

# «Научная химия»

## v. 1.0.0

Yunovidov D.V.

24 августа 2015 г.

## Содержание

<b>1 Введение. Предмет химии</b>	<b>1</b>
1.1 Актуальность науки и знаний . . . . .	1
1.2 Что такое химия и где она используется . . . . .	3
1.3 Что такое наука и как ей занимаются. . . . .	4
<b>2 Программирование и DIY (Do It Yourself)</b>	<b>6</b>
<b>3 Язык программирования.</b>	<b>6</b>
<b>4 Основные понятия языка.</b>	<b>8</b>
4.1 Среда разработки и окружение. . . . .	8
4.2 Синтаксис python — отступы, объекты, кодировка, модули. . . . .	8
4.3 Основные типы данных и структур. . . . .	11
4.4 Ввод и вывод. . . . .	13
4.5 Хранение данных и работа с файлами. Мини-программа в процедурном стиле. . . . .	13
<b>5 Процедурный, функциональный и объектно-ориентированный подходы</b>	<b>14</b>
<b>6 Программирование графического интерфейса</b>	<b>17</b>
6.1 Чем отличается от обычной программы — как правило всегда ООП	17
6.2 Что должно быть в классе для рисования окна. . . . .	17
6.3 Основные компоненты. . . . .	20
<b>7 Постановка эксперимента</b>	<b>27</b>

Весь изложенный материал представлен исключительно с личной позиции автора и не является агитацией.

# 1 Введение. Предмет химии

## 1.1 Актуальность науки и знаний

В этом небольшом введении мы бегло познакомимся с «расстановкой знаний» в современном мире с точки зрения автора.

Современная экономическая схема — схема потребления, на данный момент устаревает, но остается рабочей (в том числе и за неимением ничего лучшего). Данная схема главенствует в мире с 90 годов прошлого века. Для примера:

**Пример 1.1.** *Индустриальное общество продержалось примерно с 1910 до 1950 годов, далее великая депрессия. После — насколько направлений экономической мысли, самые «живучие» — основанные на идеях Маркса: капитализм и социализм. Далее — назовем это новой промышленной революцией и сегодня — общество потребления.*

Сейчас все большее развитие получает научная мысль, «умные гаджеты» и компьютерные технологии. Все это сплавляется с идеями потребления и доступностью для широких масс населения, что не так уж и плохо. Однако, чтобы быть профессионалом в современном мире и «не скучать» в нем в связи с ростом технологий и повсеместного их использования **необходимо понимать научные принципы и технологические особенности устройств.**

На сегодняшний день это не просто конкурентное преимущество — это образ жизни, и для нас жизненно необходимо качественно само-обучаться и самопереобучаться, умело ориентируясь в том потоке информации, который могут дать нам современные технологии (и интернет одна из них).

Проведя небольшой анализ рынка труда на сегодняшний день можно увидеть следующую картину:

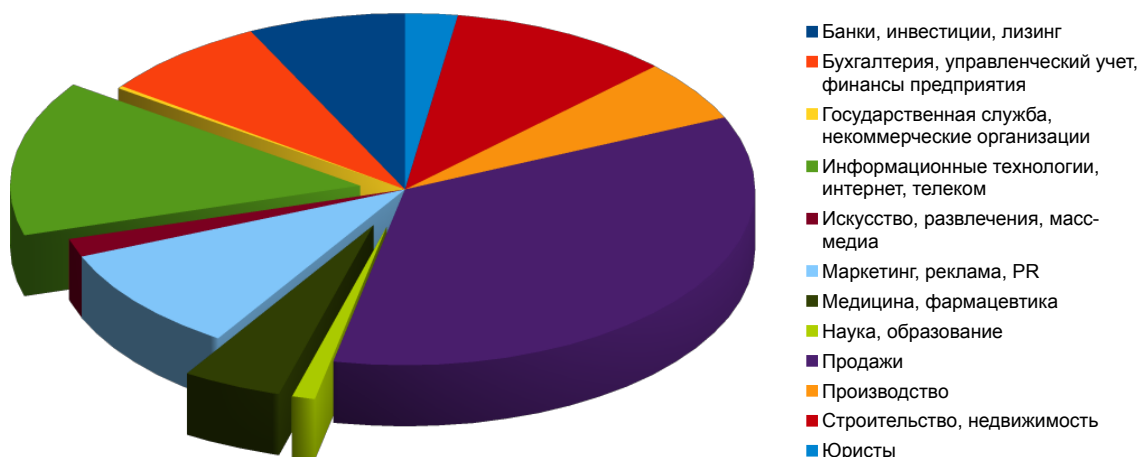


Рис. 1: Распределение вакансий на рынке труда, по данным hh.ru на апрель 2015 г. [?]

На данном примере видно, что предложения о работе поступают в основном от фирм по продажам и информационным технологиям. Однако не стоит забывать и о конкуренции внутри представленных сфер деятельности, а так же о вашей способности адаптироваться к внешним условиям. Не маловажным фактором является и карьерный рост. Если для того чтобы стать специалистом в области продаж, недвижимости или маркетинга не нужно особых научных знаний и профессиональной логики, то в сфере науки, информационных технологий (ИТ), медицины нужна логика, понимание материала и стройность знаний — что способствует развитию мышления и способностей быстро адаптироваться к изменяющимся внешним условиям. Так же не стоит забывать о карьерном росте и месте в обществе.

На следующих графиках представлен уровень зарплат и количество предложений от общего числа вакансий:

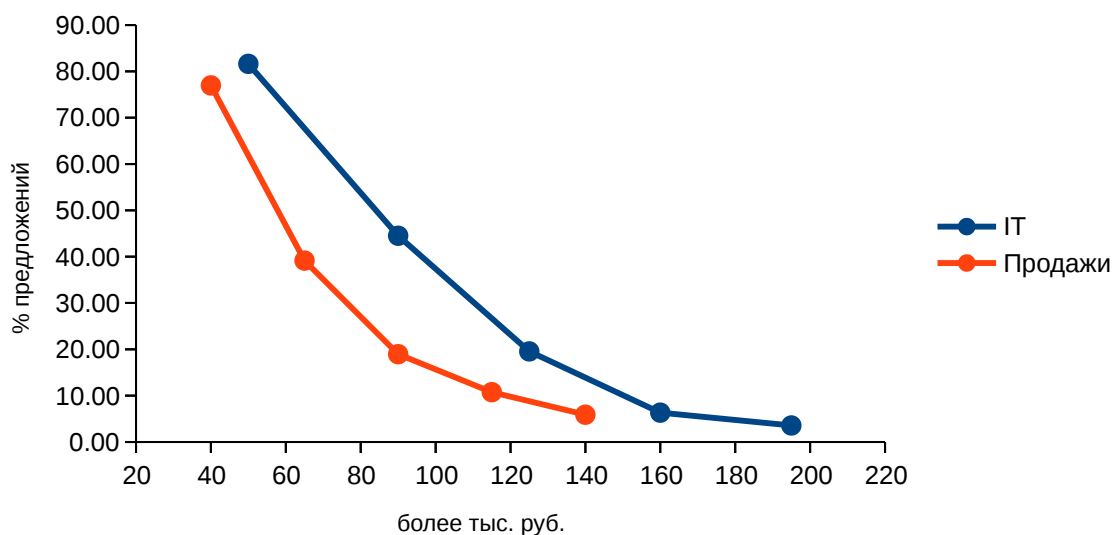


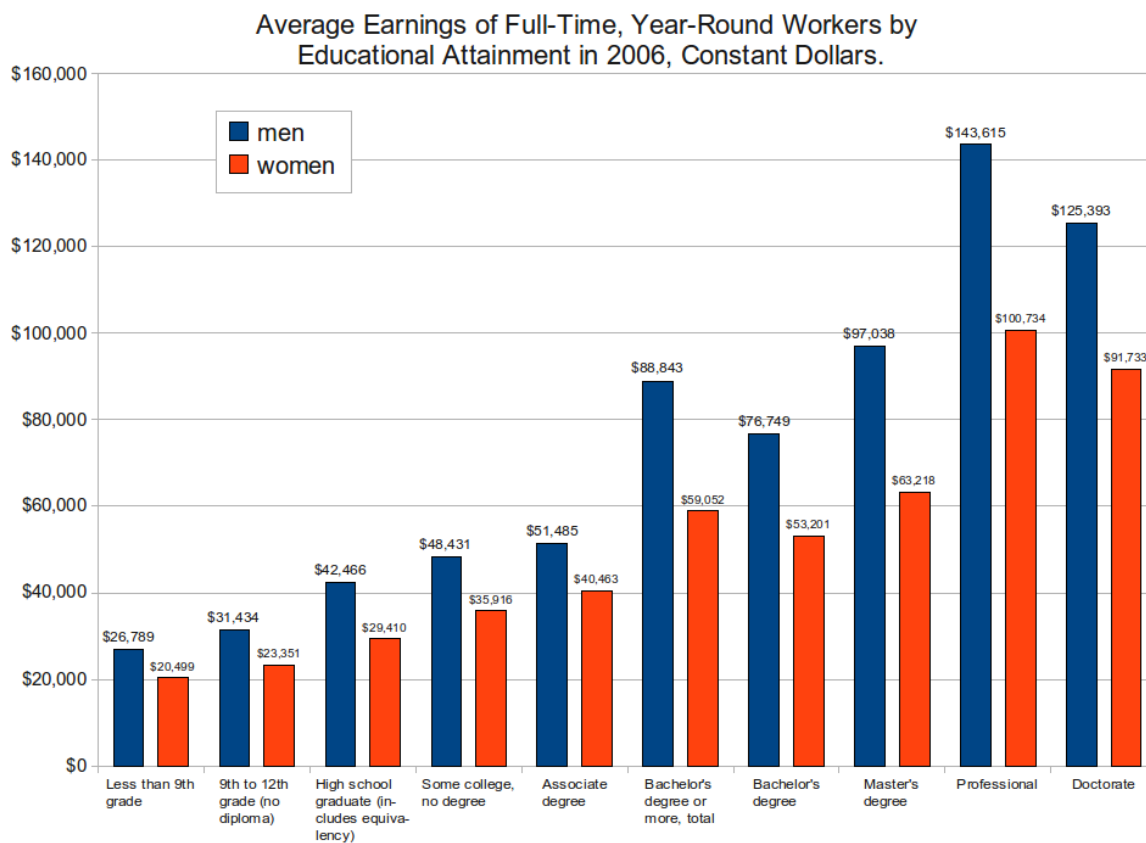
Рис. 2: Зависимость относительного количества предложений от уровня зарплаты по данным hh.ru, данные представлены на апрель 2015 г.

Можно заметить что карьерный рост выгоднее в более высокотехнологичных и научных сферах. На западе данная тенденция еще более выражена.

## 1.2 Что такое химия и где она используется

Нам требуется определить место нашего курса и это — **научная химия**. Именно данный вид химии на мой взгляд является наиболее обширным и способен охватить многие полезные в повседневной жизни вещи.

В рассмотренной структуре 1 на странице 2 химия в нашей стране, как область знаний, занимает маленькую но сформировавшуюся нишу. Зарплаты колеблются на среднем уровне, а карьерный рост не так явно выражен как в сфере информационных технологий. Но все меняется при смене системы координат. На западе



Source: Statistical Abstract 2009.

Рис. 3: Зависимость уровня зарплат от образования и пола, США, 2014 г.

данная область достаточно активно развивается, особенно в биологической отрасли и отрасли катализаторов.

При этом для западных людей проще работать на вакансиях с более простыми требованиями (без образования или в сферах маркетинга), а вот в науке и технологиях они нуждаются в работниках по контракту. Начальный уровень зарплат: от 2000 евро/долларов в месяц для PhD и от 3000-4000 евро/долларов для специалистов «без опыта», работающих в отрасли. Автор надеется что в свое время и в нашей стране наука получит финансовое признание.

Таким образом считаю актуальным и полезным рассказать вам о научной химии. Для интересующихся будем делать приличный уклон в сторону таких незаменимых на сегодняшний день инструментов химии, как программирования и обработки данных (статистика и математика), поскольку без этого в современной науке нам делать нечего.

Общепризнанным определением химии в России является:

химия — наука о веществах, их свойствах и превращениях.

Однако на сегодняшний день можно рассматривать в рамках химии материю в целом. Например в США принято рассматривать химию как:

Науку о материи и энергии.

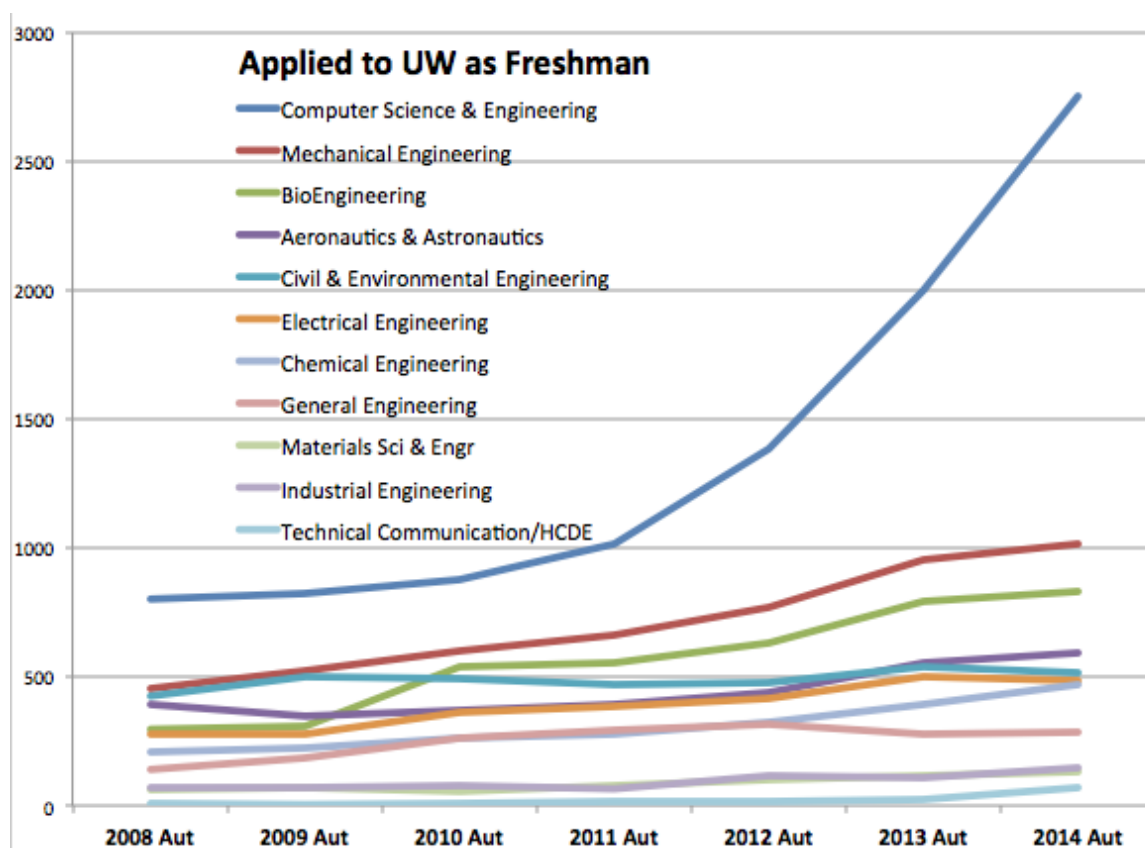


Рис. 4: Зависимость предложений на рынке от времени по специальностям. Европа.

Так же стоит включить в рассмотрение обработку информации, получаемой в рамках данной области знаний. Ибо без этого хорошим специалистом Вам не стать. Таким образом автор видит определение химии как:

Получение и обработка информации о материи и энергии, их свойствах и превращениях.

В рамках данного пособия автор постарается рассмотреть и совместить два взгляда на химию: Российский и западный.

### 1.3 Что такое наука и как ей занимаются.

Итак, наука и научный подход — что это такое?

Наука (от лат. scientia — знания) — системное представление о строении и организации знаний (информации) о вселенной с возможностью проверки и предсказания [?].

Вообще, как известно из древнегреческой философии, для изучения окружающего мира человеку доступно лишь две возможности: **наблюдение и эксперимент**. И на сегодняшний день данное утверждение можно принять за аксиому <sup>1</sup>.

<sup>1</sup>по крайней мере автор не встречал противоречий

Если использовать эти термины, то научную деятельность можно охарактеризовать следующим образом.

Основой этой деятельности является сбор фактов посредством наблюдений и экспериментов. Их постоянное обновление, систематизация и анализ, и как следствие всего этого — получение (синтез) новых знаний и законов нашего мира. Другими словами теории и гипотезы, которые подтверждаются фактами или опытами, формулируются в виде законов природы или общества [?].

Таким образом можно предположить следующий цикл научного познания, который рассмотрен например в данном курсе [?]:

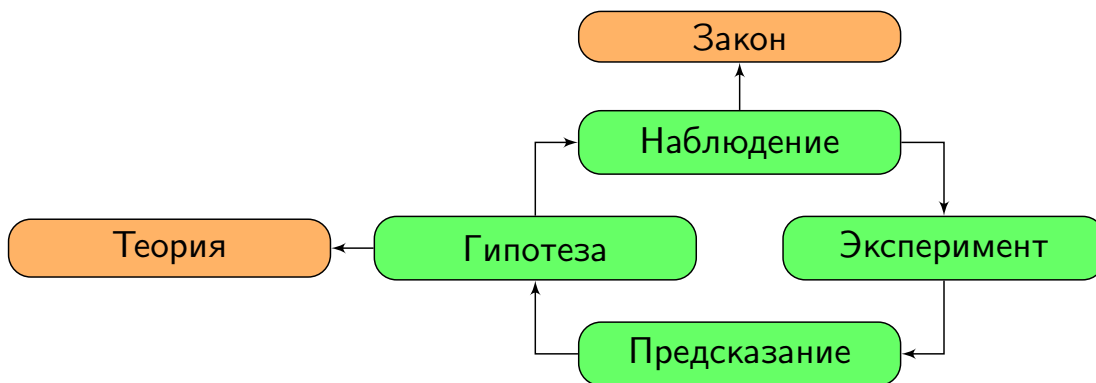


Рис. 5: Взаимосвязи научных понятий.

	применимо к единице или малому количеству случаев	применимо ко всем случаям
описывает <b>что</b> происходит	<i>наблюдение</i>	<i>закон</i>
объясняет <b>почему</b> явление происходит	<i>гипотеза</i>	<i>теория</i>

Таким образом, как правило в любой естественной науке <sup>2</sup>, мы с Вами будем работать над гипотезами и теориями, оперируя законами и наблюдениями. И все это в рамках химии будет связано с получением и обработкой информации о материи и энергии, их свойствах и превращениях. На сегодняшний день можно сказать, что это и есть химия.

Химия — наука, о получении и обработке информации о материи и энергии, их свойствах и превращениях на **микроскопическом уровне (атомов и молекул)**.

## 2 Программирование и DIY (Do It Yourself)

А начнем мы пожалуй с рассмотрения наших инструментов, поскольку именно они позволят нам качественно проводить практику и проверять теоретические

<sup>2</sup>а это в общем случае физика, химия и биология

знания. Да и работать хорошими инструментами всегда продуктивнее, легче и интереснее.

Итак, что нам даст программирование — логику мышления и возможность воплотить свое творчество/проекты вне зависимости от внешних условий.

### 3 Язык программирования.

Ключевое слово здесь — язык, но язык очень формализованный. Понимает только определенный набор слов и определенный их порядок, зато и учить легче=). Программирование это в первую очередь творческий инструмент, по крайней мере для меня. Но по скольку я не профессионал — не ждите от меня полноценного курса и объяснения всех нюансов. Расскажу о чем знаю, а если этого будет недостаточно будем вместе искать ответы и учиться. Учиться чему-то новому и полезному всегда увлекательно (советую вам себя в этом убедить=) ).

Таким образом возможность программирования дает нам творческую свободу (возможность реализовать практически весь спектр научных и бытовых идей имея под рукой лишь компьютер), возможность продуктивного общения и возможность качественной систематизации знаний. Ну и очень полезный практический навык, учитывая зарплату программистов.

Крайне рекомендую вам начать использовать ОС Ubuntu (linux), язык программирования Python 2.7 и среду программирования Pycharm (community edition).

1. Качаем выбранный дистрибутив (версия системы, на данный момент актуальна 14.04) с сайта <http://mirror.yandex.ru/ubuntu-releases/14.04.2/>, нам понадобится файл ubuntu-14.04.2-desktop-i386.iso для 32 разрядных систем и ubuntu-14.04.2-desktop-amd64.iso для 64-разрядных. На Ваш выбор.
2. Далее нужно установить программу unetbootin которая позволит нам установить образ диска на флешку и сделать ее загрузочной (при этом нам понадобится 2-4 Гб памяти). Следуйте инструкциям данной программы и создайте загрузочную флешку.
3. После создания загрузочной флешки втыкаем ее в USB и перезагружаем компьютер выставив в биосе (обычно при загрузке компьютера нужно нажать esc/tab/f12 - в зависимости от материнской платы) загрузку с флешки.
4. Устанавливаем систему. При штатной установке, нам понадобится в ручную разбить жесткий диск на следующие сектора: 1. boot — 512 М, ex3, основной; 2. swap — объем оперативки \* 2, основной; 3. /, порядка 10 Гб, ex4, основной; 4. /home — все остальное, ex4, логический.
5. После установки системы открываем терминал и вводим `sudo apt-get update`, `sudo apt-get upgrade`, `sudo apt-get install python-all`, `sudo apt-get install ipython`, `sudo apt-get install ipython-qtconsole`, `sudo apt-get install python-pyqt4` в данном порядке.
6. Заходим на сайт pycharm и качаем программу для ubuntu. (Нужна java)

7. Для более тонкой настройки системы рекомендую сайт: <https://sites.google.com/site/easylinuxtipsproject/Home>

Все возникшие вопросы вы можете задать в интернете или использовать Google для поиска ответов.

При использовании данных программ все остальные операции предельно упрощаются — запуск можно производить прямо из среды разработки. В ОС Windows — придется использовать вызов из консоли.

Так же немаловажная часть это самостоятельный поиск информации, т.к. эффективное программирование это в первую очередь самообучение и общение с интернет-сообществом. Здесь вам понадобится знание английского т.к. искать информацию на других языках в большинстве случаев не имеет смысла, *sad but true*. Ресурсы — google, stackoverflow, habrahabr.

Классический запрос по программированию состоит из названия языка, библиотеки, объекта который интересует и проблемы в 2-3 словах (или скопировать текст ошибки без уточнений).

**Пример 3.1** (Текст запроса в поисковую систему). *python math log Import error или PyQt4 QMainWindow close event*.

В качестве тренировки любой вопрос по программированию или если вы не знаете ответа на мой вопрос вам придется формулировать в виде запроса в поисковик на английском (кстати, для этого можно просто выставлять нужные слова в цепочку).

**Выбор языка программирования и какую нишу занимаем** . Наш выбор — Python: скриптовый язык и язык моделирования. Прост в изучении и пользовании, позволяет решать широкий круг задач: начиная от сетевого взаимодействия и проектирования сайтов, до создания игр и управления операционными системами. Примеров полно в интернете, можете открыть habrahabr и вбить python. Недавно начались попытки использовать для программирования микроконтроллеров.

## 4 Основные понятия языка.

### 4.1 Среда разработки и окружение.

Среда разработки — место написания программ на выбранном языке. Обычно включает в себя текстовый редактор, средства отладки и тестирования программы. Окружение — модули и библиотеки, которые будем использовать, а так же система в которой ведется разработка (windows, mac-os, \*nix). На рисунке 4.1 приведен вид Pycharm после небольшой настройки (проведите дома, на досуге)

Список горячих клавиш находится по адресу: File -> Settings -> Appearance & Behavior -> Keymap. Там же можно их и назначить по своему усмотрению. Так же среда разработки реализует множество полезных свойств, таких как: подсветка синтаксиса, автодополнение, интерактивная справка, регулярные выражения и фразы, и т.д.



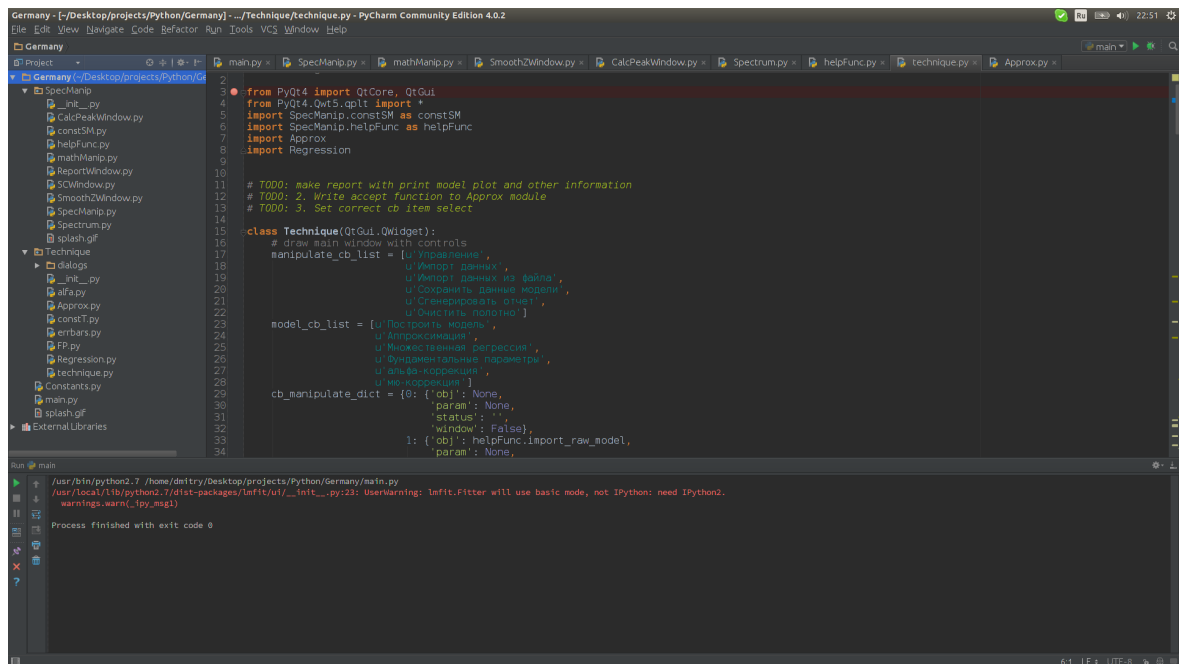


Рис. 6: Внешний вид среды разработки рυχарм. На рисунке видно окно проекта (слева), окно написание кода (справа), терминал для запуска кода (внизу). Так же в окне написание кода видна подсветка синтаксиса, регулярные фразы (TODO), точки отладки и открытые вкладки с модулями.

Так же можно настроить внешний вид среды. По умолчанию она идет в светлой теме, я лично предпочитаю темную. File -> Setting -> Apperance & Behavior -> Apperance -> Theme -> Dracula

## 4.2 Синтаксис python — отступы, объекты, кодировка, модули.

Итак — теперь к делу. Python — высокоуровневый язык с динамической типизацией, это означает что:

1. он не использует компилятор (т.е. он автоматически переводиться в двоичный код для данной машины, используя другой язык программирования);
2. он не требует сложного подхода для создания объектов, их можно объявлять динамически (т.е. в процессе набора кода). Для создания нового объекта достаточно приравнять его к понятному типу объектов.

**Пример 4.1.**  $a = 1$  — создается объект-ссылка (переменная)  $a$ , который является ссылкой на известный объект 1, типа  $int$ , который вносится в оперативную память в определенное место (индивидуальное для каждого нового объекта в памяти).

$b = 'выиф'$  — создается объект-ссылка  $b$ , -//- объект-строку

Большинство «создаваемых объектов» — это ссылки на заранее известные языку классы установленных объектов, но мы можем задавать их комбинации и присваивать им понятные нам названия (например как со строкой, каждая буква отдельно храниться в памяти но мы их объединили). При этом на сложные объекты (такие как списки и словари) используются «легкие ссылки», т.е. возможна следующая ситуация:

```
1 a = [1, 2, 3] # создаем список
2 b = [3] # создаем список
3 b = a # присваиваем одному списку значение другого
4 a.append(153) # добавляем в список a новое значение
5 print b
6
7
8 >> [1, 2, 3, 153]
```

Listing 1: Подводные камни «легких ссылок»

но это вы все прочувствуете на практике и все это легко «чиниться». Зато данное свойство значительно увеличивает скорость выполнения программ. Да, вся жизнь — сплошные компромиссы=). Так же важно соблюдать правила названия объектов — давайте пока использовать только буквы или их комбинации, так же можно использовать «\_» для разделения слов.

Исходя из всего вышеперечисленного обеспечивается гибкость и простота использования языка, но часто в ущерб его скорости выполнения, однако в нашем случае это не критично.

Чтобы язык понимал что мы хотим ему сказать важно соблюдать формальности, например — синтаксис и отступы.

Отступы — это важнейшая часть языка, показывающая ему какой блок кода в какой конструкции используется. За счет этого же обеспечивается понятность и читаемость кода.

Стандартный отступ 4 пробела (обычно назначается на клавишу tab).

```
1 a=[1,2,3,4,5] # создаем список
2 for i in a: # задаем цикл обхода значений в списке
3     print i # печатаем значение
4
5 >> 1
6 2
7 3
8 4
9 5
```

Listing 2: Отдельный блок цикла

Так же немаловажным действием при программировании является комментирование своего кода для пояснения того что вы делаете и зачем. Комментарии начинаются со знака «#» и должны кратко включать в себя основную информацию о ваших действиях, чтобы другие программисты могли понять для чего служит тот или иной блок кода.

```

1 # Create main parameter
2 a = 1
3 b = 0
4 c = 10
5

```

Listing 3: Отдельный блок цикла

С синтаксисом же предлагаю ознакомиться подробнее в процессе работы. Сейчас упомяну лишь несколько правил которые приходят на ум.

1. Числа, и в Африке — числа.
2. Строки заключаются в кавычки `'sdf'`, `"s'd'f'"` `sdfefw ''`.
3. Список — `[]`.
4. Словарь — `.`
5. Массив — `()`.
6. Принятые конструкции:
  - (a) `if: elif: else:`
  - (b) `while`
  - (c) `for i in [1,2,3,4,5]`

Со всем остальным — в процессе работы.

### 4.3 Основные типы данных и структур.

Итак, теперь загрузим необходимый минимум в наш арсенал. Lock and load!==)

**Числа.** Объекты языка, которые делятся на целые (`int`) и числа с плавающей точкой (`float`) и это пока все что нужно знать. Для этого типа объектов определены операции сложения, вычитания, умножения, сравнения и т.д. Все это позволяет использовать `python` как калькулятор — давайте поэкспериментируем в `ipython`. Пара задач на вычисления.

#### Задача 4.1.

```

2 125*2
3
4 a = b = 50
5 a += 2
6 a + b
7

```

Listing 4: Введите данные строки в консоль `ipython` и запишите результат.

**Строки.** Это буквы и их комбинации в той или иной кодировке. По умолчанию `ASCII`, которая включает латиницу но «падает» на кириллице. Так что пока везде используем английский язык. Для данного типа объектов определены операции сложения, умножения на число, сравнения, логические. Так же есть набор специальных функций для работы со строками.

**Пример 4.2.** Пару примеров (выполнить в *ipython* и посмотреть на результат).

```
1 a = '101'
2
3 b = '1'
4
5 c = 3
6
7 3 * b
8
9 a + b
10
11 a < b
12
13 not b
14
15 len(a)
16
```

Listing 5: Операции со строками.

**Списки.** Упорядоченный набор элементов. Элементы могут быть любыми: числа, строки, объекты, другие списки, словари, черти в ступе и.т.д.. Обладают рядом собственных методов.

```
1 [1, 2, 3].append(5)
2 >> [1, 2, 3, 5]
3
4 [1, 2, 3] + [4, 5]
5 >> [1, 2, 3, 4, 5]
6
7 [1, 2, 3] + 5
8 >> Error
9
10 [1, 2, 3].append([4, 5])
11 >> [1, 2, 3, [4, 5]];
12
13 len([1, 2, 3])
14 >> 3
15
```

Listing 6: Операции со списками.

Можно обращаться к элементу по индексу: `a[0]`, `a[-1]`, `a[4:7]` и т.д. читайте литература и разбирайте дальнейшие примеры.

### Словари.

Неупорядоченный набор (т.е. могут выводиться в случайном порядке и нельзя обратиться по индексу) пар значений типа `ключ:значение`. При этом ключами могут являться только не изменяемые объекты (строки, числа, массивы), значения — все что угодно, как и списки, и словари. Обладают так же набором своих методов.

```
1 a={'a':125, 'b':33, 'c':55, '1':100}
2
3 a['a']
```

```

4 >> 125
5
6 a['2'] = 101
7 a
8 >> {'a':125, 'b':33, 'c':55, '2':101, '1':100}
9
10 for i in a:
11     print 'key: ', i
12     print 'data: ', a[i]
13 >> key: b
14 data: 33
15 key: 2
16 data: 101
17 $\cdots$
18

```

Listing 7: Операции со словарями.


Подробнее о рассмотренных объектах и о прочих объектах и методах — читайте у Прохоренко.

## 4.4 Ввод и вывод.

Каждая программа и любое наше действие в жизни обладают входом и выходом. Т.е. мы имеем что-то на входе, обдумываем и производим какие-то действия и получаем что-то на выходе=). Будь то забивание гвоздя в доску или написание программы — все делается с какой-то целью: получить из данного — желаемое.

Итак, питон предоставляет нам несколько вариантов.

1. Консольные программы: используют метод (функцию) `raw_input()` которая приглашает пользователя ввести данные.

**Пример 4.3.** `a = raw_input('Please tell me you age: ')` (В python 3 - `input()`). Выведет в консоли заданный текст и приостановит действие программы пока пользователь не введет данные и не нажмет , после этого ввод пользователя будет присвоен переменной (объекту-ссылке) «a», в виде строки. Данную строку можно перевести в число методами `int()` или `float()`.

```

1 a = raw\_input('Please tell me you age: ') # Консоль будет ждать
   пользовательского ввода и присвоит его переменной a
2
3 >> 10
4
5 a = int(a) # переназначит переменную на объектцелое— число
6 print a # выведет на экран значение переменной a, при работе в консоле
   можно просто вызвать переменную и нажать Enter.
7 >> 10
8

```

Listing 8: Консольный ввод-вывод.

**Задача 4.2.** Напишите программу, которая спрашивает у пользователя имя и возраст и добавляет их в словарь `all_name`. При ответе пользователя

`all_name` — программа выводит на печать словарь, при ответе пользователя `exit` — пишет `bye` и завершается.

Дополните программу: при вводе пользователем `sort`, программа создает список возрастов, сортирует его по возрастанию и выводит данный список в консоль.

2. Программы с ГУЕм (GUI — Graphical User Interfacing) используют систему сигналов и слотов, а так же методы некоторых объектов, таких как текстовое поле (TextEdit) и линия ввода (LineEdit) для ввода и вывода информации. Об этом мы поговорим позднее на конкретных примерах.

## 4.5 Хранение данных и работа с файлами. Мини-программа в процедурном стиле.

Итак, предположим мы набросали небольшую программу запрашивающую у каждого запустившего ее имя и возраст, но как нам сохранить эти данные? Ответ прост — вести учет и использовать запись и чтение из файла (т.е. ввод данных идет через консоль, а вывод — в файл). В нашем случае возьмем «\*.txt» файлы (\* — это часть регулярных выражений, позволяющие упрощать работу с похожими файлами, в нашем случае \* отражает любое имя файла, заканчивающееся на .txt).

**Пример 4.4.** *Напишем программу запрашивающую имя и возраст пользователя и сохраняющую данную информацию в текстовый файл.*

```
1 # В начале файла нужна строка говорящая программе о том, что мы можем
   использовать не только английский язык:
2 # -*- coding: utf-8 -*-
3
4 file_out = open('/home/dmitry/Desktop/name.txt', 'w') # открытие файла,
   ой2 аргумент говорит, что открываем файл для записи, мб.. 'r', 'r+', 'w',
   'w+', 'a', 'a+' — по минимуму, но нам хватит
5
6 print 'For exit type exit' # Предупреждаем пользователя, что для выхода
   надо ввести exit и нажать <<Return>>
7
8 name = age = '' # Создаем 2 переменных и присваиваем им значение пустой
   строки
9
10 while True: # Запускаем бесконечный цикл
11     name = raw_input('Hello! Tell me your name please: ')
12     if name == 'exit':
13         break # Прерываем цикл
14     file_out.write(name + '\n')
15     age = int(raw_input('Now, tell me your age please: '))
16     if age == 'exit':
17         break
18     file_out.write(str(age) + '\n') # Записываем данные в файл
19
20 print 'Bye'
21
22 file_out.close() # После работы с файлом нужно обязательно закрыть его.
```

### Listing 9: Консольная программа с выводом в файл.

*После выполнения программы в /home/%username%/Desktop появится файл names.txt содержащий имена и возраст. Не забываем что в поле %username%, должно стоять Ваше имя.*

## 5 Процедурный, функциональный и объектно-ориентированные подходы

Итак, кое-какие основы заложили. Начинаем вещи по-веселее=). А именно какие существуют глобальные подходы к программированию:

1. Процедурный — построчное выполнение кода, например наша программа по записи данных пользователей. Но это как-то не творчески и вообще трамвайно=).
2. Функциональный — подход с использованием функций, это куски кода, собранные под определенным именем, к которому можно обращаться из любой части внешнего кода. Например, известная нам функция `print()` или `int()`. Можем и сами создать функцию.

```

1 def go_func (x, y):
2     a = 'go' + str(x)
3     b = '_func' + str(y)
4     return a+b # Данный метод используется для возвращений определенных
                  объектов после работы функции, то.. функцию можно присвоить переменной
5
6 c = go_func('a', 'b')
7 print c
8
9 >>> 'go_a_func_b'
10
```

### Listing 10: Функции

При этом объекты определенные внутри функции вне функции не определены (ввести метод `global` и пояснить работу с переменными)

Перепишем программу для записи имени и возраста в функциональном стиле:

```

1 # -*- coding: utf-8 -*-
2
3 def write_data(file_write, data_str): # Создаем функцию "write\_data"
4     " с 2 аргументами — файл1 — для записи, 2 — строка для записи
5     file_write.write(data_str)
6     print 'write well'
7
8 # Здесь мы не используем метод return, тк.. нам не нужно чтобы функция
9   что-то — возвращала, нам нужны лишь конкретные действия
10
11 file_data = open('/home/dmitry/Desktop/name.txt', 'a+') # открываем
12   файл для дополнения дозаписи() материала
```

```

9
10 print 'For exit type exit'
11
12 name = age = ''
13
14 while True:
15     name = raw_input('Hello! Tell me you name please: ')
16     age = int(raw_input('Now, tell You age please: '))
17     if name == 'exit' or age == 'exit':
18         break
19     write_data(file_data, name+'\t'+str(age)+'\n') # Вызываем
    функцию, передав ей нужные аргументы
20
21 print 'Bye'
22
23 file_data.close()
24

```

Listing 11: Функции

**Задача 5.1.** Задача: Ряд с числами Фибоначчи — см книгу.

3. Ну и наконец, самый сейчас распространенный — объектно-ориентированный. Заключается в том, что мы создаем собственный класс объектов, некое множество, которое включает в себя определенные объекты (обладающие тем или иным свойством). Этот подход очень удобен и всеми любим т.к. позволяет «очеловечить» код, разбить его на группы объектов с теми или иными свойствами. Например, если мы хотим описать наш класс, мы можем начать перечислять все объекты по очереди (т.е. используя процедурный подход) и будет составлять длинный и уникальный список для каждой из возможных аудиторий — это не наш метод, гораздо удобнее и понятнее разбить все объекты на классы, например по их функциям или материалу из которого они изготовлены и сказать что для каждого объекта из всех объектов аудитории если он деревянный то отправляем его в класс деревянных объектов, где он классифицируется и заносится в список «деревянных» объектов и т.д.. (кстати только что был пример псевдокода, ныне устаревшего понятия). Классы в питоне объявляются так:

```

1 # -*- coding: utf-8 -*-
2
3 class Wood(object): # В скобках указан предок класса, те.. та группа
    объектов в которую входит данный класс, здесь используется общее понятие
    objects, обозначающее все возможные объекты которые знает питон
4     # consist all wood objects — поясняем для чего служит тот или иной
    класс
5     objects = []
6     objects_name = []
7
8     def __init__(self, new_object, name='HZ'): # конструктор"" класса,
    функция которая вызывается при обращении к данному классу
9         self.objects.append(new_object) # для каждого создаваемого
    объекта внутри класса требуется приставка self. — опять же
    формализм...

```



```

10         self.objects_name.append(name)
11
12     # Определяем прочие функции можно( назвать методами) внутри данного
    класса
13     def take_name(self, new_object):
14         ind = self.objects.index(new_object)
15         return self.objects_name[ind]
16
17     def take_obj(self, name):
18         ind = self.objects_name.index(name)
19         return self.objects[ind]
20
21     def add_new(self, new_object, name='HZ'):
22         self.objects.append(new_object)
23         self.objects_name.append(name)
24

```

Listing 12: Функции

Важно — не забывайте о наследовании, иначе формальный язык вас не поймет, все как в лучших домах Англии=).

Подробнее классы мы будем рассматривать в программировании пользовательского интерфейса.

## 6 Программирование графического интерфейса

Пора рассмотреть то, чем можно похвастаться окружающим и почувствовать силу=). Программирование графического интерфейса или попросту GUI.

### 6.1 Чем отличается от обычной программы — как правило всегда ООП

В чем же разница GUI помимо, собственно, вида? А разница огромна. Если консольная программа позволяет сосредоточиться на алгоритмах, сложных программных подходах и математике, то интерфейс нужен для взаимодействия с широким кругом пользователей и не несет в себе вычислительной мощности и сложных подходов. Однако он является достаточно разветвленной и обширной структурой, со своими особенностями. В первую очередь это выбор библиотеки для рисования интерфейса — и это пожалуй самое верное слово. Все объекты ГУИ нарисованы на полотне и если каждая ГУИ-библиотека предоставляет пользователю способ рисования и главное полотно, то далеко не каждая может похвастаться разнообразием готовых инструментов и структур, а так же массивом возможностей их взаимодействия.

Мы остановимся на PyQt — это модификация C++ библиотеки «Qt» для питона. На данной библиотеке написаны такие вещи как интерфейс некоторых телефонов Nokia, kde-оболочки для linux и прочее. Библиотека достаточно мощная, старая и со множеством возможностей и большим сообществом. Ну и пригодится если захотите потом переползть на более серьезный C++=).

Теперь непосредственно к особенностям программирования ГУЯ.

1. Пишется все как правило в ООП-стиле, т.к. это чрезвычайно удобно для создания отдельных окон.
2. Существует главный цикл приложения, позволяющий постоянно перерисовывать интерфейс и отвечать на действия пользователя (например нажатие кнопки и т.д.) — основной компонент GUI-программы.
3. Система сигналов и слотов — аналог ввода-вывода для консоли.
4. Потоки и очереди действий, чтобы программа не зависала при длительных вычислениях.

## 6.2 Что должно быть в классе для рисования окна.

Итак к делу. Здесь приведен пример программы, разберемся какие части должен содержать каждый из компонентов.

```

1  # -*- coding: utf-8 -*-
2  # яа1 строчка нужна если в коде будут любые другие языки кроме английского, лучше
   # взять за правило всегда ее добавлять
3
4  __author__ = 'dmitry' # можем ради прикола указать автора=)
5
6  # импортируем нужные библиотеки
7  from PyQt4 import QtCore, QtGui
8  import sys
9  import time
10
11 # TODO: 1. запустить этот код=)
12
13 # создаем свой класс для рисования главного и( пока единственного) окна нашей
   # программы
14 class MainWindow(QtGui.QWidget): # для простоты пока наследуемся от класса
   # виджетов, т.е. наше главное окно программы по сути будет скоплением виджетов
15     # Main window — поясняем зачем данный класс
16     def __init__(self): # определяем конструктор<<>> класса,
17         QtGui.QWidget.__init__(self, parent=None) # запускаем конструктор
   # предка — формальная необходимость
18         desktop = QtGui.QApplication.desktop() # определяем разрешение
   # рабочего стола пользователя
19         x = desktop.width() # берем из найденного разрешения длину
20         y = desktop.height()
21         self.move(x / 2, y / 2) # перемещаем окно программы на центр
   # экрана если( не вызвать данную функцию окно будет появляться в рандомном
   # месте экрана)
22         self.setWindowTitle("Test GUI program") # Создаем заголовок окна
23         # Конец формальной части — дальше определяем объекты которые будут у
   # нас в окне программы
24         self.main_vl = QtGui.QVBoxLayout(self) # создаем объект класса (
   # self.*) который определим как вертикальный слой, содержащий другие элементы.
   # При этом слой это объект вида QVBoxLayout который наследуется от более
   # глобального класса объектов QtGui оцените( формальный синтаксис). Так же
   # поскольку он единственный объект напрямую связанный с классом нашим(
   # главным окном) он должен содержать ссылку на класс: (self). Прочие объекты

```

```

будут определяться инстанцироваться() в данном слое и ссылки на родителя там
не нужны.
25     self.main_l = QtGui.QLabel() # создаем объект-лэйбл уже ( без ссылки
на родителя, см. выше)
26     self.main_l.setText('Push the buttons and see the action') #
присваиваем лэйблы текст функцией setText(), которая определена для данного
класса объектов, поэтому можно вызвать ее через точку
27     self.main_vl.addWidget(self.main_l) # инстанцируем объект-лэйбл в
слой чтобы он корректно отображался у нас в окне
28     self.main_te = QtGui.QTextEdit() # создаем объект-редактируемое-
текстовое поле
29     self.main_vl.addWidget(self.main_te)
30     self.buttons_hl = QtGui.QHBoxLayout() # создаем
объект-горизонтальный- слой
31     self.left_pb = QtGui.QPushButton('Left_push') # создаем
объект-кнопку-
32     self.right_pb = QtGui.QPushButton('Right_push')
33     self.right_pb.setCheckable(True) # делаем одну из кнопок
зажигающей""
34     self.buttons_hl.addWidget(self.left_pb)
35     self.buttons_hl.addWidget(self.right_pb)
36     self.main_vl.addLayout(self.buttons_hl) # инстанцируем один слой в
другой
37     # Конец части определения объектов, теперь определим сигналы и слоты,
те.. как объект должен реагировать на действие со стороны пользователя
38     # Для определения сигнала и действия используется функция connect(),
из предка QObject, из праредка- QtCore, который мы импортировали в самом
начале кода, те.. можем его вызывать, а вот всех его детей<<>> нужно
указывать явно. Далее первым аргументом идет объект в отношении которого
совершается действие, в нашем случае кнопка, ым2 аргументом — какое действие
ждем, им3 — какую функцию вызываем в ответ на действие. При этом список
доступных действий определен для каждого объекта в нашей библиотеке (PyQt4)
39     QtCore.QObject.connect(self.left_pb, QtCore.SIGNAL('clicked()'),
self.left_click) # действие — нажатие (clicked())
40     QtCore.QObject.connect(self.right_pb, QtCore.SIGNAL('toggled(bool
)'), lambda x: self.right_toggle(x)) # действие — зажатие (toggled(bool
)), при этом действие дает нам булево значение зажатия (True/False), а вот
чтобы передать функции какойлибо- аргумент нужно использовать схему в lambda
функцией-, считайте это формальным синтаксисом
41
42     # Конец конструктора. Все прочие блоки кода при вызове класса автоматически
не вызываются. Далее описываем собственные методы внутри нашего класса.
43
44     def left_click(self):
45         # Call when user click left button
46         self.main_te.append('left click activated!\n') # добавляем текст
в наше текстовое поле
47
48     def right_toggle(self, flag):
49         # Call when user click right button
50         i = 0 # переменная-счетчик-: сколько секунд зажата кнопка
51         if flag: # условие, если ой2 аргумент зажата( кнопка или нет)
функции True, то выполняем блок кода
52         while self.right_pb.isChecked(): # блок кода выполняется пока
объект-кнопка- находится в нажатом состоянии, проверяем данное состояние
методом те(.. определенной для данного объекта функцией) isChecked(),

```

```

        которая возвращает True/False
53         self.main_te.append('right button toggled... {0} second!'
        .format(i)) # добавляем текст в текстовое поле
54         time.sleep(1) # говорим программе уснуть на 1 секунда
        имитируем( загруженный процесс)
55         QtGui.QApplication.processEvents() # чтобы программа не
        зависла"" для системы и система не предложила нам ее убить"", прерываем все
        остальные процессы для запуска основного цикла приложения
56         i += 1
57
58
59 reload(sys) # перезапуск системного модуля — понадобится нам если мы
        захотим использовать русские надписи, в данном примере в принципе не нужен
60 sys.setdefaultencoding('utf-8') # то же
61 app = QtGui.QApplication(sys.argv) # определяем для системы нашу программу,
        считайте пока формальностью
62 mw = MainWindow() # создаем переменную и вызываем наш класс главного окна
63 mw.resize(300, 150) # изменяем размеры нашего окна по нашему желанию
64 mw.show() # показываем наше окно пользователю, нужная формальность
65 sys.exit(app.exec_()) # говорим что при закрытии главного окна прекратить
        выполнение кода, необходимая формальность

```

Listing 13: Пример GUI-программы.

## 6.3 Основные компоненты.

Далее по плану — основные компоненты. Все это более подробно описано в книге Прохоренко, а так же можно погуглить например на habrahabr, но подробнее и лучше все это раскрыто на английском, так что гугл и правильные запросы вам в помощь. Ну а мы продолжаем.

### 1. Типы окон и их части.

Как мы видели из предыдущего примера как таковое «окно» можно и не определять — достаточно взять один из виджетов за главный и строить все относительно него. Но если хотим идти в ногу со временем и создавать что-нить более-менее приличное, нужно и общепринятое главное окно с панельками, менюшками, и т.п.... Видов их существует прилично, мы же остановимся на самом каноничном и распространенном: QMainWindow. Далее в процессе обучения мы будем пилить проект для микроскопа, попутно раскрывая какие-то аспекты написания GUI. Итак давай те ка создадим основу нашей программы. Лицо нашего софта — цикл программы и главное окно.

```

1  # -*- coding: utf-8 -*-
2
3  from PyQt4 import QtCore, QtGui
4  import sys
5
6
7  # создаем свой класс для рисования главного окна нашей программы
8  class MainWindow(QtGui.QMainWindow):
9
10     def __init__(self):

```

```

11     QtGui.QMainWindow.__init__(self, parent=None)
12     desktop = QtGui.QApplication.desktop()
13     x = desktop.width()
14     y = desktop.height()
15     self.move(x / 2, y / 2)
16     self.setWindowTitle("Project GUI micro")
17
18     self.centralwidget = QtGui.QWidget(self)
19     # Здесь обычно идет определение всех других виджетов и все они дб..
    инстанцированы в self.centralwidget
20
21     self.setCentralWidget(self.centralwidget) # запускаем""
    центральный виджет
22     self.menubar = self.menuBar() # создаем верхнее меню там( где
    файл, помощь и прочее)
23     self.setMenuBar(self.menubar)
24     self.statusbar = QtGui.QStatusBar(self) # создаем строку
    статуса
25     self.setStatusBar(self.statusbar)
26     self.statusBar().showMessage('Ready for action')
27
28 reload(sys)
29 sys.setdefaultencoding('utf-8')
30 app = QtGui.QApplication(sys.argv)
31 mw = MainWindow()
32 mw.resize(300, 150)
33 mw.show()
34 sys.exit(app.exec_())

```

Listing 14: Класс основного окна.

## 2. Слои для компоновки виджетов.

Мы с вами будем пользоваться вертикальными и горизонтальными коробками, а именно: `QtGui.QVBoxLayout()`, `QtGui.QHBoxLayout()`.

С помощью этих малышей можно вполне себе запрогать все что душе угодно. По нашим примерам вы увидите как. Для всего остального, включая любопытство, есть интернет и книги.

Все прочие основные виджеты вы посмотрите на примерах или прочитаете в книжках/интернетах. На первых порах все в принципе логично и просто (без этого язык программирования не был бы языком программирования в современном понимании). Есть родители (глобальные множества методов и объектов) и их дети (конкретные вещи, которыми мы хотим оперировать в рамках нашей программы). Чтобы указать языку какой объект мы хотим использовать важно так или иначе указывать наследование, как правило перечисление через точку или ссылаться на предка в скобках (если мы говорим о классах). Все остальное на примерах.

**Пример 6.1.** *Создадим скелет для программы под микроскоп:*

```

1 # -*- coding: utf-8 -*-
2
3 from PyQt4 import QtCore, QtGui
4 import sys

```

```

5
6
7 class MainWidget( QtGui.QMainWindow ):
8     # Создаем список со словарями для строки меню. Так удобнее будет создавать
объекты меню
9     list_menu = [{ 'name': u'Файл',
10                    'type': 'menu',
11                    'multiply': 0,
12                    'split_line': 0},
13                  { 'name': u'Помощь',
14                    'type': 'menu',
15                    'multiply': 0,
16                    'split_line': 0},
17                  { 'name': u'Информация',
18                    'type': 'menu',
19                    'multiply': 0,
20                    'split_line': 0},
21                  { 'name': u'Открыть изображение',
22                    'type': 'action',
23                    'menu': 0,
24                    'multiply': 0,
25                    'split_line': 0},
26                  { 'name': u'Сохранить изображение',
27                    'type': 'action',
28                    'menu': 0,
29                    'multiply': 0,
30                    'split_line': 1},
31                  { 'name': u'Сгенерировать отчет',
32                    'type': 'action',
33                    'menu': 0,
34                    'multiply': 0,
35                    'split_line': 0},
36                  { 'name': u'Краткое руководство по микроскопу',
37                    'type': 'action',
38                    'menu': 1,
39                    'multiply': 0,
40                    'split_line': 0},
41                  { 'name': u'Краткое руководство по программе',
42                    'type': 'action',
43                    'menu': 1,
44                    'multiply': 0,
45                    'split_line': 0},
46                  { 'name': u'Авторы программы',
47                    'type': 'action',
48                    'menu': 2,
49                    'multiply': 0,
50                    'split_line': 0}]
51 def __init__( self ):
52     QtGui.QMainWindow.__init__( self, parent=None )
53     desktop = QtGui.QApplication.desktop()
54     x = desktop.width()
55     y = desktop.height()
56     self.move( x / 2, y / 2 )
57     self.setWindowTitle( "Project GUI micro" ) # Создаем заголовок окна
58     self.centralwidget = QtGui.QWidget( self )
59     self.vl_main = QtGui.QVBoxLayout( self.centralwidget )

```

```

60     self.cb_camera = QtGui.QComboBox() # Создаем раскрывающийся
выпадающий() список для выбора камеры, если вебкамер несколько
61     self.cb_camera.addItem('0', '1', '2', '3') # добавляем
элементы в список
62     self.vl_main.addWidget(self.cb_camera)
63     self.label = QtGui.QLabel() # создаем лэйбл для отображения видео
или картинки
64     self.vl_main.addWidget(self.label)
65     self.hl_button = QtGui.QHBoxLayout()
66     self.button_start = QtGui.QPushButton('Start video')
67     self.button_start.setCheckable(True)
68     self.button_save = QtGui.QPushButton('Save picture')
69     self.hl_button.addWidget(self.button_start)
70     self.hl_button.addWidget(self.button_save)
71     self.vl_main.addLayout(self.hl_button)
72     self.setCentralWidget(self.centralwidget) # устанавливаем ""
центральный виджет
73     self.menubar = self.menuBar() # создаем верхнее меню там( где
файл, помощь и прочее)
74     for i in range(len(self.list_menu)): # создаем цикл для
назначения каждого элемента меню, свойства которых мы прописали в словаре
75         if self.list_menu[i]['type'] == 'menu': # если объект
является заголовком и не несет в себе сигнала" к действию"
76             self.menu = QtGui.QMenu(self.list_menu[i]['name'], self
.menubar) # создаем объектэлемент- меню
77             self.list_menu[i]['type'] = self.menu # сохраняем объект
в наш словарь, переписав значение ключа "type"
78             self.menubar.addMenu(self.menu) # добавляем элемент в
меню
79             elif self.list_menu[i]['type'] == 'action': # Если объект
является действием", те.. при нажатии на него должно чтото- произойти
в программе
80                 self.action = QtGui.QAction(self.list_menu[i]['name'],
self.menubar)
81                 self.list_menu[i]['type'] = self.action
82                 self.list_menu[self.list_menu[i]['menu']]['type'].
addAction(self.action)
83                 if self.list_menu[i]['split_line']: # добавляем
разделительную линию если нужно
84                     self.list_menu[self.list_menu[i]['menu']]['type'].
addSeparator()
85                     self.setMenuBar(self.menubar)
86                     self.statusbar = QtGui.QStatusBar(self) # создаем строку статуса
87                     self.setStatusBar(self.statusbar)
88                     self.statusBar().showMessage('Ready for action')
89
90
91 reload(sys)
92 sys.setdefaultencoding('utf-8')
93 app = QtGui.QApplication(sys.argv)
94 mw = MainWidget()
95 mw.resize(300, 150)
96 mw.show()
97 sys.exit(app.exec_())

```

Listing 15: Програма управління мікроскопом.

Теперь нужно задать систему сигналов и слотов чтобы «оживить» нашу программу.

```
1 # -*- coding: utf-8 -*-
2
3 from PyQt4 import QtCore, QtGui
4 import sys
5 from SimpleCV import *
6
7
8 class MainWidget(QtGui.QMainWindow):
9     webcam = None
10     is_start = False
11     list_menu = [{ 'name': u'Файл',
12                    'type': 'menu',
13                    'multiply': 0,
14                    'split_line': 0},
15                  { 'name': u'Помощь',
16                    'type': 'menu',
17                    'multiply': 0,
18                    'split_line': 0},
19                  { 'name': u'Информация',
20                    'type': 'menu',
21                    'multiply': 0,
22                    'split_line': 0},
23                  { 'name': u'Сохранить изображение',
24                    'type': 'action',
25                    'menu': 0,
26                    'multiply': 0,
27                    'split_line': 1},
28                  { 'name': u'Сгенерировать отчет',
29                    'type': 'action',
30                    'menu': 0,
31                    'multiply': 0,
32                    'split_line': 0},
33                  { 'name': u'Краткое руководство по микроскопу',
34                    'type': 'action',
35                    'menu': 1,
36                    'multiply': 0,
37                    'split_line': 0},
38                  { 'name': u'Краткое руководство по программе',
39                    'type': 'action',
40                    'menu': 1,
41                    'multiply': 0,
42                    'split_line': 0},
43                  { 'name': u'Авторы программы',
44                    'type': 'action',
45                    'menu': 2,
46                    'multiply': 0,
47                    'split_line': 0}]
48     def __init__(self, parent=None):
49         QtGui.QMainWindow.__init__(self, parent=None)
50         desktop = QtGui.QApplication.desktop()
51         x = desktop.width()
52         y = desktop.height()
53         self.move(x / 2, y / 2)
```



```

54     self.setWindowTitle("Project GUI micro")
55     self.centralwidget = QtGui.QWidget(self)
56     self.vl_main = QtGui.QVBoxLayout(self.centralwidget)
57     self.cb_camera = QtGui.QComboBox()
58     self.cb_camera.addItem('0')
59     self.cb_camera.addItem('1')
60     self.cb_camera.addItem('2')
61     self.cb_camera.addItem('3')
62     self.vl_main.addWidget(self.cb_camera)
63     self.label = QtGui.QLabel()
64     self.vl_main.addWidget(self.label)
65     self.hl_button = QtGui.QHBoxLayout()
66     self.button_start = QtGui.QPushButton('Start video')
67     self.button_start.setCheckable(True)
68     self.button_save = QtGui.QPushButton('Save picture')
69     self.hl_button.addWidget(self.button_start)
70     self.hl_button.addWidget(self.button_save)
71     self.vl_main.addLayout(self.hl_button)
72     self.setCentralWidget(self.centralwidget)
73     self.menubar = self.menuBar()
74     self.set_menu() # Вынесем назначения элементов меню в отдельную
75     # функцию для лучшей читаемости кода
76     self.setMenuBar(self.menubar)
77     self.statusbar = QtGui.QStatusBar(self)
78     self.setStatusBar(self.statusbar)
79     self.statusBar().showMessage('Ready for action')
80     # Назначим действия для каждой кнопки
81     QtCore.QObject.connect(self.button_start, QtCore.SIGNAL('
82     toggled(bool)'), lambda: self.start_video(x))
83     QtCore.QObject.connect(self.button_save, QtCore.SIGNAL('
84     clicked
85     ()'), self.save_pict)
86     def set_menu(self):
87         # Назначаем элементы главного меню
88         for i in range(len(self.list_menu)):
89             if self.list_menu[i]['type'] == 'menu':
90                 self.menu = QtGui.QMenu(self.list_menu[i]['name'], self
91                 .menubar)
92                 self.list_menu[i]['type'] = self.menu
93                 self.menubar.addMenu(self.menu)
94             elif self.list_menu[i]['type'] == 'action':
95                 self.action = QtGui.QAction(self.list_menu[i]['name'],
96                 self.menubar)
97                 # Назначаем действие для каждого активного элемента меню
98                 if i == 3:
99                     self.action.triggered.connect(self.save_pict)
100                 elif i == 4:
101                     pass
102                 elif i == 5:
103                     text1 = u'''Микроскоп состоит из вебкамеры с
104                     перевернутой оптикой и корпуса с подвижной
105                     верхней плитой. Путем регулирования расстояния до вебкамеры с помощью 4
106                     винтов находим оптимальный
107                     фокус до изучаемого объекта. Четкость изображения оцениваем визуально по
108                     получаемой
109                     картинке.'''
110                     header1 = self.list_menu[i]['name']
111                     self.action.triggered.connect(lambda: self.inform(
112                     header1, text1))
113                 elif i == 6:

```

```

101         text2 = u'''Программа предназначена для получения видео
102 с веб камеры и вывода его на экран или сохранения
103 фотографии изучаемого объекта. Для старта видеопотока— используется кнопка '
104 Start video', для
105 сохранения изображения используется кнопка 'Save picture'. '''
106         header2 = self.list_menu[i][ 'name' ]
107         self.action.triggered.connect(lambda: self.inform(
108 header2, text2))
109         elif i == 7:
110             text3 = u'''Авторы определяться позднее=)'''
111             header3 = self.list_menu[i][ 'name' ]
112             self.action.triggered.connect(lambda: self.inform(
113 header3, text3))
114         else:
115             self.action.triggered.connect(QtGui.QApplication.
116 quit)
117             self.list_menu[i][ 'type' ] = self.action
118             self.list_menu[self.list_menu[i][ 'menu' ]][ 'type' ].
119 addAction(self.action)
120             if self.list_menu[i][ 'split_line' ]:
121                 self.list_menu[self.list_menu[i][ 'menu' ]][ 'type' ].
122 addSeparator()
123
124     def save_pict(self):
125         # Данная функция используется для сохранения изображения
126         # Вызываем диалог в котором пользователю нужно выбрать место и имя
127         файла для сохранения
128         filename = QtGui.QFileDialog.getSaveFileName(None, 'Save
129 picture', filter=QtCore.QString( 'PNG (*.png) '))
130         if filename is None: # Если пользователь не выбрал куда сохранить
131             файл
132             return # То завершаем работу функции
133         filename = str(filename.toLocal8Bit()) + '.png' # Для работы с
134 русской локалью и добавляем нужное разрешение
135         if self.is_start: # Если запущено видео, сохраняем последний кадр
136             self.cam_image.save(filename)
137         else:
138             if not self.webcam: # Если вебкамера— не использовалась ранее,
139 то запускаем ее
140                 self.webcam = Camera(self.cb_camera.currentIndex(), {
141 "width": 640, "height": 480})
142             else:
143                 pass
144             image = self.webcam.getImage() # Снимаем изображение с камеры
145             image.save(filename) # Сохраняем изображение по выбранному
146 пути
147     def start_video(self, is_toggle):
148         # Данная функция используется для запуска видеосъемки и отображении ее
149 в виде QLabel
150         if self.button_start.isChecked(): # Если кнопка в нажатом
151 состоянии, то запускаем видеопоток в QLabel
152             self.is_start = True # Ставим общий флаг"" что поток видео
153 запущен, данный флаг можно использовать в других частях кода
154             if not self.webcam: # Если вебкамера— не определена раньше —
155 определяем
156                 number_cam = self.cb_camera.currentIndex()

```

```

139         self.webcam = Camera(number_cam, {"width": 640, "height": 480})
140     else:
141         pass
142         self.timer = QtCore.QTimer() # Создаем таймер
143         # Связываем таймер с обновлением картинки, каждую секунду
144         обновляем изображение на лейбле
145         QtCore.QObject.connect(self.timer, QtCore.SIGNAL('timeout()'), self.show_frame)
146         self.timer.start(1) # Запускаем таймер
147     else: # Иначе — останавливаем таймер, а значит остановиться и
148         поток обновления лейбла
149         self.is_start = False # Отключаем "" флаг потока
150         self.timer.stop()
151
152     def show_frame(self):
153         # Функция для внятия изображения с камеры и передачи его в лейбл
154         ipl_image = self.webcam.getImage() # Снимаем изображение с камеры
155         # Далее технические вещи, для корректного конвертирования
156         изображения...
157         ipl_image.dl().circle((150, 75), 50, Color.RED, filled=True)
158         data = ipl_image.getBitmap().toString()
159         image = QtGui.QImage(data, ipl_image.width, ipl_image.height, 3
160         * ipl_image.width, QtGui.QImage.Format_RGB888)
161         pixmap = QtGui.QPixmap() # Понятный Qt формат изображения
162         pixmap.convertFromImage(image.rgbSwapped())
163         self.cam_image = pixmap.scaled(1000, 500) # Можем изменять
164         длинуиширину* изображения для отображения
165         self.label.setPixmap(pixmap.scaled(1000, 500)) # Ставим
166         изображения в лейбл
167         QtGui.QApplication.processEvents() # Запускаем оборот основного
168         цикла чтобы приложение не "зависало"
169     def inform(self, header, text):
170         # Функция для отображения окон информации
171         QtGui.QMessageBox.information(None, header, text)
172
173     reload(sys)
174     sys.setdefaultencoding('utf-8')
175     app = QtGui.QApplication(sys.argv)
176     mw = MainWidget()
177     mw.resize(300, 150)
178     mw.show()
179     sys.exit(app.exec_())

```

Listing 16: Программа управления микроскопом. Рабочая версия.

**Задача 6.1.** Ну и теперь вам остается доделать несколько вещей.

1. Почистить код, устранить баги и оптимизировать параметры. Не бойтесь экспериментировать. Создайте программу удобную для вас и отправьте мне файл=).
2. Русифицируйте программу. Или наоборот — сделайте ее полностью англо-язычной. Как вам интереснее.

3. Попробуйте сделать так, чтобы пользователь мог менять/задавать изначально величину снимаемого изображения.
4. Попробуйте запустить все это под другой операционной системой=), если есть возможность.

## 7 Постановка эксперимента

Теперь когда мы написали программу для управления веб-камерой и собрали микроскоп, давайте поставим наш химико-физический эксперимент. Нам потребуются следующие осадки в количестве 1 г.

1. Йодид свинца
2. Хлорид кадмия
3. Сульфид меди

**Задача 7.1.** Напишите уравнения химических реакций которыми мы можем воспользоваться для получения осадков. Рассчитайте массу реагентов.

Ход эксперимента.

1. После получения осадков нам потребуется высушить осадки на фильтровальной бумаге. Достаточно несколько раз промокнуть их другой фильтровальной бумагой чтобы убрать избыток влаги.
2. Затем возьмем небольшой кусочек прозрачного скотча и прикоснемся клейкой стороной к осадку.
3. Заклеим клейкую сторону с осадком другим кусочком скотча.
4. Изучим получившуюся систему под микроскопом и сохраним фотографии с помощью написанного нами программного обеспечения.