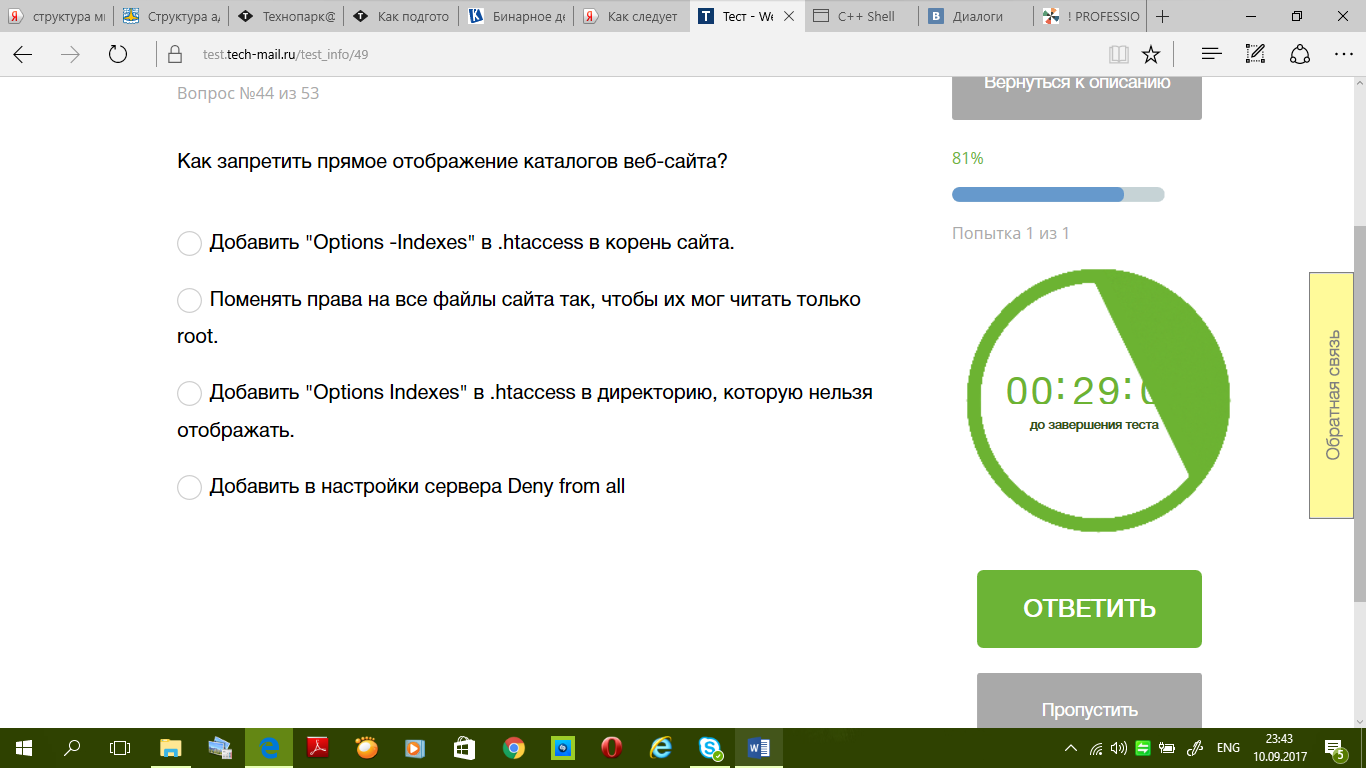
**Вопрос №43 из 53**

Верный ответ: Тип столбца TIMESTAMP обеспечивает представление времени в виде unix-timestamp, то есть целого числа — количества секунд «с начала эпохи».

TIMESTAMP — это **НЕ СИНОНИМ** DATETIME!

DATETIME Хранит время в виде целого числа вида YYYYMMDDHHMMSS, используя для этого 8 байтов. Это время не зависит от временной зоны. Оно всегда отображается при выборке точно так же, как было сохранено, независимо от того какой часовой пояс установлен в MySQL.

TIMESTAMP Хранит 4-байтное целое число, равное количеству секунд, прошедших с полуночи 1 января 1970 года по усреднённому времени Гринвича (т.е. нулевой часовой пояс, точка отсчёта часовых поясов). При получении из базы отображается с учётом часового пояса. Часовой пояс может быть задан в операционной системе, глобальных настройках MySQL или в конкретной сессии. Запомните, что сохраняется всегда количество секунд по UTC (универсальное координированное время, солнечное время на меридиане Гринвича), а не по локальному часовому поясу.

**Вопрос №44 из 53**

Правильный ответ: Добавить "Options -Indexes" в .htaccess в корень сайта.

Индексный файл или файл-индекс — это тот файл, который открывается по умолчанию при обращении пользователя через веб к каталогу, а не к конкретному файлу. По умолчанию индексными файлами являются следующие: index.html, index.htm, index.php, index.php3, index.phtml, index.shtml, default.htm или default.html.

По умолчанию включена директива Options -Indexes, и в случае отсутствия индексной страницы вы получите HTTP ошибку 403.

В ряде случаев требуется выводить список файлов в каталоге (листинг каталога) в случае отсутствия в каталоге файла, который показывается по умолчанию. Для этого необходимо добавить в .htaccess следующую строку: Options +Indexes

Файл .htaccess необходимо создавать именно в том каталоге, в котором планируется разрешить листинг. Данная директива будет действовать также и на все подкаталоги (это достигается включенной по умолчанию в настройках виртуального хоста директивой AllowOverride All).

Директивы Allow и Deny определяют клиентов, которым разрешить или запретить доступ к серверу. Директивы допускают использование:

* Одиночного IP(IP) - обычный вид IP, например, 127.0.0.1
* Группы IP(IPs) - группа IP, например, для доступа, только из локальной сети, 192.168.1.0/24
* Любого IP(all) - обозначает любой IP

**Примеры**

Файл .htaccess

Order allow,deny

# Deny from all # если вы это напишите, то даже те адреса,

# которые указаны в директивах Allow не будет пропущены

Allow from 192.168.1.0/24 11.11.11.12

В этом файле указывается доступ только для клиентов из локальной сети или с IP 11.11.11.12

**Часть файла httpd.conf**

...

<Directory "/home/Site.ru">

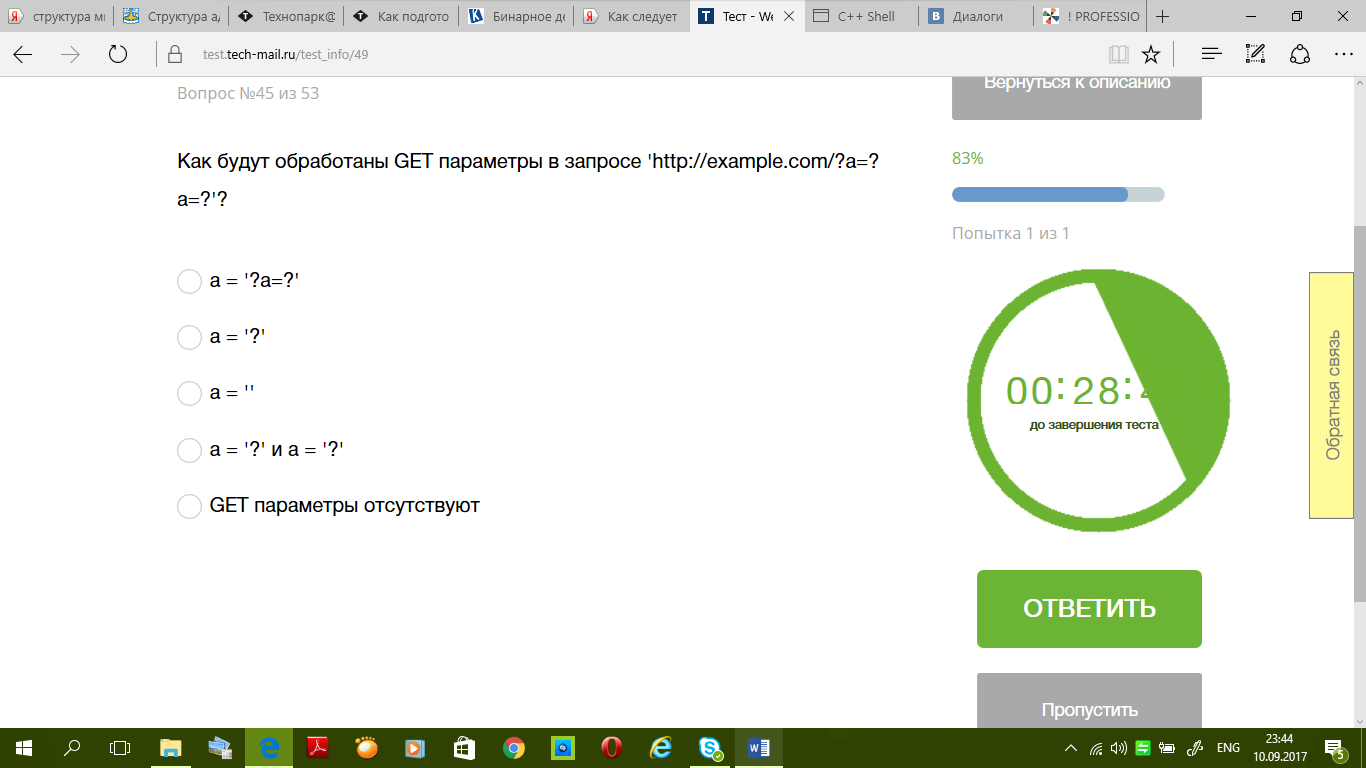
Order deny,allow

Deny from 33.250.11.25

</Diectory>

...

Так мы баним сайт для какого-нибудь одного IP

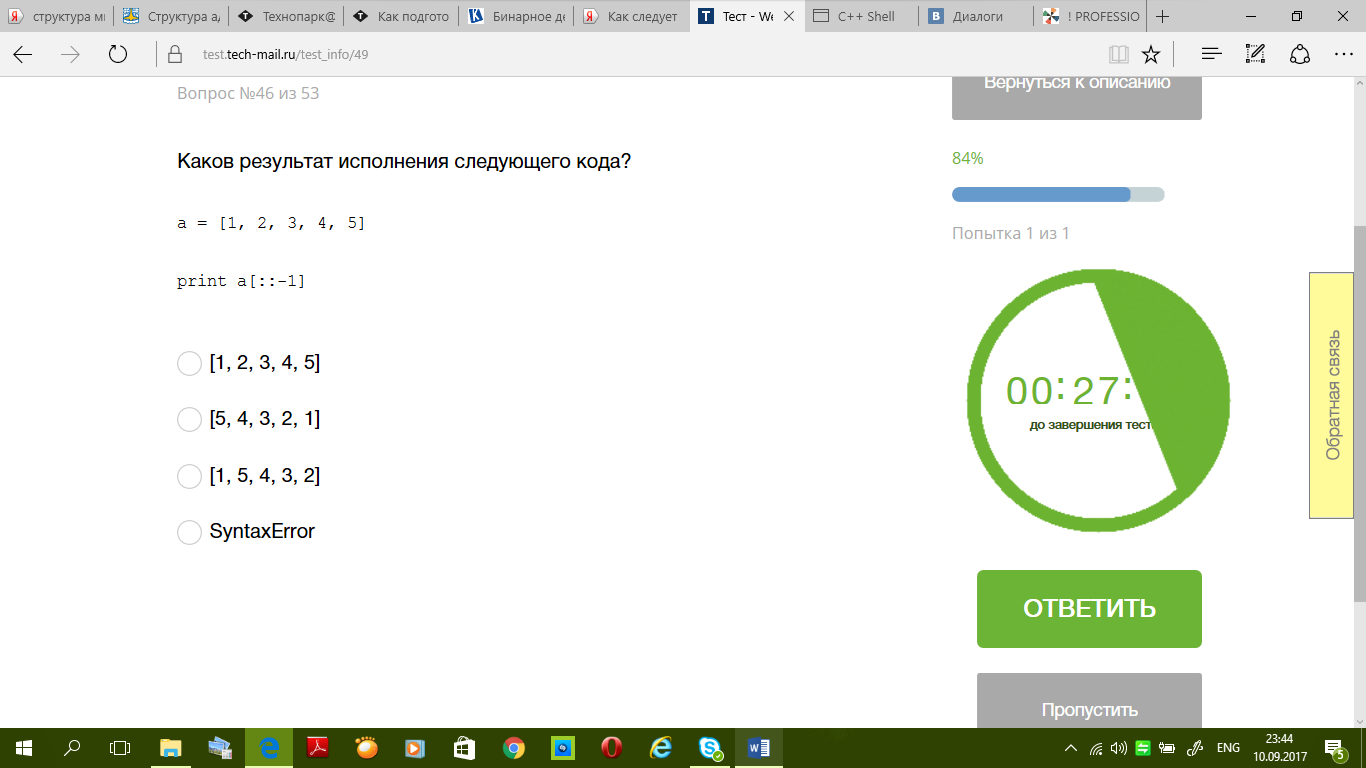
**Вопрос №45 из 53**

Правильный ответ: a='?a=?'

Согласно стандарту RFC 3986 компоненты HTTP-запроса начинаются после первого вопросительного знака ? и заканчиваются перед знаком # или концом URI.

Стандарт:

[https://tools.ietf.org/html/rfc3986#section-3.4](https://tools.ietf.org/html/rfc3986" \l "section-3.4)

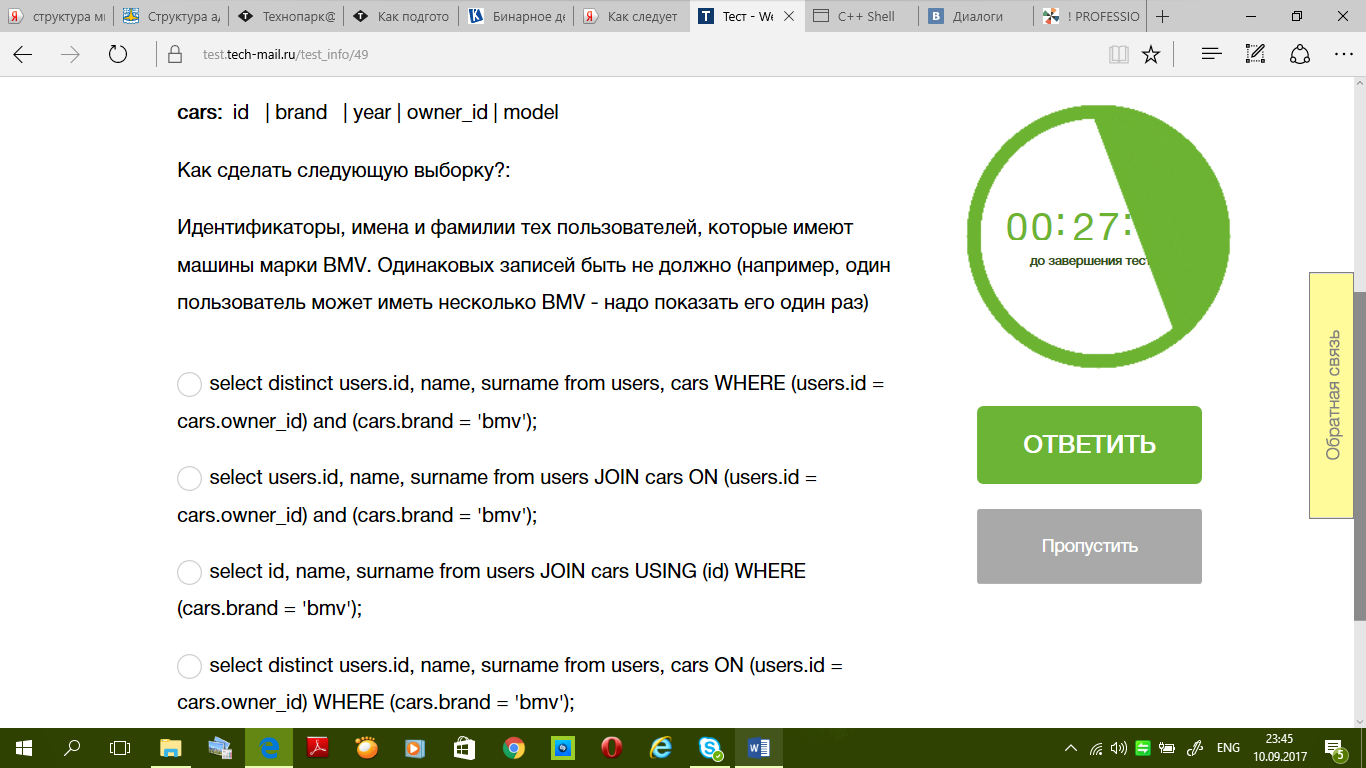
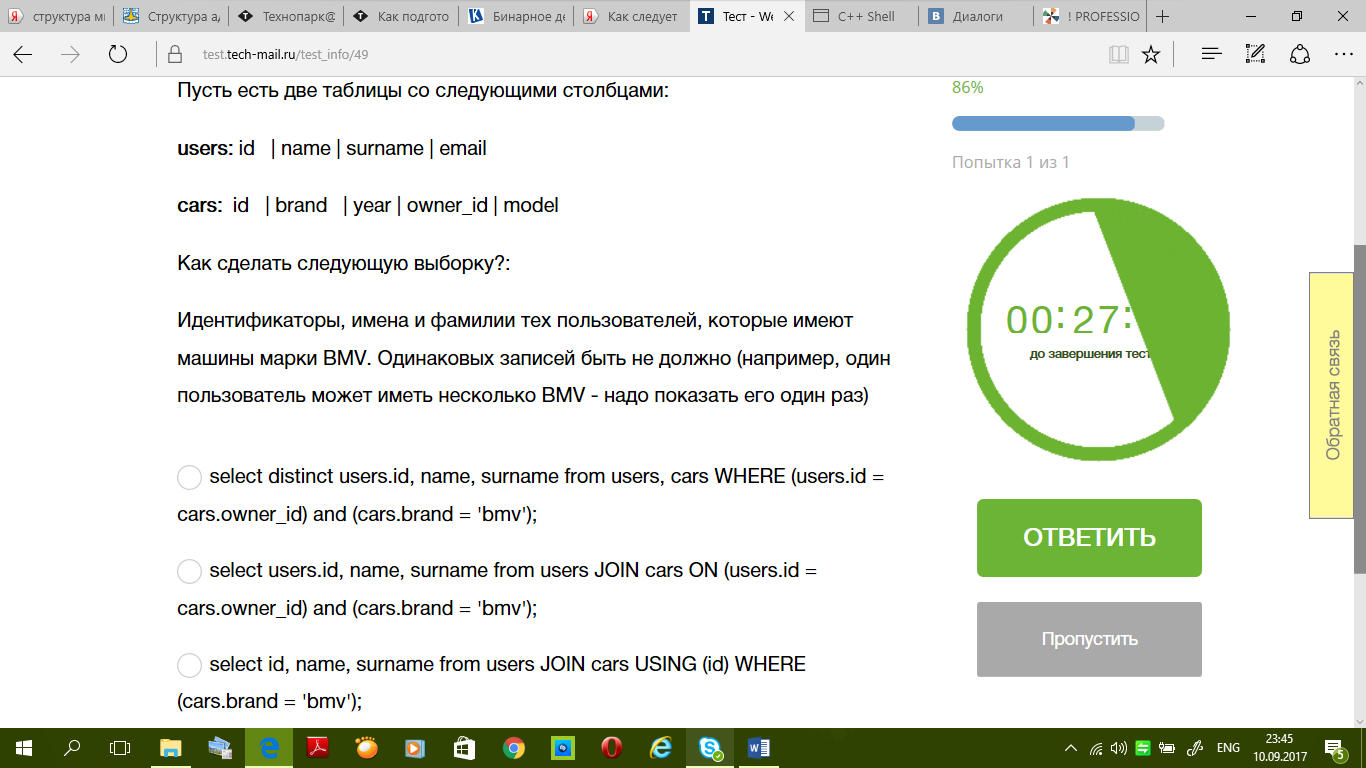
**Вопрос №46 из 53**

Правильный ответ: *[5, 4, 3, 2, 1]*

Синтаксис слайсинга:

my\_collection[start:stop:step] # старт, стоп и шаг

Отрицательное значение шага — перебор ведём в обратном порядке справа налево.

**Вопрос №47 из 53**

Верный ответ первый:

select distinct users.id, name, surname from users, cars

WHERE (users.id = cars.owner\_id) and (cars.brand='bmv')

Другие ответы:

**Покажет одинаковые записи:**

select users.id, name, surname from users JOIN cars ON

(users.id=cars.owner\_id) and (cars.brand='bmv')

**USING соединяет таблицы по заданному полю. Тут соединяются по полю id, а должны по полям id и owner\_id:**

select id, name, surname from users JOIN cars

USING (id) WHERE (cars.brand = 'bmv')

**Синтаксически не верная конструкция:**

select distinct usrs.id, name, surname from users, cars ON

(users.id=card.owner\_id) WHERE (cars.brand='bmv')

Пример: [http://sqlfiddle.com/#!9/5b6fc1/4](http://sqlfiddle.com/" \l "!9/5b6fc1/4)

CREATE TABLE IF NOT EXISTS `users` (

`id` int(6) unsigned NOT NULL,

`name` varchar(200) NOT NULL,

`surname` varchar(200) NOT NULL,

`email` varchar(200) NOT NULL,

PRIMARY KEY (`id`)

) DEFAULT CHARSET=utf8;

INSERT INTO `users` (`id`, `name`, `surname`, `email`) VALUES

('1', 'John', 'Smith', 'john.smith@gmail.com');

CREATE TABLE IF NOT EXISTS `cars` (

`id` int(6) unsigned NOT NULL,

`brand` varchar(200) NOT NULL,

`year` int(6) unsigned NOT NULL,

`owner\_id` int(6) unsigned NOT NULL,

`model` varchar(200) NOT NULL,

PRIMARY KEY (`id`)

) DEFAULT CHARSET=utf8;

INSERT INTO `cars` (`id`, `brand`, `year`, `owner\_id`, `model`) VALUES

('2', 'BMV', '2000', '1', 'X5');

INSERT INTO `cars` (`id`, `brand`, `year`, `owner\_id`, `model`) VALUES

('3', 'BMV', '2010', '1', 'X6');

select distinct users.id, name, surname from users, cars

WHERE (users.id = cars.owner\_id) and (cars.brand='bmv')

-- select users.id, name, surname from users JOIN cars ON

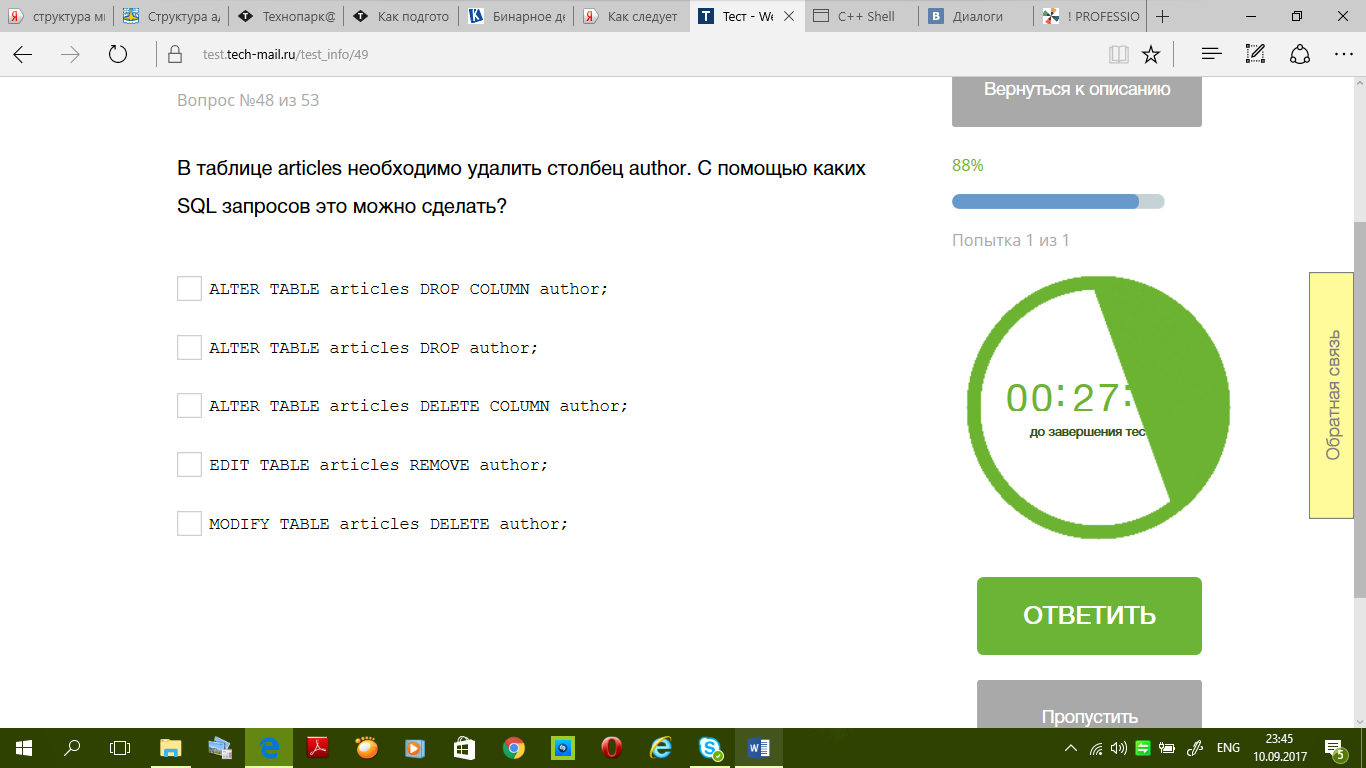
-- (users.id=cars.owner\_id) and (cars.brand='bmv')

-- select id, name, surname from users JOIN cars

-- USING (id) WHERE (cars.brand = 'bmv')

-- select distinct usrs.id, name, surname from users, cars ON

-- (users.id=card.owner\_id) WHERE (cars.brand='bmv')

**Вопрос №48 из 53**

Верные ответы:

ALTER TABLE articles DROP COLUMN author;

ALTER TABLE articles DROP author;

Синтаксически не верные ответы:

ALTER TABLE articles DELETE COLUMN author;

ALTER TABLE articles REMOVE author;

MODIFY TABLE articles DELETE author;

Пример: [http://sqlfiddle.com/#!9/365fb7/1](http://sqlfiddle.com/" \l "!9/365fb7/1)

CREATE TABLE IF NOT EXISTS `articles` (

`id` int(6) unsigned ,

`author` varchar(200) NOT NULL

) DEFAULT CHARSET=utf8;

INSERT INTO `articles` (`id`, `author`) VALUES

('1', 'Ayden'),

('2', 'Jayden'),

('3', 'Barack'),

('4', 'Sam'),

('5', 'Willa');

ALTER TABLE articles DROP COLUMN author;

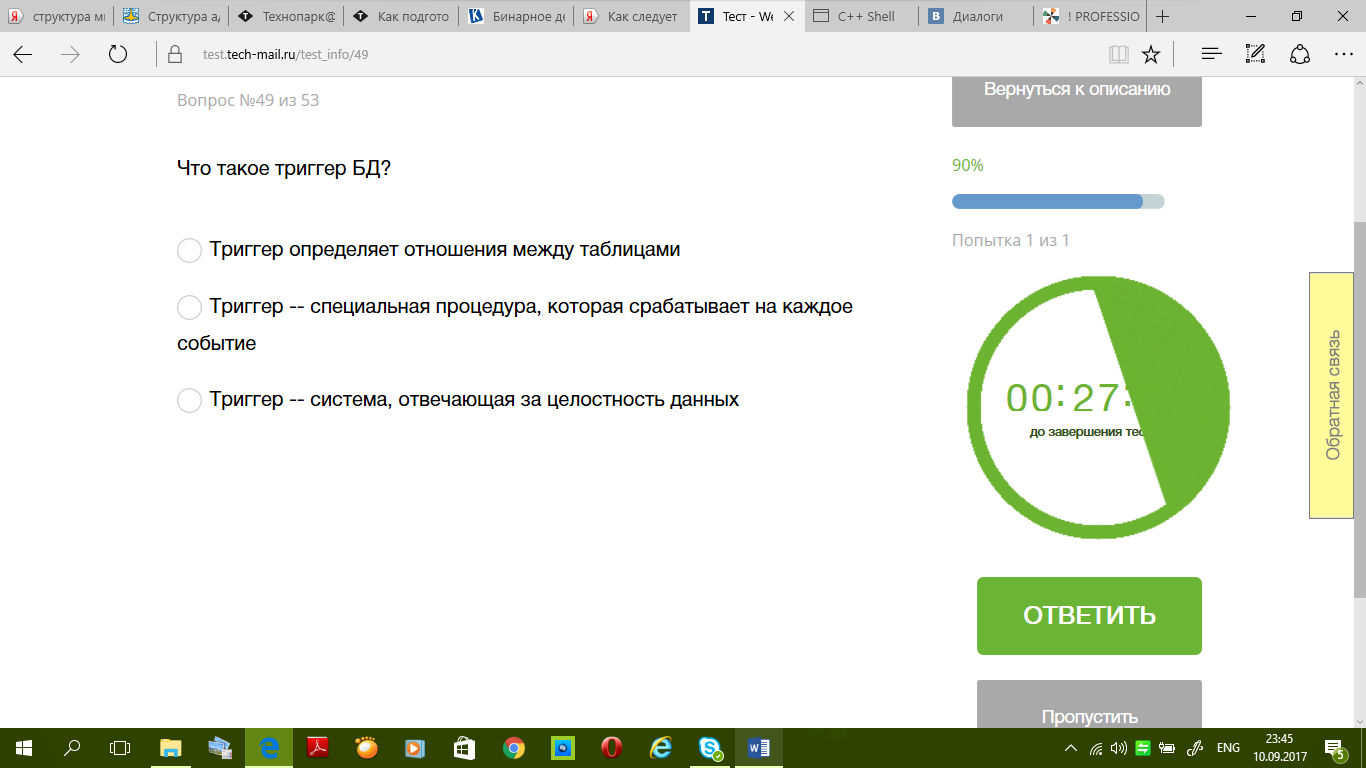
-- ALTER TABLE articles DROP author;

-- ALTER TABLE articles DELETE COLUMN author;

-- ALTER TABLE articles REMOVE author;

-- MODIFY TABLE articles DELETE author;

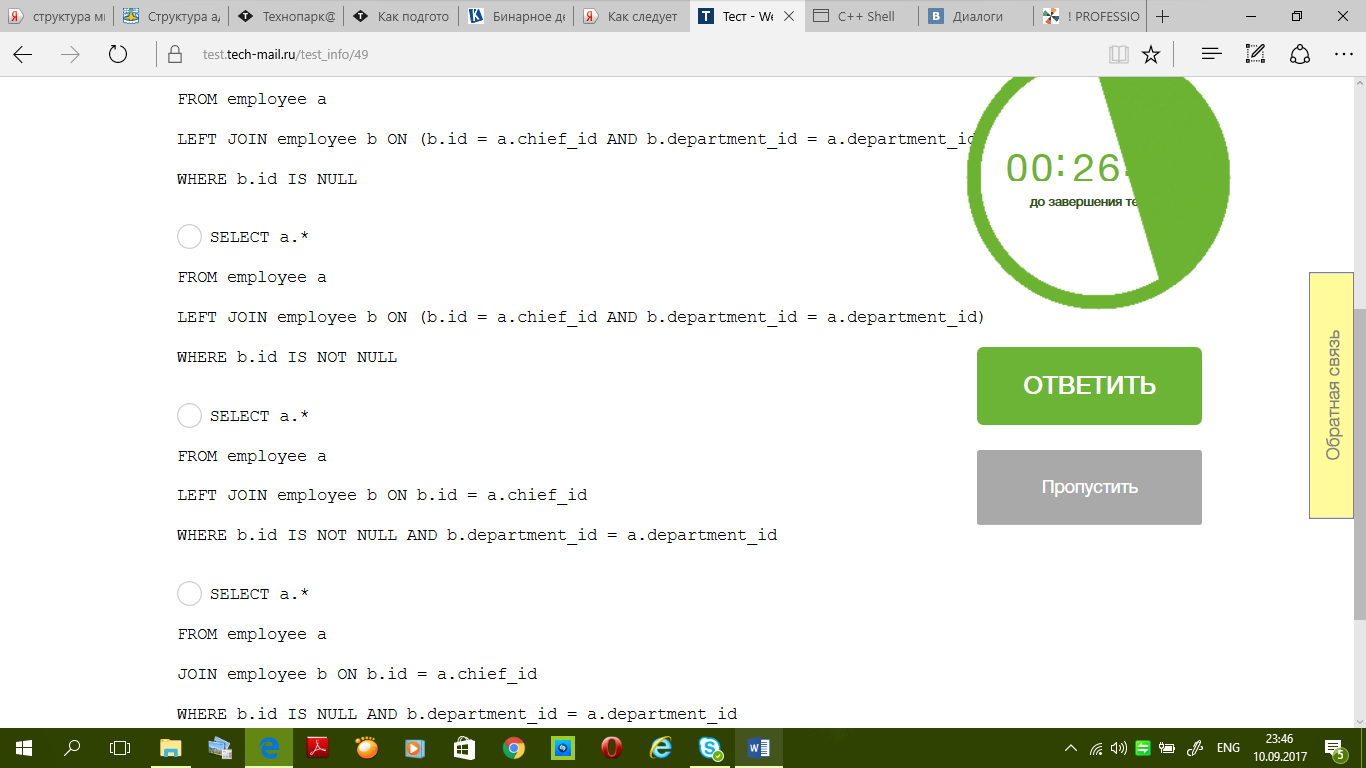
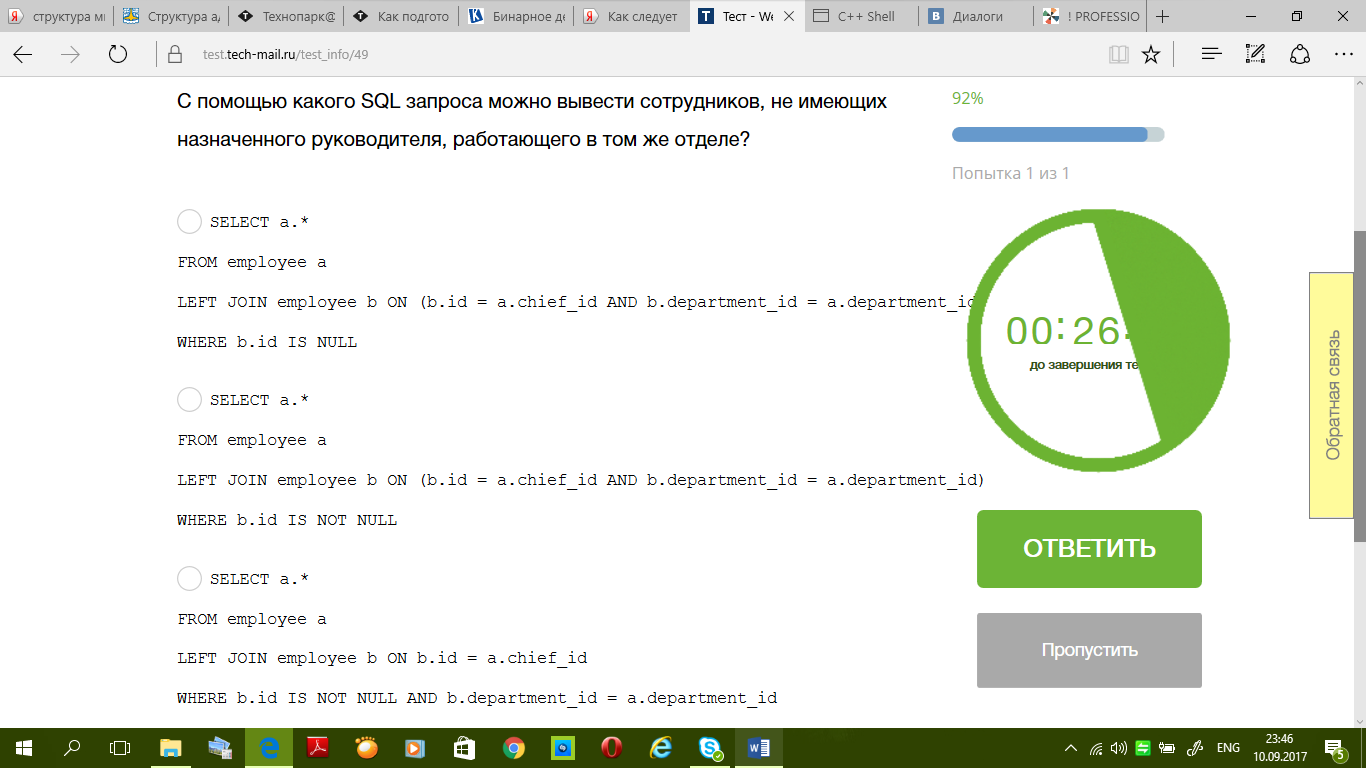
SELECT `author` FROM `articles`;

**Вопрос №49 из 53**

Правильный ответ:

Триггер — специальная процедура, которая срабатывает на каждое событие.

Три́ггер (англ. trigger) — хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определённом столбце заданной таблицы реляционной базы данных.

Вопрос №50 из 53

Правильный ответ:

SELECT a.\*

FROM employee a

LEFT JOIN employee b ON (b.id = a.chief\_id AND b.department\_id = a.department\_id)

WHERE b.id is NULL

Варианты:

№1

**Правильный вариант:**

SELECT a.\*

FROM employee a

LEFT JOIN employee b ON (b.id = a.chief\_id AND b.department\_id = a.department\_id)

WHERE b.id is NULL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **chief\_id** | **department\_id** | **id** | **chief\_id** | **department\_id** |
| 2 | 1 | 10 | 1 | (null) | 10 |
| 4 | 3 | 20 | 3 | (null) | 10 |
| 1 | (null) | 10 | (null) | (null) | (null) |
| 3 | (null) | 10 | (null) | (null) | (null) |

№2

**Тут выводятся сотрудники имеющие назначенного руководителя работающего в том же отделе::**

SELECT a.\*

FROM employee a

LEFT JOIN employee b ON (b.id = a.chief\_id AND b.department\_id = a.department\_id)

WHERE b.id is NOT NULL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **chief\_id** | **department\_id** | **id** | **chief\_id** | **department\_id** |
| 2 | 1 | 10 | 1 | (null) | 10 |
| 4 | 3 | 20 | 3 | (null) | 10 |
| 1 | (null) | 10 | (null) | (null) | (null) |
| 3 | (null) | 10 | (null) | (null) | (null) |

№3

**Тут опять выводятся сотрудники имеющие назначенного руководителя работающего в том же отделе:**

SELECT a.\*

FROM employee a

LEFT JOIN employee b ON b.id = a.chief\_id

WHERE b.id is NOT NULL AND b.department\_id = a.department\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **chief\_id** | **department\_id** | **id** | **chief\_id** | **department\_id** |
| 2 | 1 | 10 | 1 | (null) | 10 |
| 4 | 3 | 20 | 3 | (null) | 10 |
| 1 | (null) | 10 | (null) | (null) | (null) |
| 3 | (null) | 10 | (null) | (null) | (null) |

№4

**Условие b.id is NULL противоречит условию b.department\_id = a.department\_id:**

SELECT a.\*

FROM employee a

LEFT JOIN employee b ON b.id = a.chief\_id

WHERE b.id is NULL AND b.department\_id = a.department\_id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **chief\_id** | **department\_id** | **id** | **chief\_id** | **department\_id** |
| 2 | 1 | 10 | 1 | (null) | 10 |
| 4 | 3 | 20 | 3 | (null) | 10 |
| 1 | (null) | 10 | (null) | (null) | (null) |
| 3 | (null) | 10 | (null) | (null) | (null) |

Пример: [http://sqlfiddle.com/#!9/815b07/6](http://sqlfiddle.com/" \l "!9/815b07/6)

CREATE TABLE IF NOT EXISTS `employee` (

`id` int(6) unsigned NOT NULL,

`chief\_id` int(3) unsigned,

`department\_id` int(3) unsigned NOT NULL,

PRIMARY KEY (`id`)

) DEFAULT CHARSET=utf8;

INSERT INTO `employee` (`id`, `chief\_id`, `department\_id`) VALUES

(1, NULL, 10),

(2, 1, 10),

(3, NULL, 10),

(4, 3, 20);

SELECT a.\*

FROM employee a

LEFT JOIN employee b ON (b.id = a.chief\_id AND b.department\_id = a.department\_id)

WHERE b.id is NULL

/\*SELECT a.\*

FROM employee a

LEFT JOIN employee b ON (b.id = a.chief\_id AND b.department\_id = a.department\_id)

WHERE b.id is NOT NULL\*/

/\*SELECT a.\*

FROM employee a

LEFT JOIN employee b ON b.id = a.chief\_id

WHERE b.id is NOT NULL AND b.department\_id = a.department\_id\*/

/\*SELECT a.\*

FROM employee a

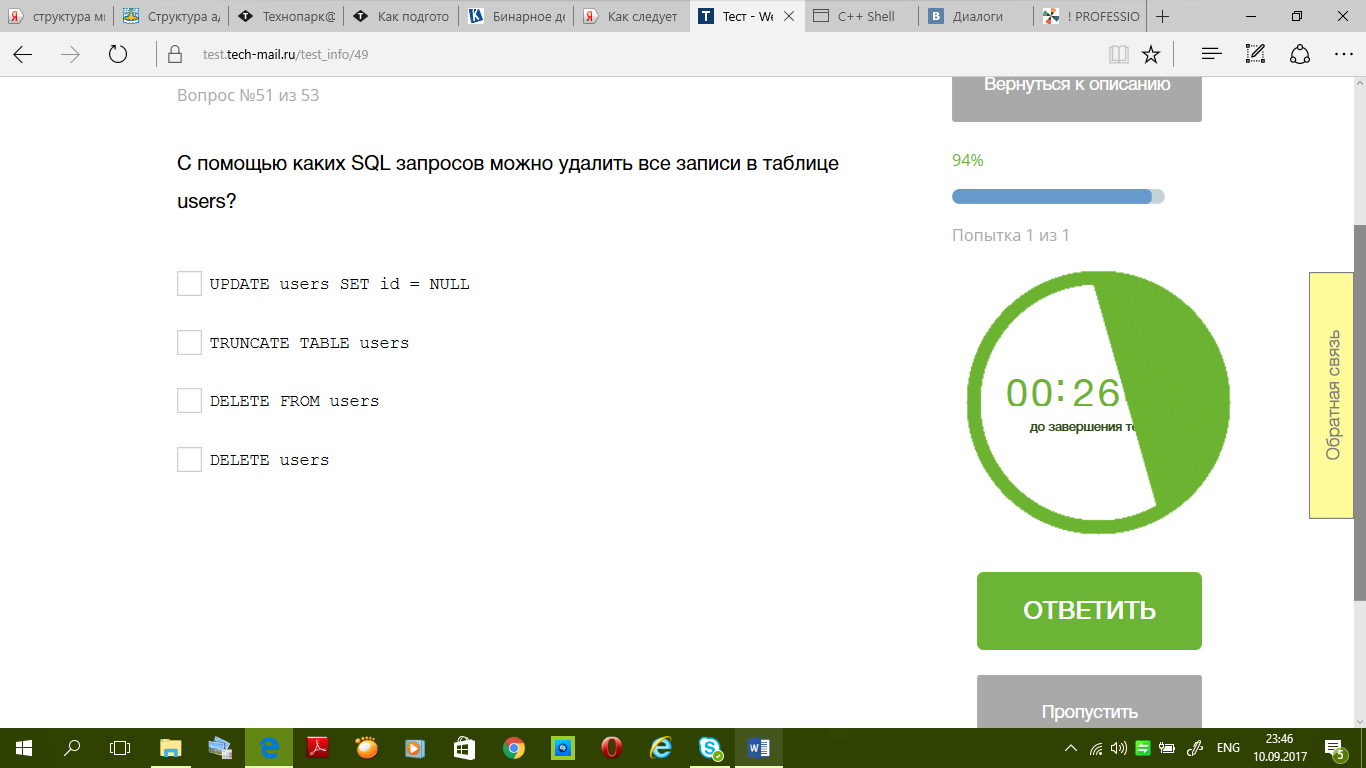
LEFT JOIN employee b ON b.id = a.chief\_id

WHERE b.id is NULL AND b.department\_id = a.department\_id\*/

/\*SELECT a.\*, b.\*

FROM employee a

LEFT JOIN employee b ON b.id = a.chief\_id\*/

**Вопрос №51 из 53**

Правильные варианты:

* TRUNCATE TABLE `users`;
* DELETE FROM `users`;

Запрос

UPDATE `users` SET `id` = NULL;

просто установить NULL всем записям в идентификатор, если это допустимо схемой.

DELETE `users`; синтаксически неверный запрос.

Пример: [http://sqlfiddle.com/#!9/a8bdc/1](http://sqlfiddle.com/" \l "!9/a8bdc/1)

CREATE TABLE IF NOT EXISTS `users` (

`id` int(6) unsigned ,

`name` varchar(200) NOT NULL

) DEFAULT CHARSET=utf8;

INSERT INTO `users` (`id`, `name`) VALUES

('1', 'Ayden'),

('2', 'Jayden'),

('3', 'Barack'),

('4', 'Sam'),

('5', 'Willa');

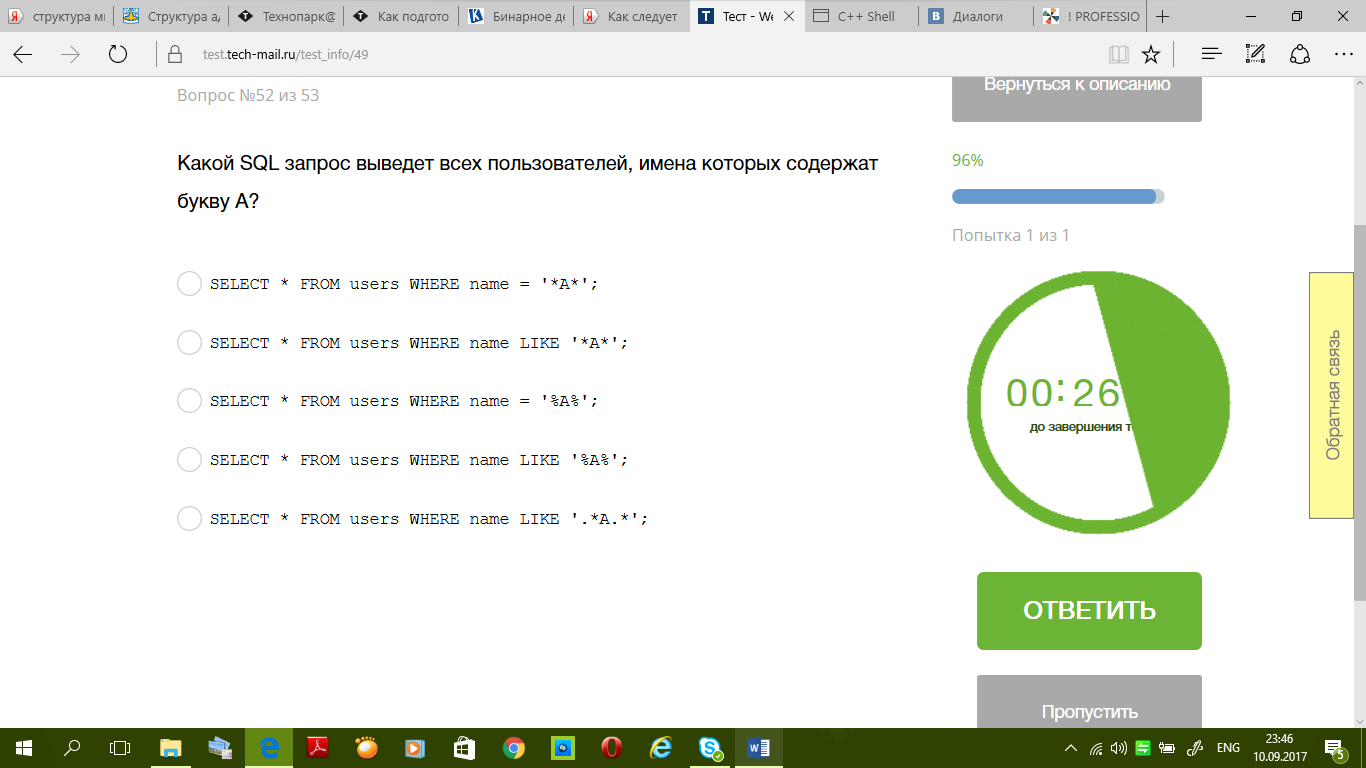
TRUNCATE TABLE `users`;

-- DELETE FROM `users`;

-- UPDATE `users` SET `id` = NULL;

-- DELETE `users`;

SELECT \* FROM `users`;

**Вопрос №52 из 53**

Верный ответ:

SELECT \* FROM users WHERE name LIKE '%A%';

Оператор SQL LIKE устанавливает соответствие символьной строки с шаблоном. Символ % обозначает строку любой длины.

Пример: [http://sqlfiddle.com/#!9/2884e6/17](http://sqlfiddle.com/" \l "!9/2884e6/17)

CREATE TABLE IF NOT EXISTS `users` (

`id` int(6) unsigned NOT NULL,

`name` varchar(200) NOT NULL,

PRIMARY KEY (`id`)

) DEFAULT CHARSET=utf8;

INSERT INTO `users` (`id`, `name`) VALUES

('1', 'Ayden'),

('2', 'Jayden'),

('3', 'Barack'),

('4', 'Sam'),

('5', 'Willa');

SELECT \* FROM users WHERE name LIKE '%A%';

/\*

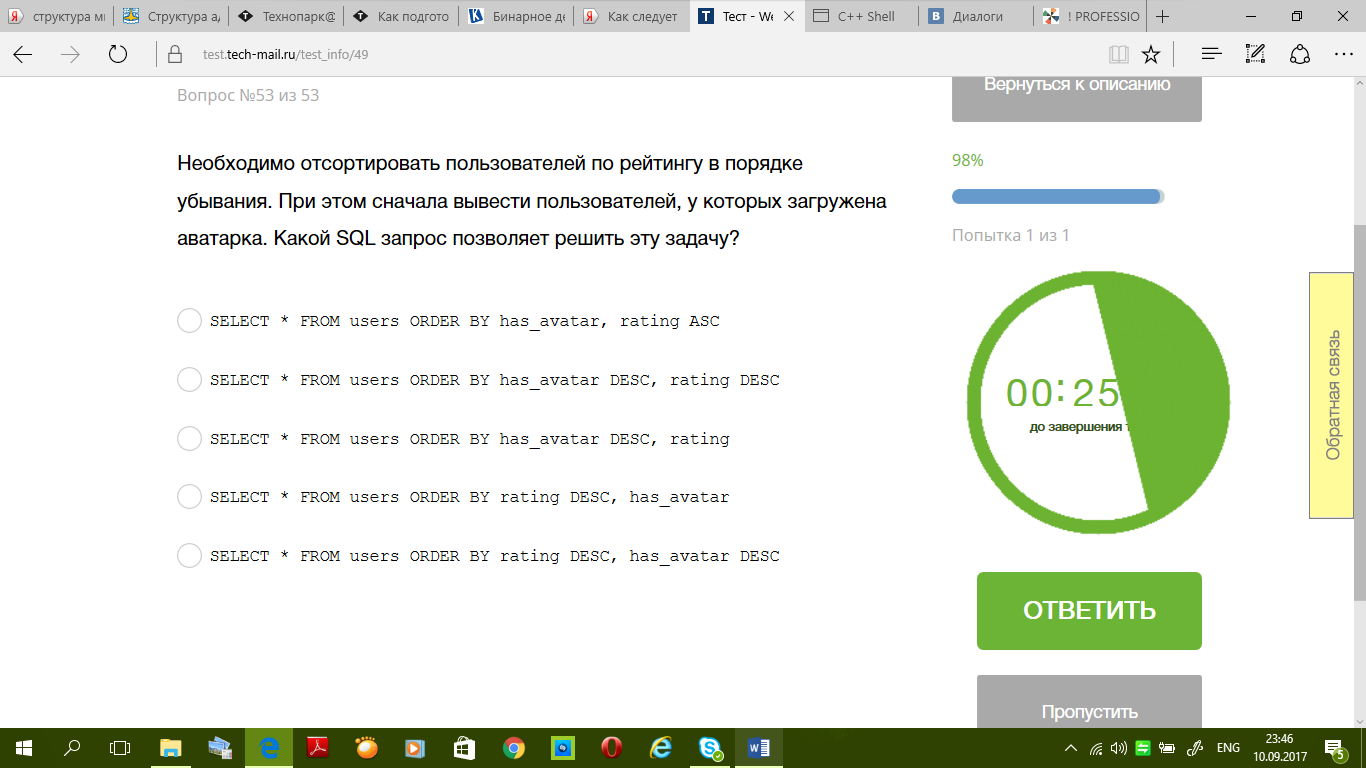
SELECT \* FROM users WHERE name = '\*A\*';

SELECT \* FROM users WHERE name LIKE '\*A\*';

SELECT \* FROM users WHERE name = '%A%';

SELECT \* FROM users WHERE name LIKE '.\*A.\*';

\*/

**Вопрос №53 из 53**

Верный ответ:

SELECT \* FROM users ORDER BY has\_avatar DESC, rating DESC

Во первых нужно вывести тех у кого есть аватарка, значит после ORDER BY должно сначала идти поле has\_avatar. boolean столбцы сортируются от false (0) к true (1), значит нужно ключевое слово DESC.

Во вторых нужно вывести пользователей по рейтингу тоже в порядке убывания, значит и здесь нужно слово DESC.

Пример: [http://sqlfiddle.com/#!9/c8335b/6](http://sqlfiddle.com/" \l "!9/c8335b/6)

CREATE TABLE IF NOT EXISTS `users` (

`id` int(6) unsigned NOT NULL,

`has\_avatar` boolean NOT NULL,

`rating` int(3) unsigned NOT NULL,

PRIMARY KEY (`id`)

) DEFAULT CHARSET=utf8;

INSERT INTO `users` (`id`, `has\_avatar`, `rating`) VALUES

('1', false, 10),

('2', true, 20),

('3', false, 30),

('4', true, 40);

SELECT \* FROM users ORDER BY has\_avatar DESC, rating DESC