

Федеральное государственное образовательное бюджетное учреждение  
высшего профессионального образования  
**«Финансовый университет при Правительстве Российской Федерации»**  
**(Финансовый университет)**  
**Тульский филиал**

**Кафедра «Математика и информатика»**

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ  
ПО РАБОТЕ НА ПРАКТИЧЕСКИХ ЗАНЯТИЯХ  
ПО ДИСЦИПЛИНЕ  
ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ**

**Для направления подготовки бакалавров:**

**38.03.05 «Бизнес-информатика»**

**Тула 2015**

**В рамках выполнения практического задания студенту необходимо выполнить соответствующие задания. (n – номер по списку в журнале)**

Все задания оформляются в документе word.

Тит лист

Упражнение 1...

Упражнение 2 ...

.....

В каждом упражнении указывается:

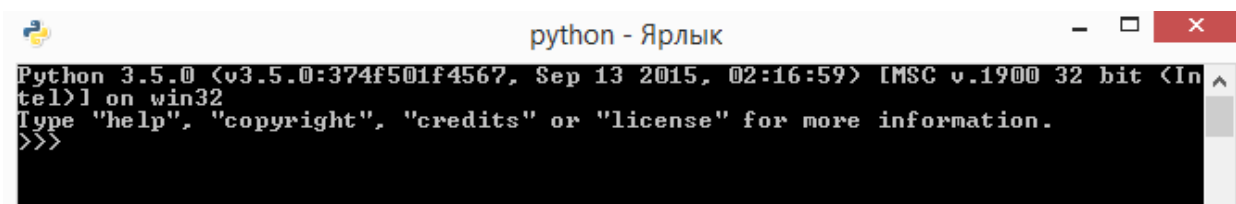
Задание: ...

Решение: включает текст программы, ее скриншот и результат выполнения также в виде скриншота, если не указано другое в упражнении

## **Практическое задание № 1**

Знакомство с Python

Для диалога с Python нам необходимо прежде всего запустить интерпретатор Python. Нажмите на соответствующий ярлык на рабочем столе.



Диалоговое окно позволяет вводить команды построчно

Перейдем к диалогу:

Напечатайте:

`print ('Доброе утро')`

В результате должно получиться:

```
>>> print ('доброе утро')
доброе утро
>>> _
```

Замените апострофы на кавычки и тройные кавычки, в результате ничего не изменится:

```
>>> print ('доброе утро')
доброе утро
>>> print ("Доброе утро")
Доброе утро
>>> print("""Доброе утро""")
Доброе утро
>>>
```

**Упражнение 1.** Напечатайте: Сегодня 8 марта

Введите следующие строки:

```
print ("Сегодня", 8, "марта")
print ("Завтра "+"будет", 8+1, "марта")
print ("Послезавтра будет", 8+2, "марта")
```

Посмотрите, что будет в интерпретаторе. Сделайте скриншот

**Упражнение 2.** Сложите  $n$  столов и  $2n$  стульев

В интерпретаторе должно получиться в одной строке:  $n$  столов +  $2n$  стульев равняется  $X$  предметов мебели

## Работа с Python как с калькулятором

Для справки:

При использовании стандартных функций необходимо подключить модуль math:

```
from math import *
```

Основные арифметические операции приведены в таблице

$x + y$	Сложение
$x - y$	Вычитание
$x * y$	Умножение
$x / y$	Деление
$x // y$	Получение целой части от деления
$x \% y$	Остаток от деления
$-x$	Смена знака числа
$\text{abs}(x)$	Модуль числа
$\text{divmod}(x, y)$	Пара ( $x // y, x \% y$ )
$x ** y$	Возведение в степень
$\text{pow}(x, y[, z])$	$x^y$ по модулю (если модуль задан)

**math.ceil(X)** – округление до ближайшего большего числа.

**math.copysign(X, Y)** - возвращает число, имеющее модуль такой же, как и у числа X, а знак - как у числа Y.

**math.fabs(X)** - модуль X.

**math.factorial(X)** - факториал числа X.

**math.floor(X)** - округление вниз.

**math.fmod(X, Y)** - остаток от деления X на Y.

**math.frexp(X)** - возвращает мантиссу и экспоненту числа.

**math.ldexp(X, I)** -  $X * 2^I$ . Функция, обратная функции **math.frexp()**.

**math.fsum(последовательность)** - сумма всех членов последовательности.

Эквивалент встроенной функции **sum()**, но **math.fsum()** более точна для чисел с плавающей точкой.

**math.isfinite(X)** - является ли X числом.

**math.isnan(X)** - является ли X NaN (Not a Number - не число).  
**math.modf(X)** - возвращает дробную и целую часть числа X. Оба числа имеют тот же знак, что и X.  
**math.trunc(X)** - усекает значение X до целого.  
**math.exp(X)** -  $e^X$ .  
**math.expm1(X)** -  $e^X - 1$ . При  $X \rightarrow 0$  точнее, чем **math.exp(X)-1**.  
**math.log(X, [base])** - логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм.  
**math.log1p(X)** - натуральный логарифм (1 + X). При  $X \rightarrow 0$  точнее, чем **math.log(1+X)**.  
**math.log10(X)** - логарифм X по основанию 10.  
**math.log2(X)** - логарифм X по основанию 2. Новое в [Python 3.3](#).  
**math.pow(X, Y)** -  $X^Y$ .  
**math.sqrt(X)** - квадратный корень из X.  
**math.acos(X)** - арккосинус X. В радианах.  
**math.asin(X)** - арксинус X. В радианах.  
**math.atan(X)** - арктангенс X. В радианах.  
**math.atan2(Y, X)** - арктангенс Y/X. В радианах. С учетом четверти, в которой находится точка (X, Y).

**math.cos(X)** - косинус X (X указывается в радианах).  
**math.sin(X)** - синус X (X указывается в радианах).  
**math.tan(X)** - тангенс X (X указывается в радианах).  
**math.hypot(X, Y)** - вычисляет гипотенузу треугольника с катетами X и Y (**math.sqrt(x \* x + y \* y)**).  
**math.degrees(X)** - конвертирует радианы в градусы.  
**math.radians(X)** - конвертирует градусы в радианы.  
**math.cosh(X)** - вычисляет гиперболический косинус.  
**math.sinh(X)** - вычисляет гиперболический синус.  
**math.tanh(X)** - вычисляет гиперболический тангенс.  
**math.acosh(X)** - вычисляет обратный гиперболический косинус.  
**math.asinh(X)** - вычисляет обратный гиперболический синус.  
**math.atanh(X)** - вычисляет обратный гиперболический тангенс.  
**math.erf(X)** - функция ошибок.  
**math.erfc(X)** - дополнительная функция ошибок (1 - **math.erf(X)**).  
**math.gamma(X)** - гамма-функция X.  
**math.lgamma(X)** - натуральный логарифм гамма-функции X.  
**math.pi** -  $\pi = 3,1415926...$   
**math.e** -  $e = 2,718281..$

### Упражнение 3

Составьте выражение на Python, которое вычисляет стоимость покупки яблок, мандаринов и бананов, если известны их вес и цена за килограмм.

Результат представить в виде скриншота, где печатается соответствующая цена

**Упражнение 4** Вычислите выражения сделайте скриншот, результат представьте в виде таблицы

Выражение	Результат выполнения

$$\frac{2^{-9} \cdot 2^7}{2^{-4}}, (\sqrt{17} - \sqrt{12})(\sqrt{17} + \sqrt{12}), \frac{(2\sqrt{6})^2}{25}, (\sqrt{2\frac{4}{7}} - \sqrt{7\frac{1}{7}}) : \sqrt{\frac{2}{63}},$$

$$h(10 + x) + h(10 - x), \text{ если } h(x) = \sqrt[5]{x} + \sqrt[5]{x - 20}. \quad (x=n),$$

$$1 + \frac{2}{3 + \frac{4}{5 + \frac{6}{7 + x}}}$$

$$1 + \sin^2(n) * 3^3 + \arctg(8 \cdot \pi / 3)$$

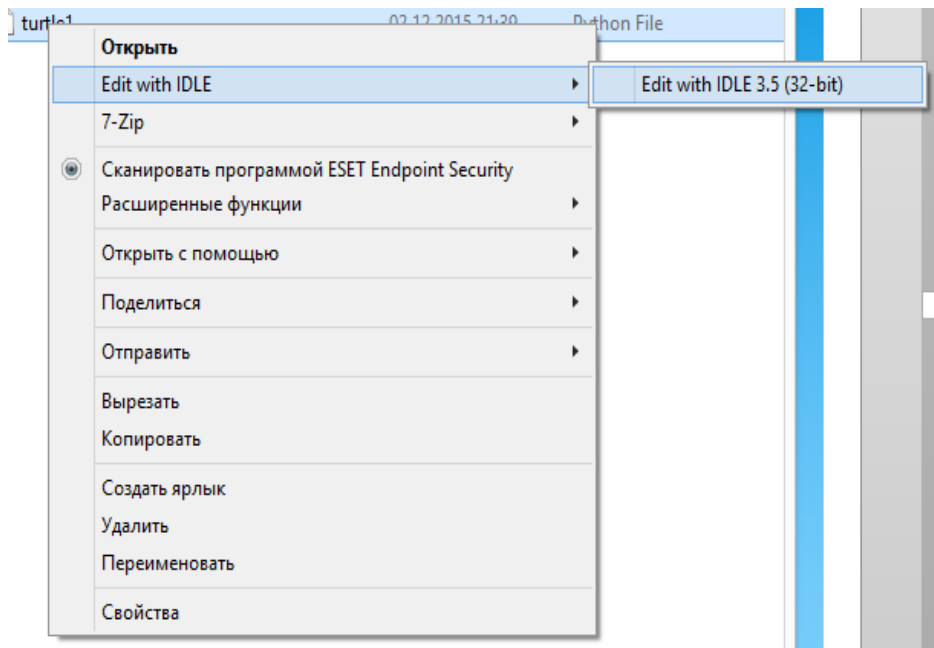
$$\frac{\ln^2(n^3)}{1 + n^2} \bmod(2n) + 4 \cos^2(n) + \operatorname{arcc}tg(9 \cdot \pi / 2)$$

$$\frac{2^n(n^3)}{n^3 + n^2} \operatorname{div}(2n) + 4 \cos^2(n) \operatorname{tg}(n + 6), \text{ округлить до целых}$$

$$\sqrt{n^3 + n^2} / \frac{2^n(n^3)}{3^4} * (3n) + 4 \cos^2(n) \operatorname{tg}(n + 6) \bmod(2), \text{ взять остаток}$$

## Практическое задание №2 Работа в IDLE

Создайте файл с расширением `myprogram1.py` через текстовый редактор. Откройте его через среду IDLE.

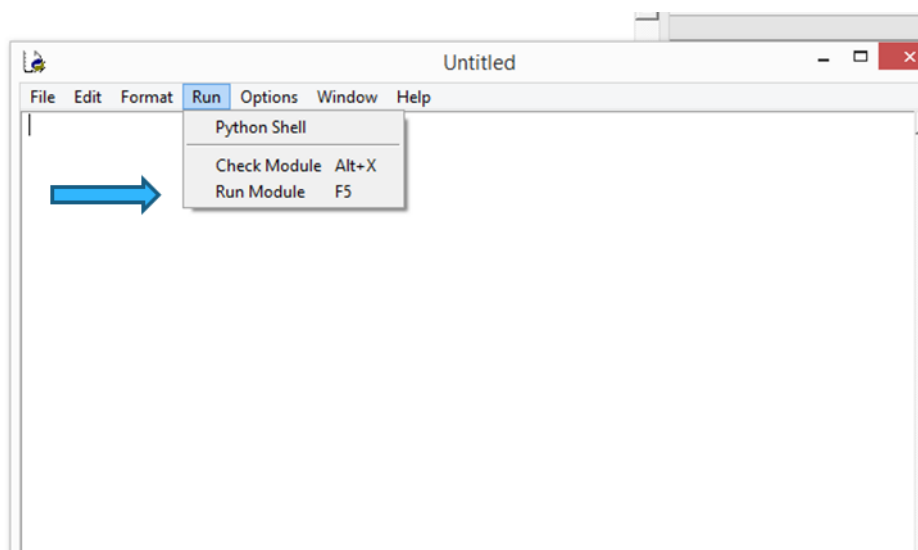


## Упражнение 5

Введите следующий код программы

```
print ("Hello world!")  
print ("Здравствуй мир!")  
print ('Как настроение?')  
input ()  
print ('Могло бы быть и хуже')
```

Запустите выполнение программы как показано на следующем рисунке или нажав F5.



Проявите творчество. Сделайте скриншот.

## Упражнение 6

Составьте арифметические выражения для вычисления следующих величин:

1.  $n$ -е четное число (первым считается 2, вторым 4 и т.д.)
2.  $n$ -е нечетное число (первое равно 1, второе 3 и т.д.)
3. В очереди стоят  $n$  людей, сколько человек находится между  $i$ -м и  $k$ -м в очереди.
4. Сколько нечетных чисел на отрезке  $(a, b)$ , если  $a$  и  $b$  – четные? Если  $a$  и  $b$  – нечетные?  $a$  – четное,  $b$  – нечетное?
5. Сколько полных минут и часов содержится в  $x$  секундах?
6. В доме 9 этажей, на каждом этаже одного подъезда по 4 квартиры. В каком подъезде, и на каком этаже находится  $n$ -я квартира.
7. Старинными русскими денежными единицами являются: 1 рубль – 100 копеек, 1 гривна — 10 копеек, 1 алтын — 3 копейки, 1 полушка — 0,25 копейки. Имеется  $A$  копеек. Запишите выражения для представления имеющейся суммы в рублях, гривнах, алтынах и полушках.
8. Стрелка прибора вращается с постоянной скоростью, совершая  $w$  оборотов в секунду (не обязательно стрелка прибора, может быть это волчок в игре «Что? Где? Когда?» и т.п.) Угол поворота стрелки в нулевой момент времени примем за 0. Каков будет угол поворота через  $t$  секунд?
9. Вы стоите на краю дороги и от вас до ближайшего фонарного столба  $x$  метров. Расстояние между столбами  $y$  метров. На каком расстоянии от вас находится  $n$ -й столб?
10. Та же ситуация, что и в предыдущей задаче. Длина вашего шага  $z$  метров. Мимо скольких столбов вы пройдете, сделав  $n$  шагов.

Ввод переменных осуществляется через среду IDLE

Пример решения 1-ой задачи приведен на скриншоте, нижняя часть – результат выполнения



```
File Edit Format Run Options Window Help
import math
print ('Программа вычисления n-го четного числа')
print ('Введите число')
n=int(input())
a=2*n
print ("Четное число с номером ",n," составит ",a, " единиц")

Ln: 9 Col: 0

Программа вычисления n-го четного числа
Введите число
5
Четное число с номером 5 составит 10 единиц
>>>
===== RESTART: D:/Рома/Работа с python/upr6.py =====
Программа вычисления n-го четного числа
Введите число
10
Четное число с номером 10 составит 20 единиц
>>>
```

## Практическое задание №3

### Упражнение 7

#### Составление логических выражений, условный оператор

1. Напишите программу, которая запрашивает два числа и, а затем выводит их в порядке возрастания, сначала меньшее затем большее.
2. Создайте программу, которая запрашивает у пользователя три числа, а затем сообщает ему, какое из этих чисел наибольшее.
3. Создайте программу, которая запрашивает у пользователя число и сообщает, является ли это число четным.
4. Даны три числа  $a$ ,  $b$ ,  $c$ . Если среди них есть отрицательные, возведите их в квадрат. Если после возведения в квадрат число стало больше 20, умножьте его на 2.
5. Напишите программу, которая запрашивает значение  $x$ , а затем выводит значение следующей функции от  $x$ :

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

6. Напишите программу, которая запрашивает значения  $x, y, z$ , а затем выводит значение следующих функций:  $\max(x, \min(y, z))$  и  $\min(\min(x, y), z)$ .

7. Модифицируйте вашу программу расчета корней квадратного уравнения, добавив к ней проверку неотрицательности дискриминанта. Если дискриминант отрицательный, сообщайте пользователю, что уравнение не имеет корней.

8. Модифицируйте вашу программу расчета корней квадратного уравнения, добавив к ней проверку того, что первый коэффициент не равен нулю. В противном случае сообщайте пользователю, что уравнение не квадратное, а линейное и вычислите его единственный корень. Если первые два коэффициента оба равны нулю, а третье не равно, сообщите пользователю, что корней нет. А если все коэффициенты равны нулю, сообщите, что любое число является корнем.

9. Пользователь вводит три числа. Сообщите ему, упорядочены ли введенные числа по возрастанию.

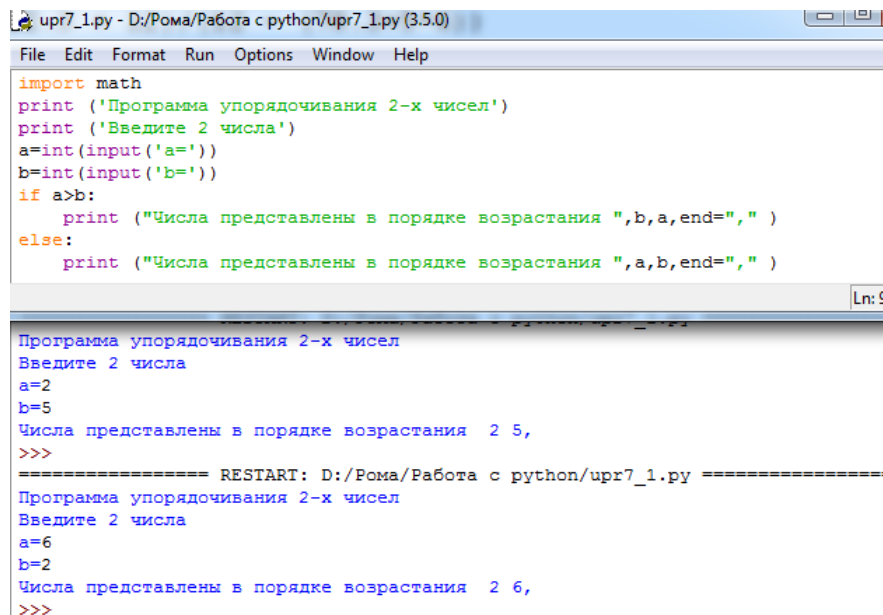
10. Пользователь вводит три числа – длины сторон треугольника. Программа должна сообщить пользователю:

является ли треугольник равносторонним; является ли треугольник равнобедренным; является ли треугольник разносторонним; является ли треугольник прямоугольным; существует ли вообще такой треугольник (такого треугольника не может быть, если сумма любых двух сторон окажется меньше третьей стороны).

11. «Узник замка Иф» За многие годы заточения узник замка Иф проделал вилкой в стене прямоугольное отверстие размером  $d \times e$ . Замок Иф сложен из кирпичей размером  $a \times b \times c$ . Узник хочет узнать, сможет ли он выбрасывать кирпичи в море из этого отверстия, чтобы сделать подкоп. Снабдите его необходимым для решения задачи софтом. На вход программе подаются 5 чисел ( $a, b, c, d, e$ ), программа должна давать ответ YES или NO.

12. Напишите программу, которая в зависимости от введенного возраста добавляет слова «год», «года» или «лет». Например, при вводе возраста 1, программа сообщает «1 год», при числе 2 – «2 года», при числе 125 – «125 лет».

Пример решения 1-ой задачи приведен на скриншоте, нижняя часть – результат выполнения.



```
upr7_1.py - D:/Рома/Работа с python/upr7_1.py (3.5.0)
File Edit Format Run Options Window Help

import math
print ('Программа упорядочивания 2-х чисел')
print ('Введите 2 числа')
a=int(input('a='))
b=int(input('b='))
if a>b:
    print ("Числа представлены в порядке возрастания ",b,a,end="," )
else:
    print ("Числа представлены в порядке возрастания ",a,b,end="," )

Ln: 9

Программа упорядочивания 2-х чисел
Введите 2 числа
a=2
b=5
Числа представлены в порядке возрастания 2 5,
>>>
===== RESTART: D:/Рома/Работа с python/upr7_1.py =====
Программа упорядочивания 2-х чисел
Введите 2 числа
a=6
b=2
Числа представлены в порядке возрастания 2 6,
>>>
```

## Упражнение 8 Цикл for, приемы накопления суммы и произведения

### 1. Цикл for

1.1. Напечатайте таблицу умножения на 5. предпочтительно печатать  $1 \times 5 = 5$ ,  $2 \times 5 = 10$ , а не просто 5, 10 и т.д.

1.2. Напечатайте в столбик нечетные числа от 3 до 25.

1.3. Напечатайте свое имя так, чтобы оно располагалось в углах окна вывода.

1.4. Выведите на экран таблицу значений синуса от 0 до  $2\pi$ . В каждой строке должны стоять один аргумент и одно значение. Количество значений аргумента пусть задает пользователь.

### 2. Прием накопления суммы

2.1. Напишите программу, которая вычисляет сумму квадратов чисел от 1 до N. Число N программа должна запрашивать у пользователя.

2.2. Напишите программу, перемножающую целые числа без использования операции «\*». Например, при умножении целых чисел  $n \times m$  число m надо сложить само с собой n раз ( $m+m+\dots+m$ ).

2.3. Используя прием накопления суммы, найдите сумму нечетных чисел от 1 до N. Число N программа должна запрашивать у пользователя.

2.4. Выведите на экран последовательность сумм чисел от 1 до n. n меняется от 1 до 10. То есть первые члены последовательности: 1, 3 (1+2), 6 (1+2+3), 10 (1+2+3+4) и т.д.

2.5. Вычислите сумму элементов последовательности сумм из предыдущего задания.

2.6. Найдите сумму 100 синусов от аргументов в диапазоне от 0 до  $2\pi$  (из задачи 1.4).

### 3. Прием накопления произведения

3.1. Факториалом целого числа n (обозначается n!) называется произведение всех целых чисел от 1 до n. Напишите программу вычисления факториала введенного пользователем числа.

3.2. Напишите программу возведения числа в целую степень. Число и степень запрашивайте у пользователя.

### 4. Комбинация обоих приемов

4.1. Используя комбинацию обоих приемов, напишите программу, вычисляющую функцию

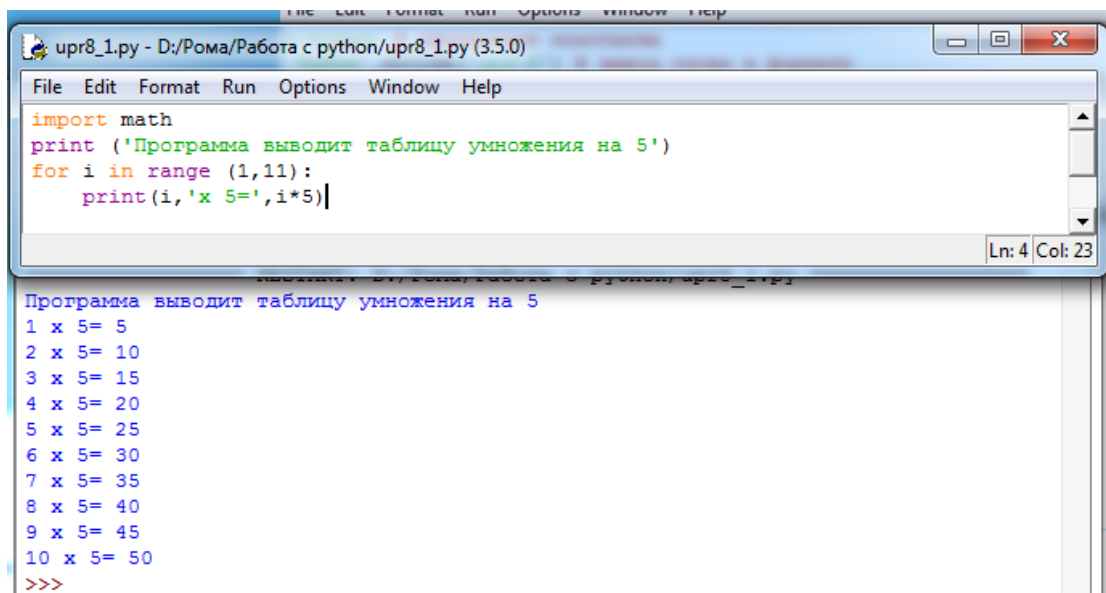
$$1 + x + x^2 + x^3 + \dots + x^{10}.$$

4.2. Вычислите сумму ряда:

$$s_n = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$$

Из мат. анализа известно, что  $\lim_{n \rightarrow \infty} s_n = \exp(x)$ . Выясните, на сколько 5-й и 10-й член последовательности таких сумм отличается от  $\exp(x)$ .

Пример решения 1-ой задачи приведен на скриншоте, нижняя часть – результат выполнения.



```
upr8_1.py - D:/Рома/Работа с python/upr8_1.py (3.5.0)
File Edit Format Run Options Window Help
import math
print ('Программа выводит таблицу умножения на 5')
for i in range (1,11):
    print(i, 'x 5=', i*5)

Ln: 4 Col: 23

Программа выводит таблицу умножения на 5
1 x 5= 5
2 x 5= 10
3 x 5= 15
4 x 5= 20
5 x 5= 25
6 x 5= 30
7 x 5= 35
8 x 5= 40
9 x 5= 45
10 x 5= 50
>>>
```

## Упражнение 9 Цикл while

1. Напечатайте на экране 10 раз слово *Hello* с помощью цикла **while** .
2. Напечатайте в столбик нечетные числа от 3 до 25.
3. Найти минимальное число большее 300, которое делится на 19.
4. Определить, является ли введенное пользователем число степенью тройки (не используя логарифмы).
5. Из мат. анализа известно, что последовательность сумм вида:

$$s_n = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$$

сходится

к

функции  $\cos(x)$ . Напишите программу, которая будет суммировать этот ряд до тех пор, пока очередная добавка не окажется меньше  $10^{-6}$ .

6. Напишите программу, которая запрашивает у пользователя числа до тех пор, пока каждое следующее число больше предыдущего. В конце программа сообщает, сколько чисел было введено.
7. Последовательность Фибоначчи определяется рекуррентным соотношением  $x_{n+1} = x_n + x_{n-1}$ , где  $x_1 = 1, x_2 = 1$ . Найдите первое число в последовательности Фибоначчи, которое больше 1000.

8. Для  $n$ -го члена в последовательности Фибоначчи существует явная формула

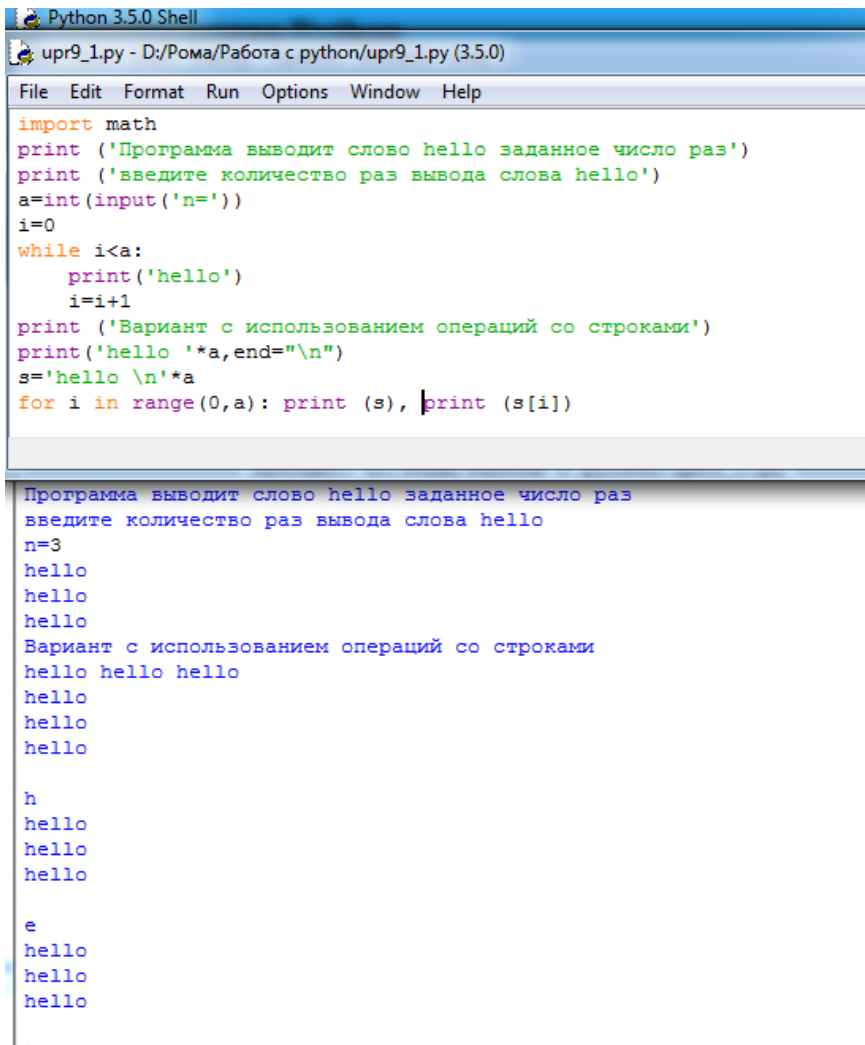
$$x_n = \frac{1}{\sqrt{5}} \left( \left( \frac{1+\sqrt{5}}{2} \right)^n - \left( \frac{1-\sqrt{5}}{2} \right)^n \right)$$

Поскольку операции с вещественными числами происходят с конечной точностью, то, начиная с определенного  $n$ , эта формула будет врать. Найдите  $n$ , начиная с которого, отличие от истинного значения составит 0.001.

9. Создайте программу, играющую с пользователем в орлянку. Программа должна спрашивать у пользователя орел или решка. Если пользователь вводит 0, то выбирает орла, 1 – решку, любое другое число – конец игры. Программа должна вести учет выигрышей и проигрышей и после каждого раунда сообщать пользователю о состоянии его счета. Пусть вначале на счету 1 рубль и ставка в каждом коне тоже 1 рубль. Если денег у пользователя не осталось игра прекращается.

10. Усовершенствуйте разработанный в предыдущем задании «игровой автомат» таким образом, чтобы выигрыш происходил только в 40% случаев.

Пример решения 1-ой задачи приведен на скриншоте, нижняя часть – результат выполнения. Задачу также можно решить при помощи операций со строками



```
Python 3.5.0 Shell
upr9_1.py - D:/Рома/Работа с python/upr9_1.py (3.5.0)
File Edit Format Run Options Window Help

import math
print ('Программа выводит слово hello заданное число раз')
print ('введите количество раз вывода слова hello')
a=int(input('n='))
i=0
while i<a:
    print('hello')
    i=i+1
print ('Вариант с использованием операций со строками')
print('hello '*a,end="\n")
s='hello \n'*a
for i in range(0,a): print (s), print (s[i])

Программа выводит слово hello заданное число раз
введите количество раз вывода слова hello
n=3
hello
hello
hello
Вариант с использованием операций со строками
hello hello hello
hello
hello
hello

h
hello
hello
hello

e
hello
hello
hello
.
```

## Упражнение 10 Функции

### Простейшие функции

1. Создайте функцию, печатающую на экране слово Hello и программу, которая с ее помощью напечатает слово Hello 10 раз.

### Локальные переменные

2. Создайте функцию, которая выводит в одну строку слово Hello 5 раз. Осуществите это с помощью цикла. Переменную счетчик цикла сделайте локальной. С помощью этой процедуры выведите 5 таких строк.

### Параметры функцию

3. Создайте процедуру, печатающую слово Hello заданное число раз. Количество раз передавайте в процедуру как параметр-значение.
4. Создайте процедуру, увеличивающую значение переменной на единицу.

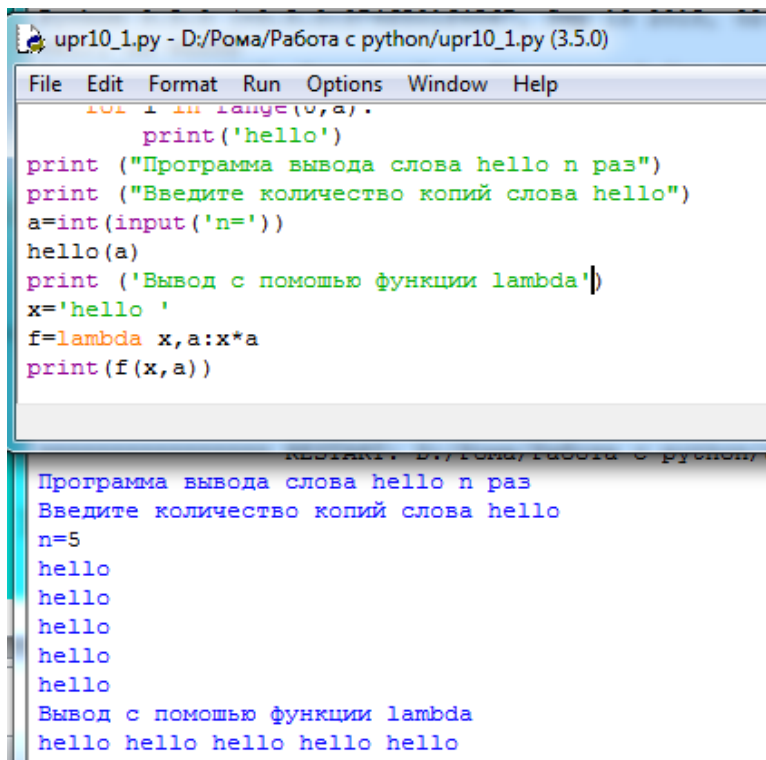
5. Создайте функцию, меняющую значения двух переменных местами.
6. Создайте функцию, располагающую два числа по возрастанию. На входе она должна получать две переменные и если первая больше второй, то значения должны поменяться местами. Операцию перестановки выполняйте, обращаясь к процедуре из первой задачи.
7. Создайте функцию, упорядочивающую три числа по возрастанию. Сравнений в этой функции не проводите. Вместо этого обращайтесь к функции из предыдущего задания.
8. Создайте функцию, которая пару последовательных чисел Фибоначчи преобразует в следующую пару. То есть, если на входе даны элементы с номерами  $(n-1)$  и  $n$ , то процедура должна в те же переменные записать элементы с номерами  $n$  и  $(n+1)$ . С помощью этой функции найдите 10-е число в последовательности Фибоначчи.
9. Создайте функцию для возведения числа в целую степень. Число и степень должны быть параметрами-значениями, а результат должен записываться в параметр-переменную.
10. Поворот на угол  $\varphi$  против часовой стрелки относительно начала координат приводит к следующему преобразованию координат:

$$\begin{cases} x' = x \cos \varphi - y \sin \varphi \\ y' = x \sin \varphi + y \cos \varphi \end{cases}$$

Создайте функцию, осуществляющую такое преобразование.

Пример решения 1-ой задачи приведен на скриншоте, нижняя часть – результат выполнения.





The screenshot shows a Python IDE window titled 'upr10\_1.py - D:/Рома/Работа с python/upr10\_1.py (3.5.0)'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code in the editor is as follows:

```
for i in range(0, a):
    print('hello')
print ("Программа вывода слова hello n раз")
print ("Введите количество копий слова hello")
a=int(input('n='))
hello(a)
print ('Вывод с помощью функции lambda')
x='hello '
f=lambda x, a: x*a
print (f(x, a))
```

The output window below the editor shows the following text:

```
Программа вывода слова hello n раз
Введите количество копий слова hello
n=5
hello
hello
hello
hello
hello
Вывод с помощью функции lambda
hello hello hello hello hello
```

## Упражнение 11 Работа с последовательностями.

### Задание 1

1. Свяжите переменную с любой строкой, состоящей не менее чем из 8 символов. Извлеките из строки первый символ, затем последний, третий с начала и третий с конца. Измерьте длину вашей строки.
2. Присвойте произвольную строку длиной 10-15 символов переменной и извлеките из нее следующие срезы:
  - первые восемь символов;
  - четыре символа из центра строки;
  - символы с индексами кратными трем.

### Задание 2

1. Создайте два любых списка и свяжите их с переменными.
2. Извлеките из первого списка второй элемент.
3. Измените во втором списке последний объект. Выведите список на экран.

4. Соедините оба списка в один, присвоив результат новой переменной. Выведите получившийся список на экран.
5. "Снимите" срез из соединенного списка так, чтобы туда попали некоторые части обоих первых списков. Срез свяжите с очередной новой переменной. Выведите значение этой переменной.
6. Добавьте в список-срез два новых элемента и снова выведите его.

### Задание 3

1. Создайте словарь, связав его с переменной work, и наполните его данными, которые бы отражали количество сотрудников в десяти разных подразделениях.
2. Узнайте сколько человек в каком-нибудь подразделении.
3. Представьте, что на фирме произошли изменения, внесите их в словарь: ° в трех подразделениях изменилось количество сотрудников; ° на фирме появилось два новых подразделения; ° на фирме расформировали одно из подразделений.
4. Выведите содержимое словаря на экран.

Для справки приведены основные операции со строками списками, кортежами и словарями

Функция или метод	Назначение
<code>S = 'str'; S = "str"; S = '''str'''; S = ""str""</code>	<a href="#">Литералы строк</a>
<code>S = "s\np\ta\nbbb"</code>	Экранированные последовательности
<code>S = r"C:\temp\new"</code>	Неформатированные строки (подавляют экранирование)
<code>S = b"byte"</code>	Строка <a href="#">байтов</a>
<code>S1 + S2</code>	Конкатенация (сложение строк)
<code>S1 * 3</code>	Повторение строки
<code>S[i]</code>	Обращение по индексу
<code>S[i:j:step]</code>	Извлечение среза
<code>len(S)</code>	Длина строки
<code>S.find(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
<code>S.rfind(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
<code>S.index(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
<code>S.rindex(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError

Метод	Что делает
<b>list.append(x)</b>	Добавляет элемент в конец списка
<b>list.extend(L)</b>	Расширяет список list, добавляя в конец все элементы списка L
<b>list.insert(i, x)</b>	Вставляет на i-ый элемент значение x
<b>list.remove(x)</b>	Удаляет первый элемент в списке, имеющий значение x
<b>list.pop([i])</b>	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
<b>list.index(x, [start [, end]])</b>	Возвращает положение первого элемента от start до end со значением x
<b>list.count(x)</b>	Возвращает количество элементов со значением x
<b>list.sort([key = функция])</b>	Сортирует список на основе функции
<b>list.reverse()</b>	Разворачивает список
<b>list.copy()</b>	Поверхностная копия списка (новое в <a href="#">python 3.3</a> )
<b>list.clear()</b>	Очищает список (новое в python 3.3)

словарь: {ключ1: значение1, ключ2: значение2, . . ., ключN: значениеN, }

**dict.clear()** - очищает словарь.

**dict.copy()** - возвращает копию словаря.

classmethod **dict.fromkeys(seq[, value])** - создает словарь с ключами из seq и значением value (по умолчанию None).

**dict.get(key[, default])** - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).

**dict.items()** - возвращает пары (ключ, значение).

**dict.keys()** - возвращает ключи в словаре.

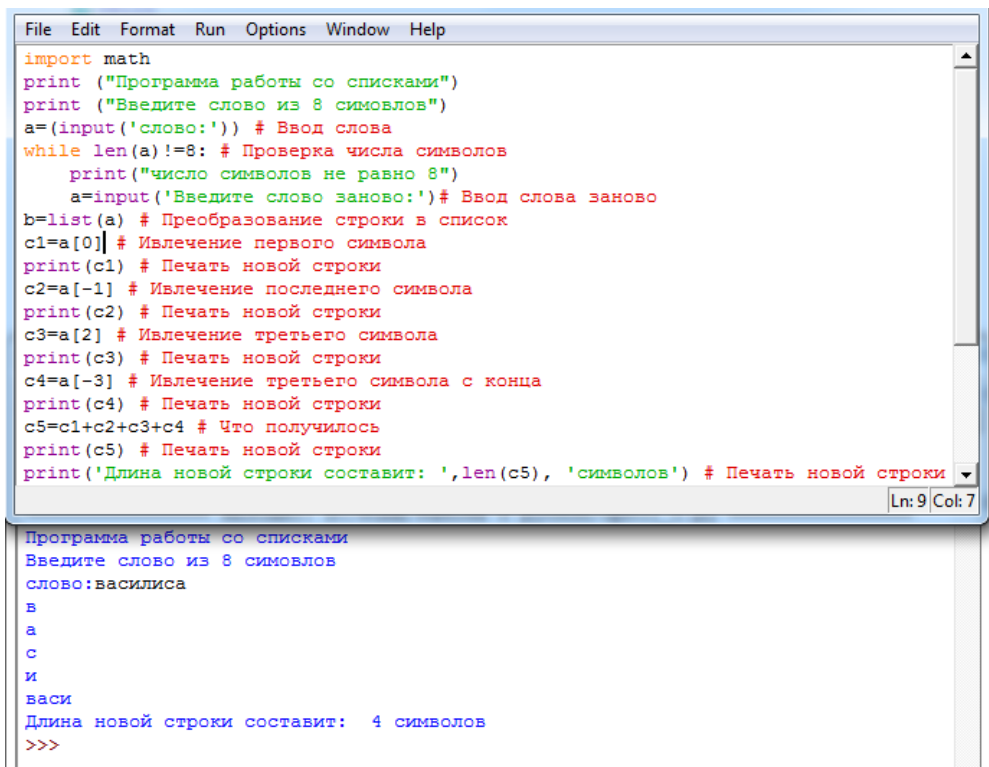
**dict.pop(key[, default])** - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).

**dict.popitem()** - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.

**dict.setdefault(key[, default])** - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None).

**dict.update([other])** - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).

**dict.values()** - возвращает значения в словаре.



```
File Edit Format Run Options Window Help
import math
print ("Программа работы со списками")
print ("Введите слово из 8 символов")
a=(input('слово:')) # Ввод слова
while len(a)!=8: # Проверка числа символов
    print("число символов не равно 8")
    a=input('Введите слово заново:')# Ввод слова заново
b=list(a) # Преобразование строки в список
c1=a[0] # Извлечение первого символа
print(c1) # Печать новой строки
c2=a[-1] # Извлечение последнего символа
print(c2) # Печать новой строки
c3=a[2] # Извлечение третьего символа
print(c3) # Печать новой строки
c4=a[-3] # Извлечение третьего символа с конца
print(c4) # Печать новой строки
c5=c1+c2+c3+c4 # Что получилось
print(c5) # Печать новой строки
print('Длина новой строки составит: ',len(c5), 'символов') # Печать новой строки
Ln: 9 Col: 7

Программа работы со списками
Введите слово из 8 символов
слово:василиса
в
а
с
и
васи
Длина новой строки составит:  4 символов
>>>
```

Пример решения 1-ой задачи приведен на скриншоте, нижняя часть – результат выполнения. Немного отличается от задания, но смысл тот же.

## Практическое задание №4

### Упражнение 12. Работа с файлами.

В файл можно записывать данные, редактировать, удалять данные, читать информацию из файла.

Предварительно его нужно открыть с этой целью используется функция

`open`

Файлу присваивается переменная.

Синтаксис: `open('имя файла','режим', [encoding])`

'имя файла' – путь к файлу, если указано только имя, то файл открывается или создается в текущей папке.

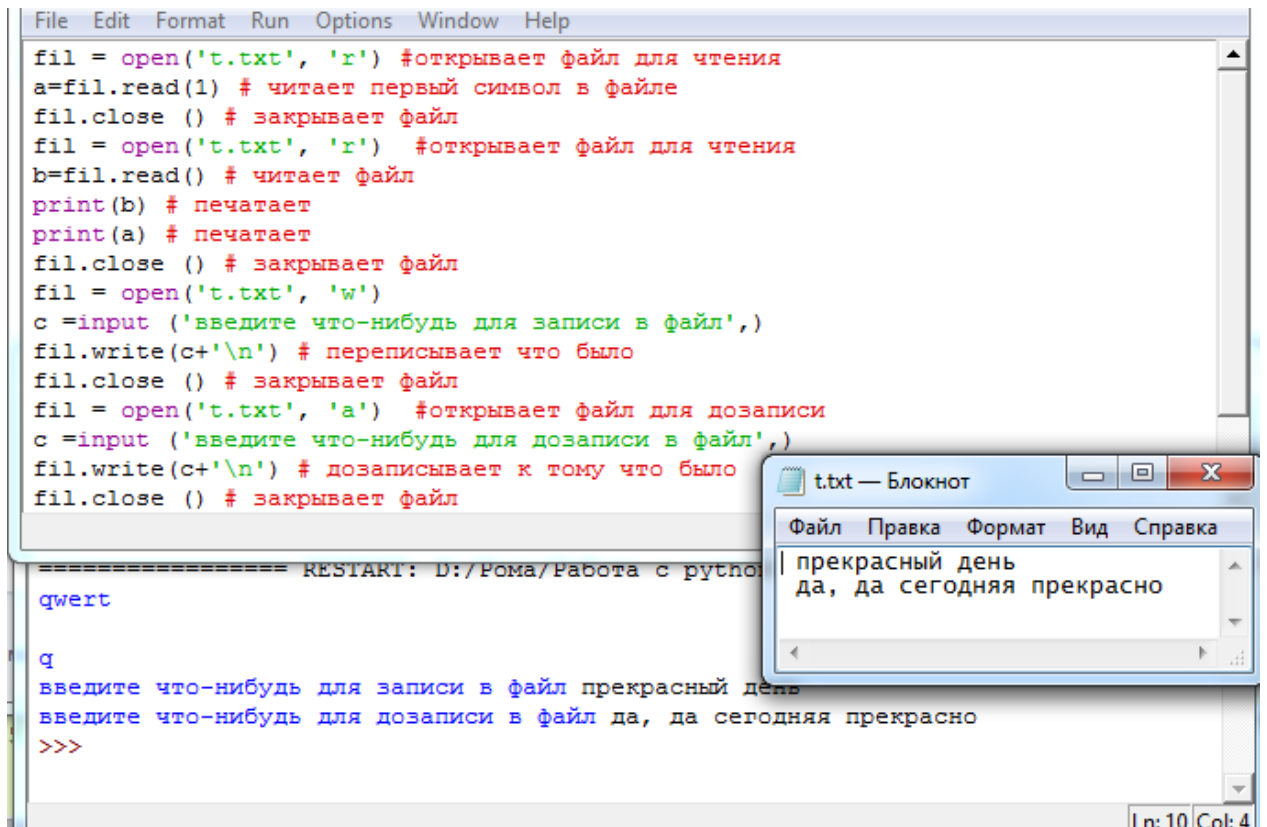
Режимы представлены в таблице.

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

Режимы могут быть объединены, то есть, к примеру, 'rb' - чтение в двоичном режиме. По умолчанию режим равен 'rt'.

Последний аргумент, encoding, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

На рисунке наглядно видна реализация соответствующих команд работы с файлами:



The screenshot shows a Python IDE window with a menu bar (File, Edit, Format, Run, Options, Window, Help) and a code editor. The code in the editor demonstrates file operations: opening 't.txt' for reading, reading the first character, closing the file, opening it again for reading, reading the entire file, printing it, closing it, opening it for writing, taking user input, writing the input, closing it, opening it for appending, taking more input, appending it, and closing it. Below the code, there is a prompt '==== RESTART: D:/Рома/Работа с python' followed by the user input 'qwert' and the program output 'прекрасный день' and 'да, да сегодня прекрасно'. A Notepad window titled 't.txt — Блокнот' is overlaid on the IDE, showing the text 'прекрасный день' and 'да, да сегодня прекрасно'.

```
File Edit Format Run Options Window Help
fil = open('t.txt', 'r') #открывает файл для чтения
a=fil.read(1) # читает первый символ в файле
fil.close () # закрывает файл
fil = open('t.txt', 'r') #открывает файл для чтения
b=fil.read() # читает файл
print(b) # печатает
print(a) # печатает
fil.close () # закрывает файл
fil = open('t.txt', 'w')
c =input ('введите что-нибудь для записи в файл',)
fil.write(c+'\n') # переписывает что было
fil.close () # закрывает файл
fil = open('t.txt', 'a') #открывает файл для дозаписи
c =input ('введите что-нибудь для дозаписи в файл',)
fil.write(c+'\n') # дозаписывает к тому что было
fil.close () # закрывает файл

==== RESTART: D:/Рома/Работа с python
qwert

q
введите что-нибудь для записи в файл прекрасный день
введите что-нибудь для дозаписи в файл да, да сегодня прекрасно
>>>
```


### Задание 1. Работа с Excel

Для работы с файлами Excel необходимо загрузить соответствующие библиотеки с официального сайта <http://www.python-excel.org/>

Для работы потребуется библиотека **XlsxWriter**

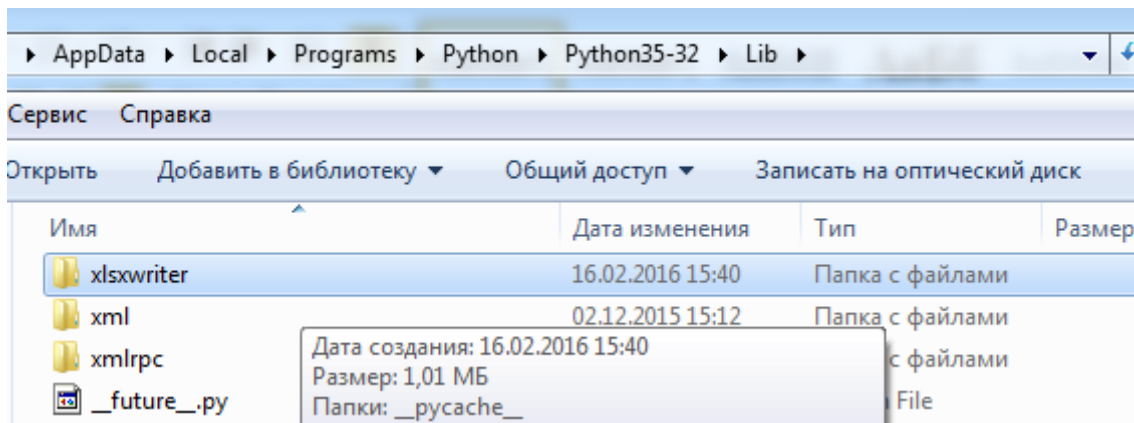
Для ее установи нужно зайти на страницу <https://pypi.python.org/pypi/XlsxWriter>

Где из таблицы скачать и затем распаковать соответствующий архив:



File	Type	Py Version	Uploaded on	Size
XlsxWriter-0.8.4-py2.py3-none-any.whl (md5)	Python Wheel	2.7	2016-01-16	131KB
XlsxWriter-0.8.4.tar.gz (md5)	Source		2016-01-16	229KB

В распакованном архиве выбрать папку `xlsxwriter` и скопировать его в папку библиотек Python.

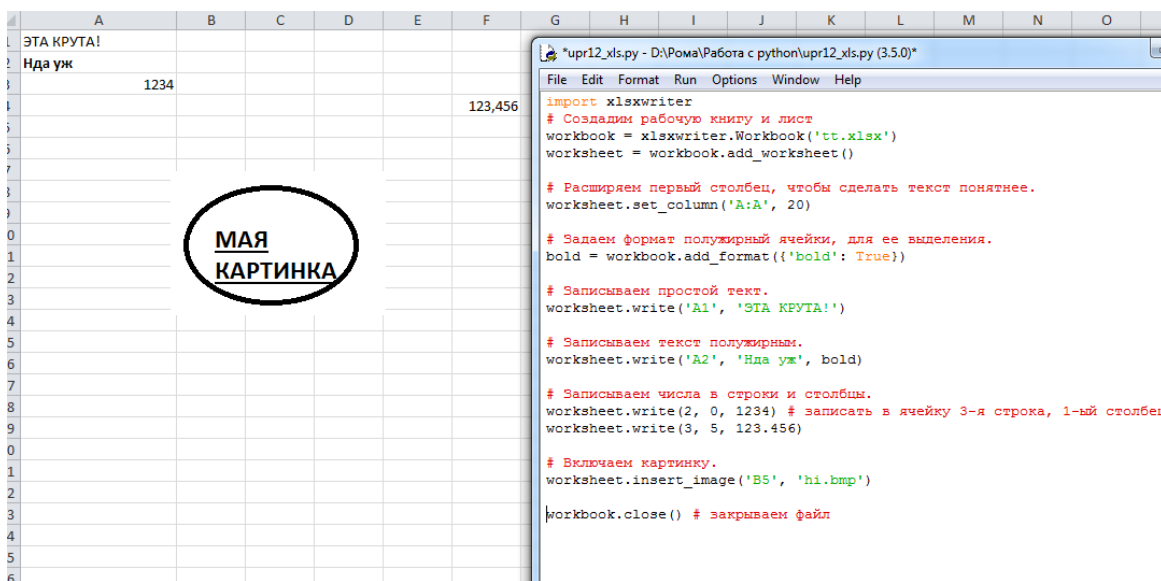


Теперь в программу можно подключить соответствующий модуль

`import xlswriter.`

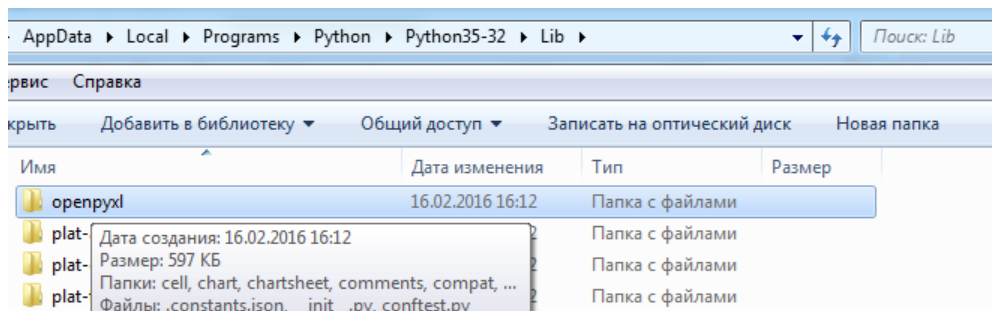
Документацию можно скачать на этом же сайте <http://xlswriter.readthedocs.org>.

Пример:



**Чтение из файла.**

Понадобится модуль `openpyxl` <https://pypi.python.org/pypi/openpyxl>, который также скачайте с сайта и распакуйте в библиотеку с питоном, документация <http://openpyxl.readthedocs.org/en/2.3.3/>



Далее необходимо скачать дополнительный модуль `jdcal` <https://pypi.python.org/simple/jdcal/> также распаковываем его в папку `lib`, а также `et_xmlfile` [https://pypi.python.org/pypi/et\\_xmlfile](https://pypi.python.org/pypi/et_xmlfile). Также нужно запустить в каждой папке файл `setup.py`

Создайте файл `t.xlsx` как показано на рисунке

	A	B	C	D
1	1	2	3	
2	4	5	6	
3	7	8	9	
4				
5				
6				

Напишите следующий код программы на Python и посмотрите на результат выполнения, дополните программу комментариями.

```

File Edit Format Run Options Window Help
from openpyxl import load_workbook
wb2 = load_workbook('t.xlsx')
print(wb2.get_sheet_names())
sheet_ranges = wb2['Первенец']
s2=''
s5=''
s6=''
for i in range(1,4):
    s=str(i)
    s1='A'+s
    s3='B'+s
    s4='C'+s
    print(sheet_ranges[s1].value)
    s2=s2+str(sheet_ranges[s1].value)
    s5=s5+str(sheet_ranges[s3].value)
    s6=s6+str(sheet_ranges[s4].value)
print(list(s2))
print(list(s5))
print(list(s6))
s7=list(s2)+list(s5)+list(s6)
print(s7)
ss=[int(s2[0]),int(s2[1]),int(s2[2]),int(s5[0]),int(s5[1]),int(s5[2]),int(s6[0]),int(s6[1]),int(s6[2])]
print("Вложенный список", ss)
def dterm():
    d1 = int(s2[0])*int(s5[1])*int(s6[2])
    d2 = int(s2[2])*int(s5[0])*int(s6[1])
    d3 = int(s2[1])*int(s5[2])*int(s6[0])
    d4 = int(s2[2])*int(s5[1])*int(s6[0])
    d5 = int(s2[0])*int(s5[2])*int(s6[1])
    d6 = int(s2[1])*int(s5[0])*int(s6[2])
    d = d1+d2+d3-d4-d5-d6
    print('Определитель 3x3 равен ',d)
    return d
dterm()

```

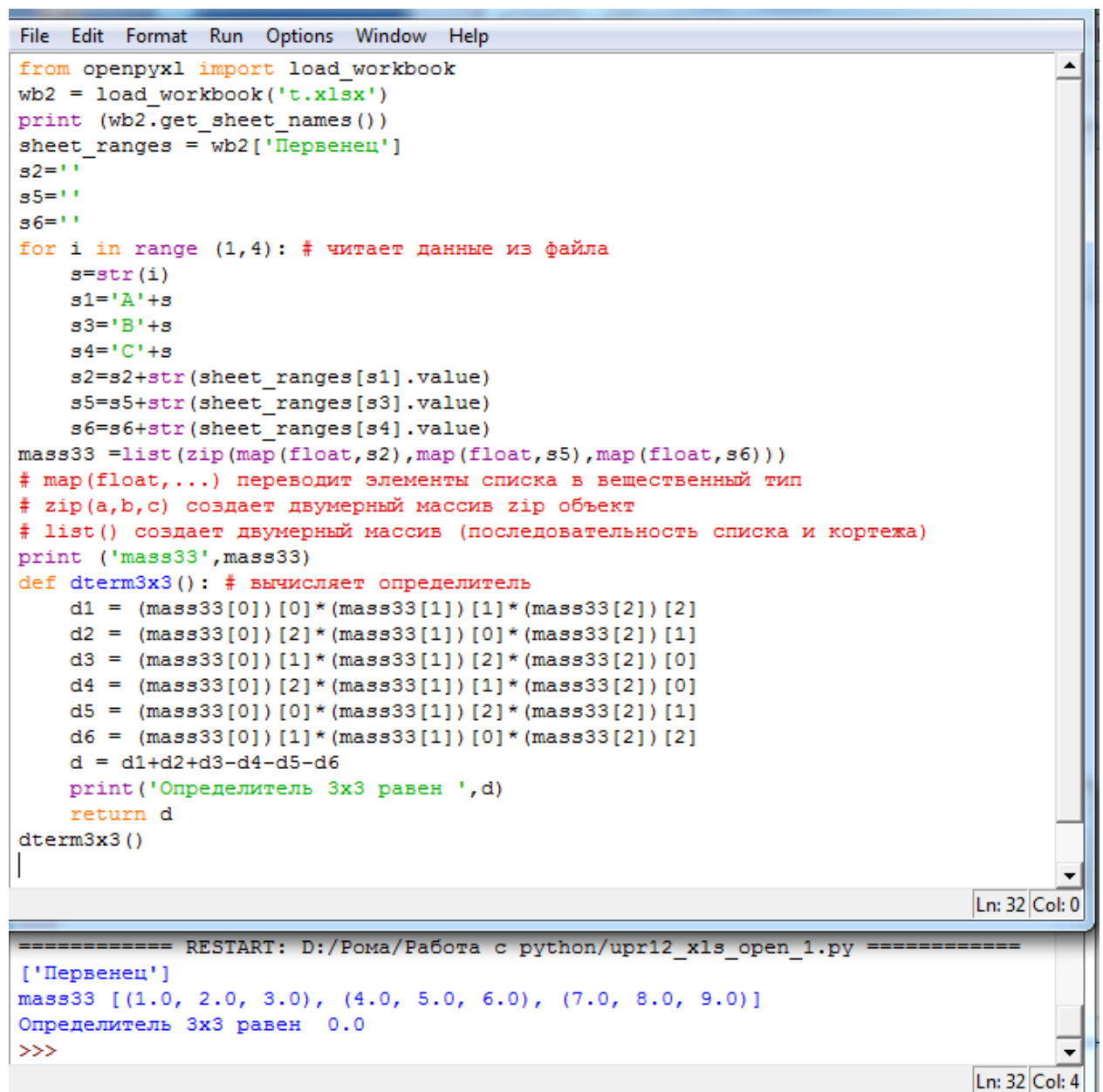
```

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:1
tel) on win32
Type "copyright", "credits" or "license()" for more
>>>
===== RESTART: D:\Рома\Работа с python\upr12.
['Первенец']
1
4
7
['1', '4', '7']
['2', '5', '8']
['3', '6', '9']
['1', '4', '7', '2', '5', '8', '3', '6', '9']
Вложенный список [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
Определитель 3x3 равен 0
>>>

```



Другой вариант создания массива представлен на рисунке ниже. Результат такой же.



```
File Edit Format Run Options Window Help
from openpyxl import load_workbook
wb2 = load_workbook('t.xlsx')
print (wb2.get_sheet_names())
sheet_ranges = wb2['Первенец']
s2=''
s5=''
s6=''
for i in range (1,4): # читает данные из файла
    s=str(i)
    s1='A'+s
    s3='B'+s
    s4='C'+s
    s2=s2+str(sheet_ranges[s1].value)
    s5=s5+str(sheet_ranges[s3].value)
    s6=s6+str(sheet_ranges[s4].value)
mass33 =list(zip(map(float,s2),map(float,s5),map(float,s6)))
# map(float,...) переводит элементы списка в вещественный тип
# zip(a,b,c) создает двумерный массив zip объект
# list() создает двумерный массив (последовательность списка и кортежа)
print ('mass33',mass33)
def dterm3x3(): # вычисляет определитель
    d1 = (mass33[0][0]*(mass33[1][1]*(mass33[2][2])
    d2 = (mass33[0][2]*(mass33[1][1]*(mass33[2][1])
    d3 = (mass33[0][1]*(mass33[1][2]*(mass33[2][0])
    d4 = (mass33[0][2]*(mass33[1][1]*(mass33[2][0])
    d5 = (mass33[0][0]*(mass33[1][2]*(mass33[2][1])
    d6 = (mass33[0][1]*(mass33[1][0]*(mass33[2][2])
    d = d1+d2+d3-d4-d5-d6
    print ('Определитель 3x3 равен ',d)
    return d
dterm3x3()
|
```

Ln: 32 Col: 0

```
===== RESTART: D:/Рома/Работа с python/upr12_xls_open_1.py =====
['Первенец']
mass33 [(1.0, 2.0, 3.0), (4.0, 5.0, 6.0), (7.0, 8.0, 9.0)]
Определитель 3x3 равен  0.0
>>>
```

Ln: 32 Col: 4

Задание 2. Создайте программу, которая:

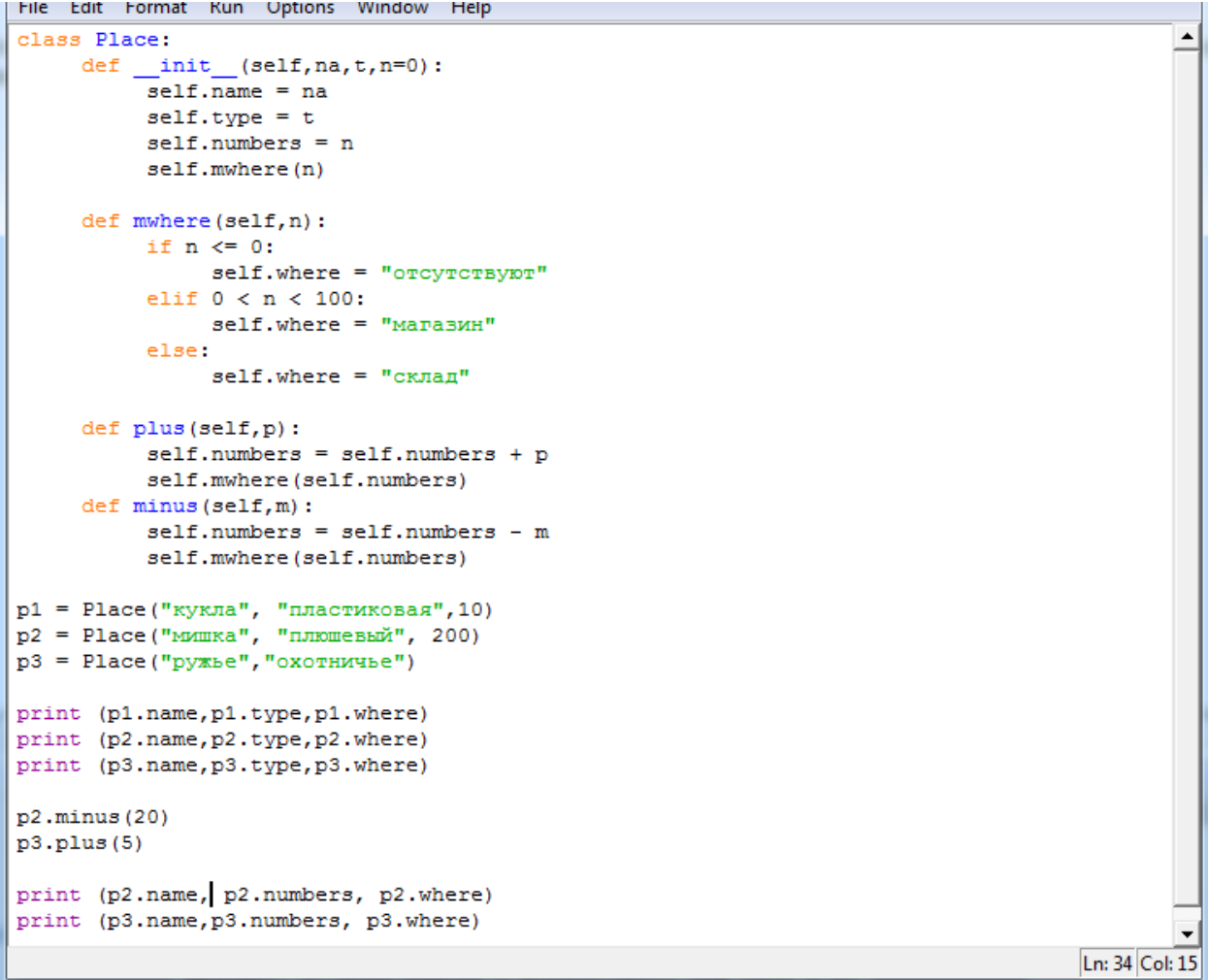
1. Формирует список (квадратную матрицу произвольного порядка). Ввод элементов матрицы может осуществляться с клавиатуры, либо из файла (можно \*.txt файл) (ветвление осуществляется через диалог с пользователем).
2. Выводит в файл или на экран элементы матрицы, а также считает определитель (ветвление осуществляется через диалог с пользователем).

### Упражнение 13. Создание классов и объектов.

Создайте класс студентов, имеющих следующие атрибуты: ФИО, номер группы, оценку на последнем экзамене, которая влияет на свойство, принимающее значения: (отличник, хорошист, середнячок, двоечник).

Организуйте ввод и вывод информации о студентах (4 объекта).

Пример работы с классами и методами приведен на рисунке.



```
File Edit Format Run Options Window Help
class Place:
    def __init__(self,na,t,n=0):
        self.name = na
        self.type = t
        self.numbers = n
        self.mwhere(n)

    def mwhere(self,n):
        if n <= 0:
            self.where = "отсутствуют"
        elif 0 < n < 100:
            self.where = "магазин"
        else:
            self.where = "склад"

    def plus(self,p):
        self.numbers = self.numbers + p
        self.mwhere(self.numbers)
    def minus(self,m):
        self.numbers = self.numbers - m
        self.mwhere(self.numbers)

p1 = Place("кукла", "пластиковая",10)
p2 = Place("мишка", "плюшевый", 200)
p3 = Place("ружье", "охотничье")

print (p1.name,p1.type,p1.where)
print (p2.name,p2.type,p2.where)
print (p3.name,p3.type,p3.where)

p2.minus(20)
p3.plus(5)

print (p2.name, p2.numbers, p2.where)
print (p3.name,p3.numbers, p3.where)
Ln: 34 Col: 15

кукла пластиковая магазин
мишка плюшевый склад
ружье охотничье отсутствуют
мишка 180 склад
ружье 5 магазин
>>>
```

## Упражнение 14. Наследование. Добавление новых свойств

На основании предыдущего примера добавьте к объекту 1 и 2 новое свойство: рост. Выведите на печать все их свойства.

Пример добавления новых свойств представлен на рисунке.

---

```
class Things:
    def __init__(self,n,t):
        self.namething = n
        self.total = t

th1 = Things("стол", 5)
th2 = Things("стулья", 7)

print (th1.namething,th1.total) # Вывод: столов 5
print (th2.namething,th2.total) # Вывод: стульев 7

th1.color = "ольха" # новое свойство объекта th1

print (th1.color) # Вывод: ольха
print (th2.color) # ОШИБКА: у объекта th2 нет свойства color!
```

## Упражнение 15. Наследование. Создание подкласса

Создайте дополнительные классы компьютерных и рабочих столов с дополнительными, отличающимися от примера на рисунке методами

```
File Edit Format Run Options Window Help
class Table:
    def __init__(self,l,w,h):
        self.long = l
        self.width = w
        self.height = h
    def outing(self):
        print (self.long,self.width,self.height)

class Kitchen(Table):
    def howplaces(self,n):
        if n < 2:
            print ("Это не кухонный стол")
        else:
            self.places = n
    def outplaces(self):
        print (self.places)

t_1 = Kitchen(2,1,0.5)
t_1.outing()
t_1.howplaces(5)
t_1.outplaces()

t_2 = Table(1,3,0.7)
t_2.outing()
t_2.howplaces(8) # ОШИБКА
```

## Упражнение 15. Полиморфизм

Напишите программу, запрашивающую у пользователя ввод числа. Если число принадлежит диапазону от -100 до 100, то создается объект одного класса, во всех остальных случаях создается объект другого класса. В обоих классах должен быть метод-конструктор `__init__`, который в первом классе возводит число в куб, а во-втором - умножает на три.

Пример полиморфизма показан на рисунке ниже

```
File Edit Format Run Options Window Help
class T1:
    n=10
    def total(self,N):
        self.total = int(self.n) + int(N)

class T2:
    def total(self,s):
        self.total = len(str(s))

t1 = T1()
t2 = T2()
t1.total(45)
t2.total(45)
print (t1.total) # Вывод: 55
print (t2.total) # Вывод: 2
```

### Упражнение 16. Переопределение и расширение методов

Создайте класс зарплаты, который вычисляет сумму зарплаты в зависимости от тарифной ставки и числа отработанных дней. Расширьте метод начислением процентов в зависимости от объема продаж

Переопределите метод начисления зарплаты, в случае, если объем продаж в месяц превысил 1 000 000 то проценты добавляются к зарплате, в противном случае вычитаются.

### Упражнение 17. Перегрузка операторов

Дополните код программы, представленной на рисунке, дополнительными методами класса методами `__mul__` (вызывается при использовании объекта в операциях умножения) и `__sub__` (вычитание). Вызовите данные методы с помощью соответствующих операций с объектами. Для каких объектов невозможно использовать метод `__sub__`?

```
File Edit Format Run Options Window Help

class Newclass:
    def __init__(self, base):
        self.base = base
    def __add__(self, a):
        self.base = self.base + a
    def __str__(self):
        return "%s !!! " % self.base

a = Newclass(10)
a + 20
print (a)

b = Newclass("yes")
b + "terday"
print (b)

c = Newclass([2,6,3])
c + [7, 1]
print (c)
|

30 !!!
yesterday !!!
[2, 6, 3, 7, 1] !!!
>>>
```

## Упражнение 18. Перегрузка операторов `__Call__`

Создайте класс с методом `__call__`, принимающим два параметра и производящим над ними те или иные математические операции. Создайте несколько объектов класса и, затем, обратитесь к ним как к функциям.

Пример показан на рисунке

```
File Edit Format Run Options Window Help
class Changeable:
    def __init__(self, color):
        self.color = color
    def __call__(self, newcolor):
        self.color = newcolor
    def __str__(self):
        return "%s" % self.color

canvas = Changeable("green")
frame = Changeable("blue")

canvas("red")
frame("yellow")

print (canvas, frame)
red yellow
>>>
```

Для справки: наиболее часто используемыми операторами являются следующие:

**\_\_new\_\_(cls[, ...])** — управляет созданием экземпляра. В качестве обязательного аргумента принимает класс (не путать с экземпляром). Должен возвращать экземпляр класса для его последующей его передачи методу **\_\_init\_\_**.

**\_\_init\_\_(self[, ...])** - как уже было сказано выше, конструктор.

**\_\_del\_\_(self)** - вызывается при удалении объекта сборщиком мусора.

**\_\_repr\_\_(self)** - вызывается встроенной функцией `repr`; возвращает "сырые" данные, использующиеся для внутреннего представления в python.

**\_\_str\_\_(self)** - вызывается функциями `str`, `print` и `format`. Возвращает строковое представление объекта.

**\_\_bytes\_\_(self)** - вызывается функцией `bytes` при преобразовании к байтам.

**\_\_format\_\_(self, format\_spec)** - используется функцией `format` (а также методом `format` у строк).

**\_\_lt\_\_(self, other)** -  $x < y$  вызывает `x.__lt__(y)`.

**\_\_le\_\_(self, other)** -  $x \leq y$  вызывает `x.__le__(y)`.

**\_\_eq\_\_(self, other)** -  $x == y$  вызывает `x.__eq__(y)`.

**\_\_ne\_\_**(self, other) -  $x \neq y$  вызывает **\_\_ne\_\_**(y)

**\_\_gt\_\_**(self, other) -  $x > y$  вызывает **\_\_gt\_\_**(y).

**\_\_ge\_\_**(self, other) -  $x \geq y$  вызывает **\_\_ge\_\_**(y).

**\_\_hash\_\_**(self) - получение хэш-суммы объекта, например, для добавления в словарь.

**\_\_bool\_\_**(self) - вызывается при проверке истинности. Если этот метод не определён, вызывается метод **\_\_len\_\_** (объекты, имеющие ненулевую длину, считаются истинными).

**\_\_getattr\_\_**(self, name) - вызывается, когда атрибут экземпляра класса не найден в обычных местах (например, у экземпляра нет метода с таким названием).

**\_\_setattr\_\_**(self, name, value) - назначение атрибута.

**\_\_delattr\_\_**(self, name) - удаление атрибута (del obj.name).

**\_\_call\_\_**(self[, args...]) - вызов экземпляра класса как функции.

**\_\_len\_\_**(self) - длина объекта.

**\_\_getitem\_\_**(self, key) - доступ по индексу (или ключу).

**\_\_setitem\_\_**(self, key, value) - назначение элемента по индексу.

**\_\_delitem\_\_**(self, key) - удаление элемента по индексу.

**\_\_iter\_\_**(self) - возвращает итератор для контейнера.

**\_\_reversed\_\_**(self) - итератор из элементов, следующих в обратном порядке.

**\_\_contains\_\_**(self, item) - проверка на принадлежность элемента контейнеру (item in self).

### Перегрузка арифметических операторов

**\_\_add\_\_**(self, other) - сложение.  $x + y$  вызывает **\_\_add\_\_**(y).

**\_\_sub\_\_**(self, other) - вычитание ( $x - y$ ).

**\_\_mul\_\_**(self, other) - умножение ( $x * y$ ).

**\_\_truediv\_\_**(self, other) - деление ( $x / y$ ).

**\_\_floordiv\_\_**(self, other) - целочисленное деление ( $x // y$ ).

**\_\_mod\_\_**(self, other) - остаток от деления ( $x \% y$ ).

**\_\_divmod\_\_**(self, other) - частное и остаток (divmod(x, y)).



**\_\_pow\_\_**(self, other[, modulo]) - возведение в степень ( $x ** y$ , `pow(x, y[, modulo])`)).

**\_\_lshift\_\_**(self, other) - битовый сдвиг влево ( $x << y$ ).

**\_\_rshift\_\_**(self, other) - битовый сдвиг вправо ( $x >> y$ ).

**\_\_and\_\_**(self, other) - битовое И ( $x \& y$ ).

**\_\_xor\_\_**(self, other) - битовое ИСКЛЮЧАЮЩЕЕ ИЛИ ( $x \wedge y$ ).

**\_\_or\_\_**(self, other) - битовое ИЛИ ( $x | y$ ).

Пойдём дальше.

**\_\_radd\_\_**(self, other),

**\_\_rsub\_\_**(self, other),

**\_\_rmul\_\_**(self, other),

**\_\_rtruediv\_\_**(self, other),

**\_\_rfloordiv\_\_**(self, other),

**\_\_rmod\_\_**(self, other),

**\_\_rdivmod\_\_**(self, other),

**\_\_rpow\_\_**(self, other),

**\_\_rlshift\_\_**(self, other),

**\_\_rrshift\_\_**(self, other),

**\_\_rand\_\_**(self, other),

**\_\_rxor\_\_**(self, other),

**\_\_ror\_\_**(self, other) - делают то же самое, что и арифметические операторы, перечисленные выше, но для аргументов, находящихся справа, и только в случае, если для левого операнда не определён соответствующий метод.

Например, операция  $x + y$  будет сначала пытаться вызвать `x.__add__(y)`, и только в том случае, если это не получилось, будет пытаться вызвать `y.__radd__(x)`. Аналогично для остальных методов.

Идём дальше.

**\_\_iadd\_\_**(self, other) -  $+=$ .

**\_\_isub\_\_**(self, other) -  $-=$ .

**\_\_imul\_\_**(self, other) -  $*=$ .

**\_\_itruediv\_\_**(self, other) - /=.  
**\_\_ifloordiv\_\_**(self, other) - //=.  
**\_\_imod\_\_**(self, other) - %=.  
**\_\_ipow\_\_**(self, other[, modulo]) - \*\*=.  
**\_\_ilshift\_\_**(self, other) - <<=.  
**\_\_irshift\_\_**(self, other) - >>=.  
**\_\_iand\_\_**(self, other) - &=.  
**\_\_ixor\_\_**(self, other) - ^=.  
**\_\_ior\_\_**(self, other) - |=.  
**\_\_neg\_\_**(self) - унарный -.  
**\_\_pos\_\_**(self) - унарный +.  
**\_\_abs\_\_**(self) - модуль (abs()).  
**\_\_invert\_\_**(self) - инверсия (~).  
**\_\_complex\_\_**(self) - приведение к complex.  
**\_\_int\_\_**(self) - приведение к int.  
**\_\_float\_\_**(self) - приведение к float.  
**\_\_round\_\_**(self[, n]) - округление.  
**\_\_enter\_\_**(self), **\_\_exit\_\_**(self, exc\_type, exc\_value, traceback) - реализация менеджеров контекста.

## Практическое задание №6 Разработка графических интерфейсов

Описание опций методов, виджетов модуля tkinter см.  
<http://www.russianlutheran.org/python/life/life.htm#ClassEntry>

### Упражнение 19.

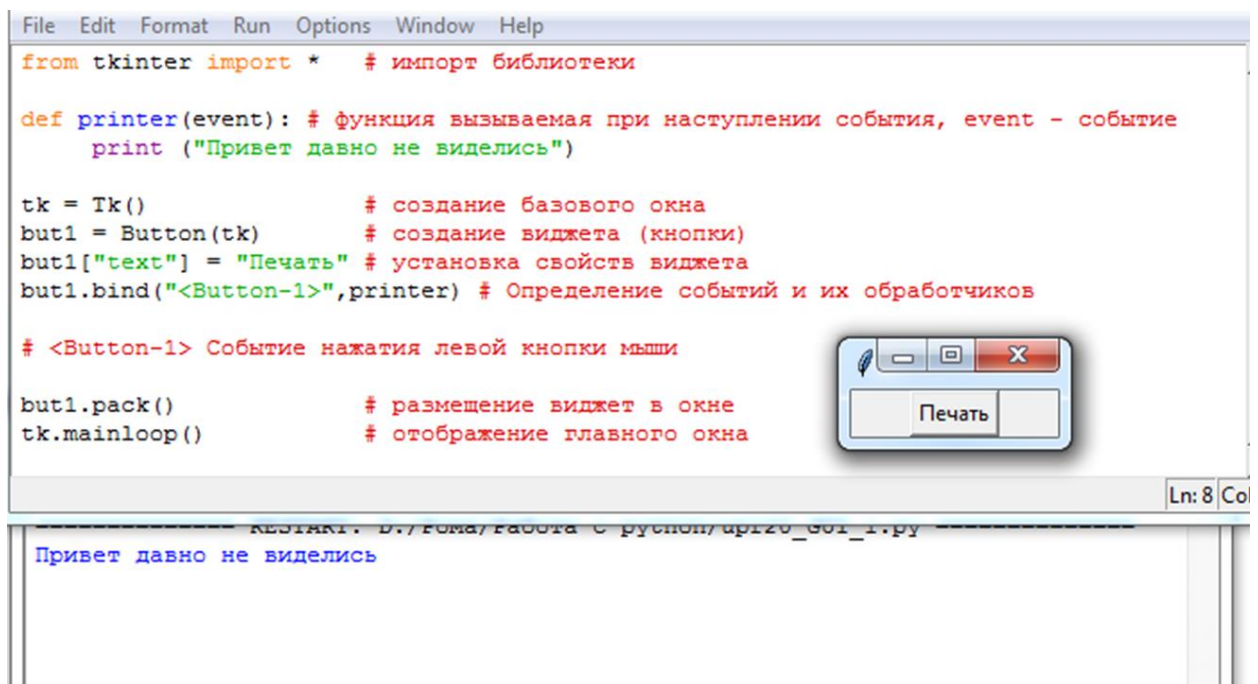
1. Импортируйте модуль tkinter, создайте объект главного окна, примените к нему метод mainloop. Затем выполните скрипт. Что вы видите?
2. Добавьте кнопку на главное окно с помощью такой команды:

```
but = Button(root, text="Печать")
```

В данном случае, при создании кнопки, в класс сразу передается и значение свойства text. Это наиболее часто используемый способ установки свойств (по-сравнению с тем, который приводится в примере: `but["text"] = "Печать"`).

3. Расположите виджету на главном окне с помощью метода `pack`. Запустите скрипт. Что вы видите? Нажмите левой кнопкой мыши на кнопку в окне. Что-нибудь происходит?
4. Создайте какую-нибудь функцию (отличную от примера) и свяжите ее с событием нажатия кнопки.
5. Снова запустите скрипт и нажмите кнопку. По идее, должно что-то произойти.

Создайте программу в 2-х вариантах операциональном и объектно-ориентированном.



```
File Edit Format Run Options Window Help
from tkinter import * # импорт библиотеки

def printer(event): # функция вызываемая при наступлении события, event - событие
    print ("Привет давно не виделись")

tk = Tk() # создание базового окна
but1 = Button(tk) # создание виджета (кнопки)
but1["text"] = "Печать" # установка свойств виджета
but1.bind("<Button-1>",printer) # Определение событий и их обработчиков

# <Button-1> Событие нажатия левой кнопки мыши

but1.pack() # размещение виджет в окне
tk.mainloop() # отображение главного окна
```

Ln: 8 Co

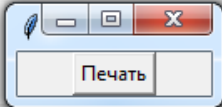
Привет давно не виделись

*Обычный вариант*

```
File Edit Format Run Options Window Help
from tkinter import * # импорт библиотеки

class But1_print:
    def __init__(self):
        self.but1 = Button(tk) # создание виджета (кнопки)
        self.but1["text"] = "Печать" # установка свойств виджета
        self.but1.bind("<Button-1>",self.printer) # Определение событий и их обработчиков
        self.but1.pack() # размещение виджет в окне
    def printer(self,event): # функция вызываемая при наступлении события, event - событие
        print ("Привет давно не виделись")

tk = Tk() # создание базового окна
obj1 = But1_print() # создание объекта класса кнопка
tk.mainloop() # отображение главного окна
```

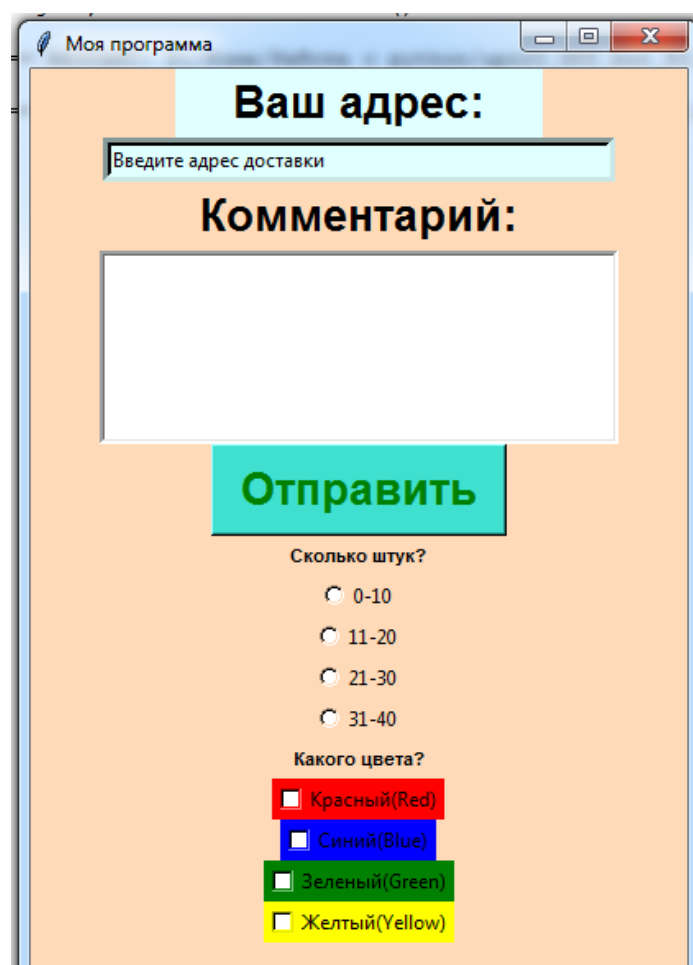


*Объектно-ориентированный вариант.*

Внимание в 3-м питоне модули пишутся с маленькой буквы, а классы с большой.

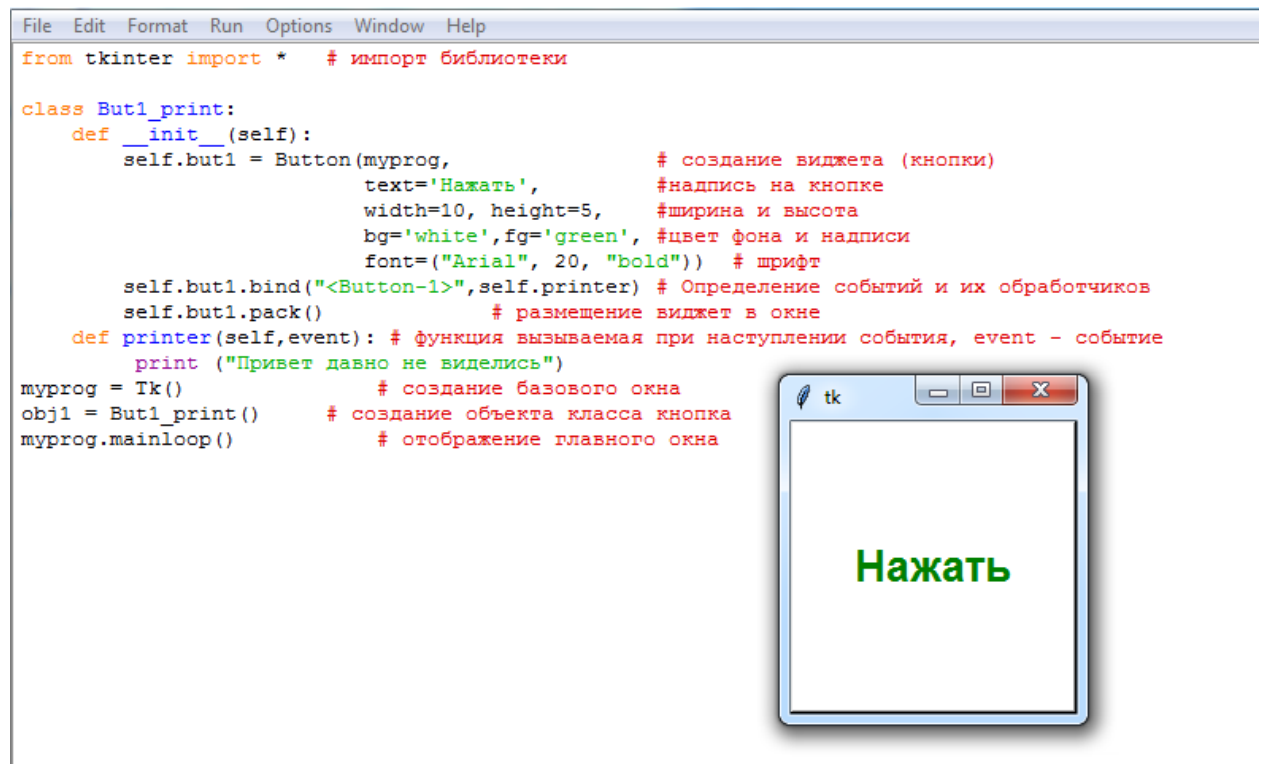
## Упражнение 20. Работа с основными виджетами

Создайте программу, которая бы формировала следующее окно:

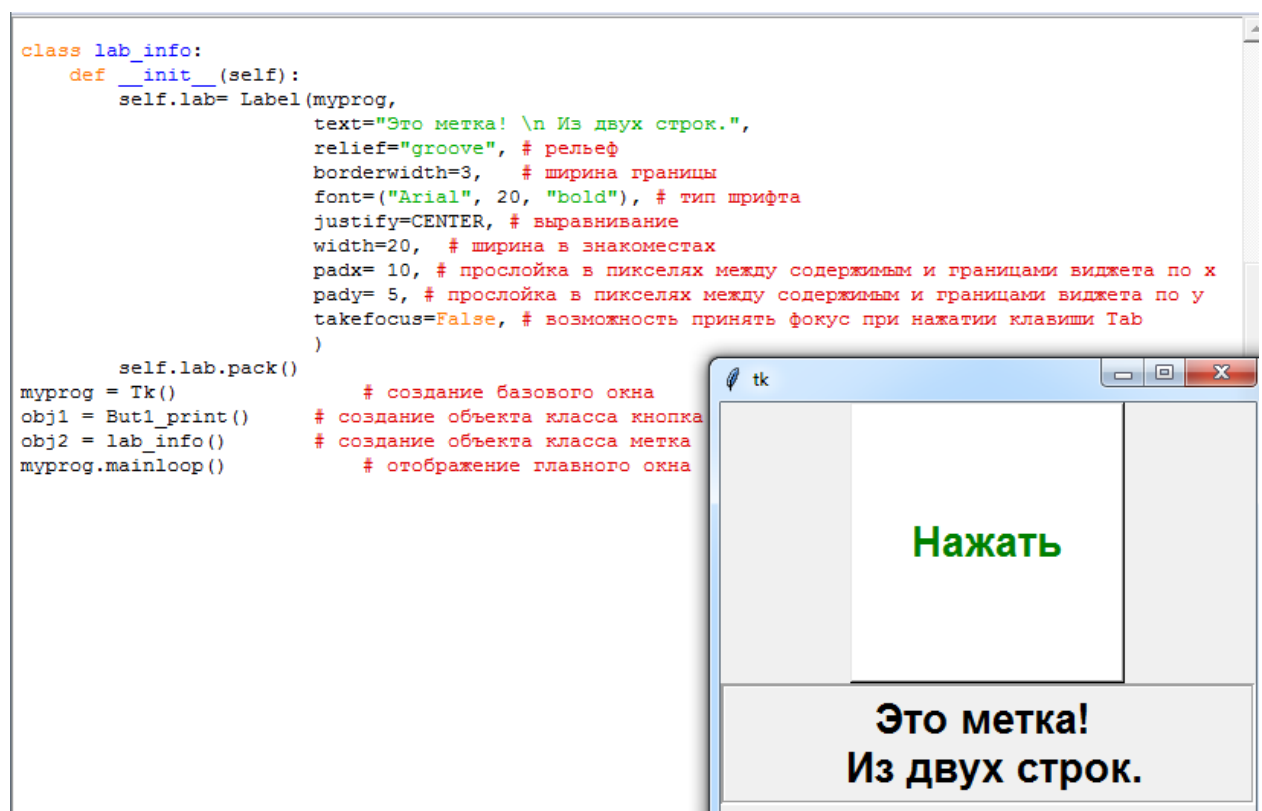


Используйте объектно-ориентированный стиль. Таблицу цветовых кодов можете посмотреть [здесь](#).

Исходный код для формирования соответствующих виджетов представлен на следующих рисунках



Виджет кнопка



Виджет метка

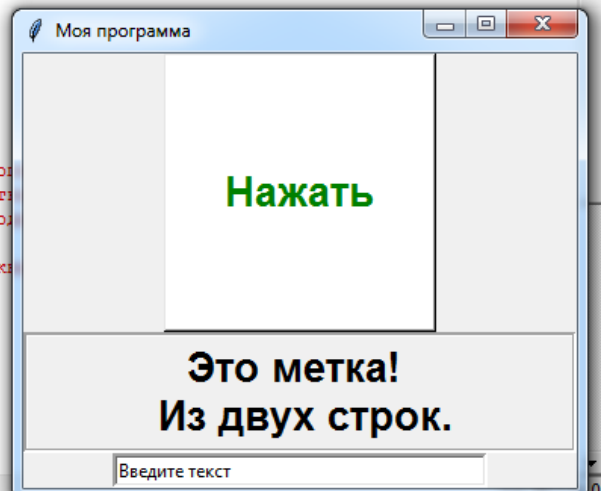
```

class ent_one:
    def __init__(self):
        self.ent = Entry(myprog,
                        textvariable=tv, # текстовая переменная
                        takefocus=True, # возможность принять фокус при нажатии клавиши Tab
                        width=40, bd=3) # bd - это сокращение от borderwidth (ширина границы)

        self.ent.pack()

myprog = Tk() # создание базового окна
tv = StringVar()
myprog.title("Моя программа")
obj1 = But1_print() # создание объекта класса кнопки
obj2 = lab_info() # создание объекта класса метки
obj3 = ent_one() # создание объекта класса Entry
tv.set("Введите текст")
myprog.mainloop() # отображение главного окна

```



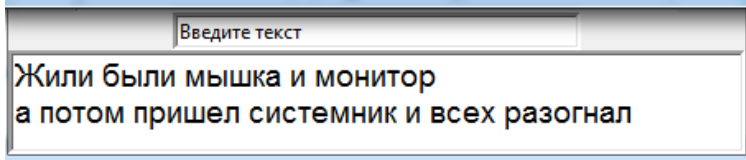
Однострочное текстовое поле

```

class tex_one:
    def __init__(self):
        self.tex = Text(myprog,
                        takefocus=True, # возможность принять фокус при нажатии клавиши Tab
                        width=40, bd=3, # bd - это сокращение от borderwidth (ширина границы)
                        font='Arial 14', # шрифт
                        wrap=WORD) # тип переноса слов

        self.tex.pack()

```



Однострочное текстовое поле

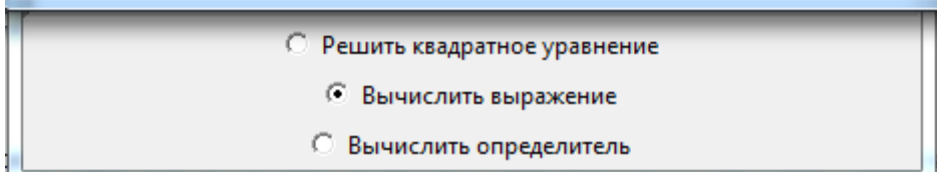
```

class rad_but:
    def __init__(self):
        self.rad0 = Radiobutton(myprog, text="Решить квадратное уравнение",
                                variable=rv, value=0)
        self.rad1 = Radiobutton(myprog, text="Вычислить выражение",
                                variable=rv, value=1)
        self.rad2 = Radiobutton(myprog, text="Вычислить определитель",
                                variable=rv, value=2)

        self.rad0.pack()
        self.rad1.pack()
        self.rad2.pack()

myprog = Tk()                # создание базового окна
tv = StringVar()             # переменная для однострочного текстового поля
rv = IntVar()                # переменная для выбора радиокнопки
myprog.title("Моя программа")
obj1 = But1_print()          # создание объекта класса кнопка
obj2 = lab_info()            # создание объекта класса метка
obj3 = ent_one()             # создание объекта класса однострочное текстовое поле
obj4 = tex_one()             # создание объекта класса многострочное текстовое поле
obj5 = rad_but()             # создание объекта класса радиокнопки
tv.set("Введите текст")
rv.set(1)
myprog.mainloop()           # отображение главного окна

```



Радиокнопка

```

class check_but:
    def __init__(self):
        c1=IntVar()
        c2=IntVar()
        c3=IntVar()
        c1.set(0)
        c2.set(0)
        c3.set(0)
        self.che1 = Checkbutton(myprog, text="Вывести результат", justify=LEFT,
                                variable=c1, onvalue=1, offvalue=0)
        self.che2 = Checkbutton(myprog, text="Показать решение", justify=LEFT,
                                variable=c2, onvalue=2, offvalue=0)
        self.che3 = Checkbutton(myprog, text="Сделать вывод ", justify=LEFT,
                                variable=c3, onvalue=3, offvalue=0)
        self.che1.pack()
        self.che2.pack()
        self.che3.pack()

```

☐ Вывести результат  
☐ Показать решение  
☐ Сделать вывод

## Флажки

```

class list_b:
    def __init__(self):
        r = ['Delphi', 'Python', 'Basic', 'Fortran']
        self.lis = Listbox(myprog, selectmode=SINGLE, height=4)
        for i in r:
            self.lis.insert(END, i)
        self.lis.pack()

```

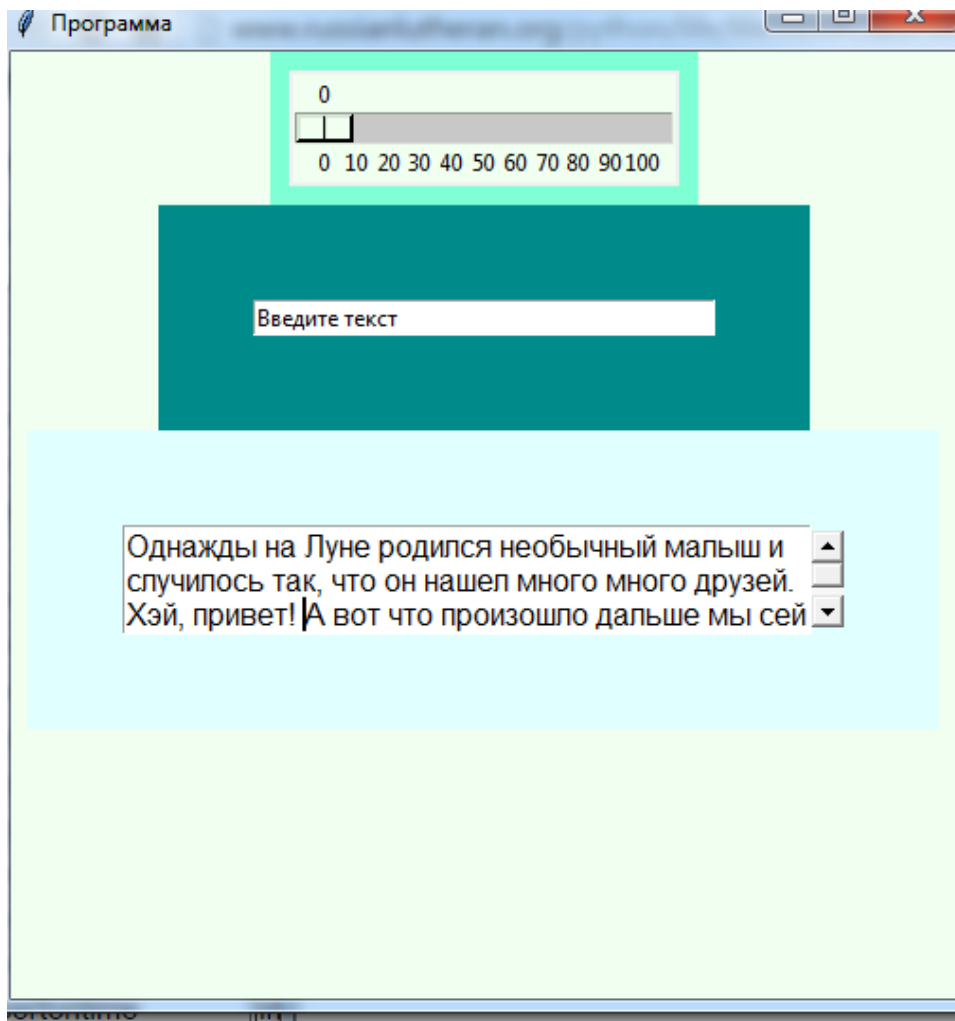
Delphi  
 Python  
 Basic  
 Fortran

## Списки

### Упражнение 21

Создайте программу, реализующую следующее окно (первоначальный размер окна можно задать как `prog.minsize(500,500)` либо `prog.geometry('500x500')`, цвет `prog["bg"]="цвет"`).





**Внимание.** Для того чтобы фрейм не удалялся необходимо в ее опции добавить `bd=10` (граница)

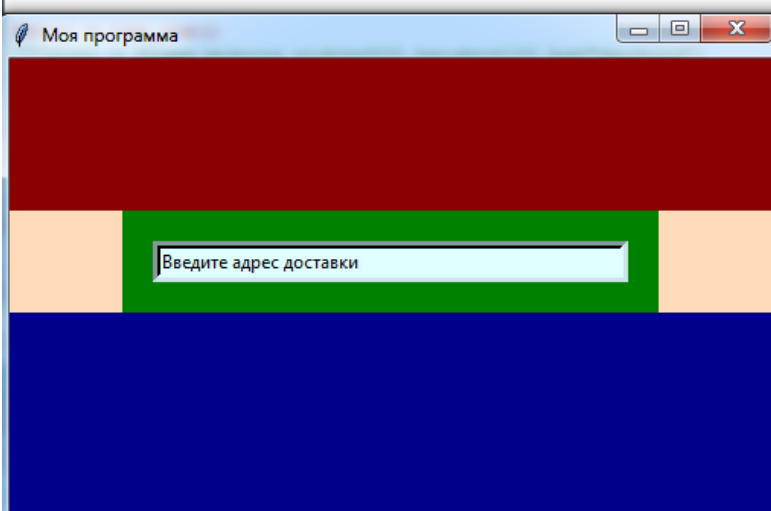
Пример кода создания фреймов и вставки виджетов представлен на рисунках ниже.

```

class fr_3:
    def __init__(self,w,h,c,bd=0,a=0):
        self.fra = Frame(width=w,height=h,bg=c,bd=bd)
        self.fra.pack()
        if a==1: ent1=ent_one(self.fra)

myprog = Tk() # создание базового окна
myprog.title("Моя программа") # Название окна
myprog["bg"]="Peachpuff"
fra1 = fr_3(500,100,'darkred')
fra2 = fr_3(300,200,'green',20,1)
fra3 = fr_3(500,150,'darkblue')

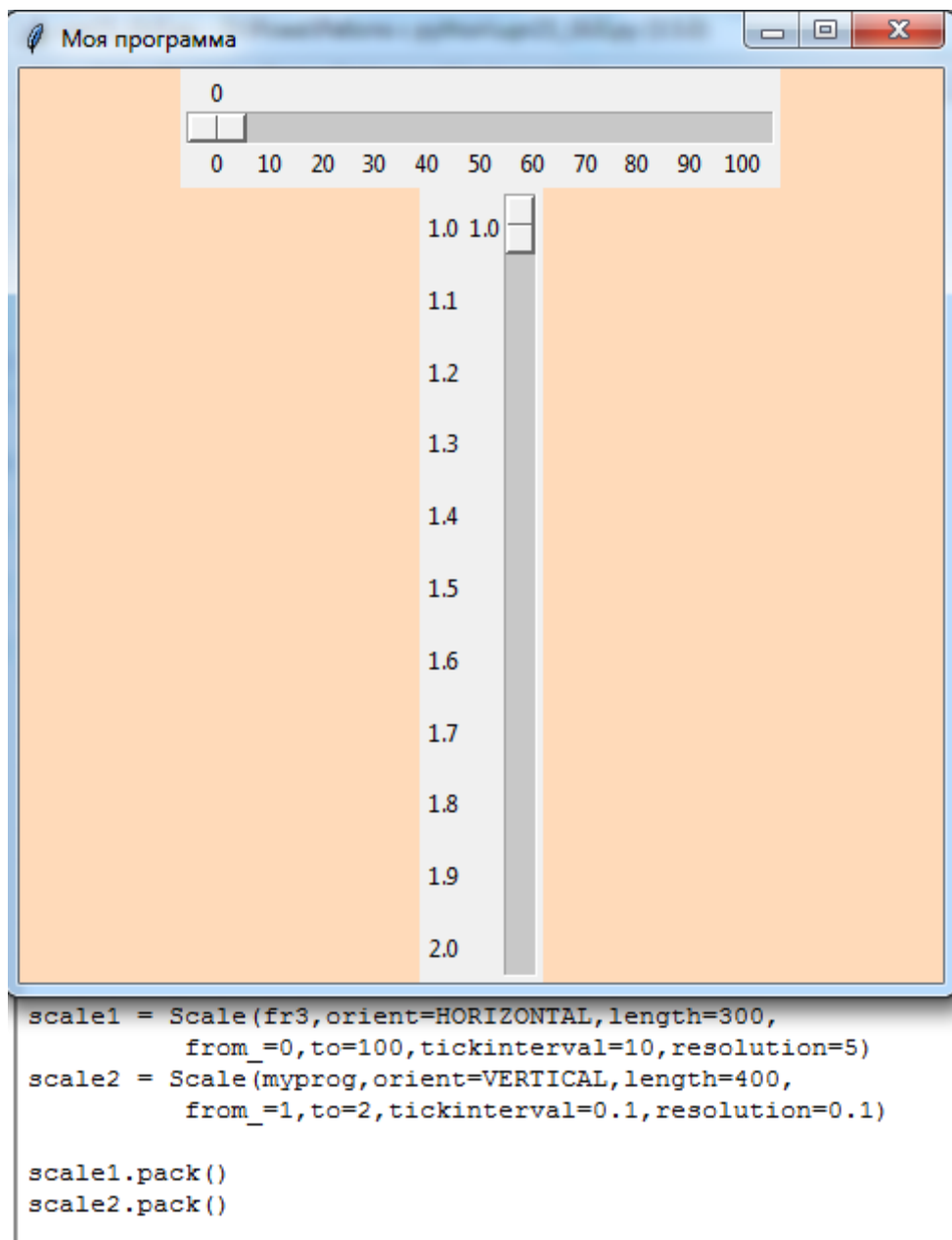
```



*Создание фреймов и вставка виджета ООП*



*Создание фреймов и вставка виджета в операторной форме*



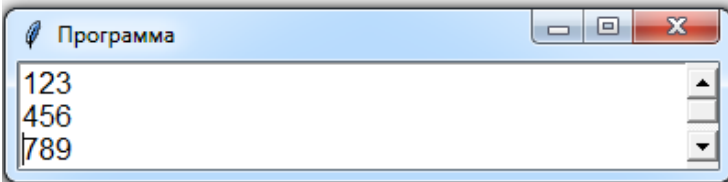
*Шкалы*

File Edit Format Run Options Window Help

```
from tkinter import *
```

```
prog = Tk()
prog.title('Программа')
t = Text(prog,width=40,height=3,font='14') # создание текстового поля
scr = Scrollbar(prog,command=t.yview) # полоса прокрутки по полю t верт оси yview
t.configure(yscrollcommand=scr.set) # конфигурация прокрутки

t.grid(row=0,column=0) # расположение виджета текст
scr.grid(row=0,column=1) # расположение виджета прокрутки
prog.mainloop() # инициализация окна
```

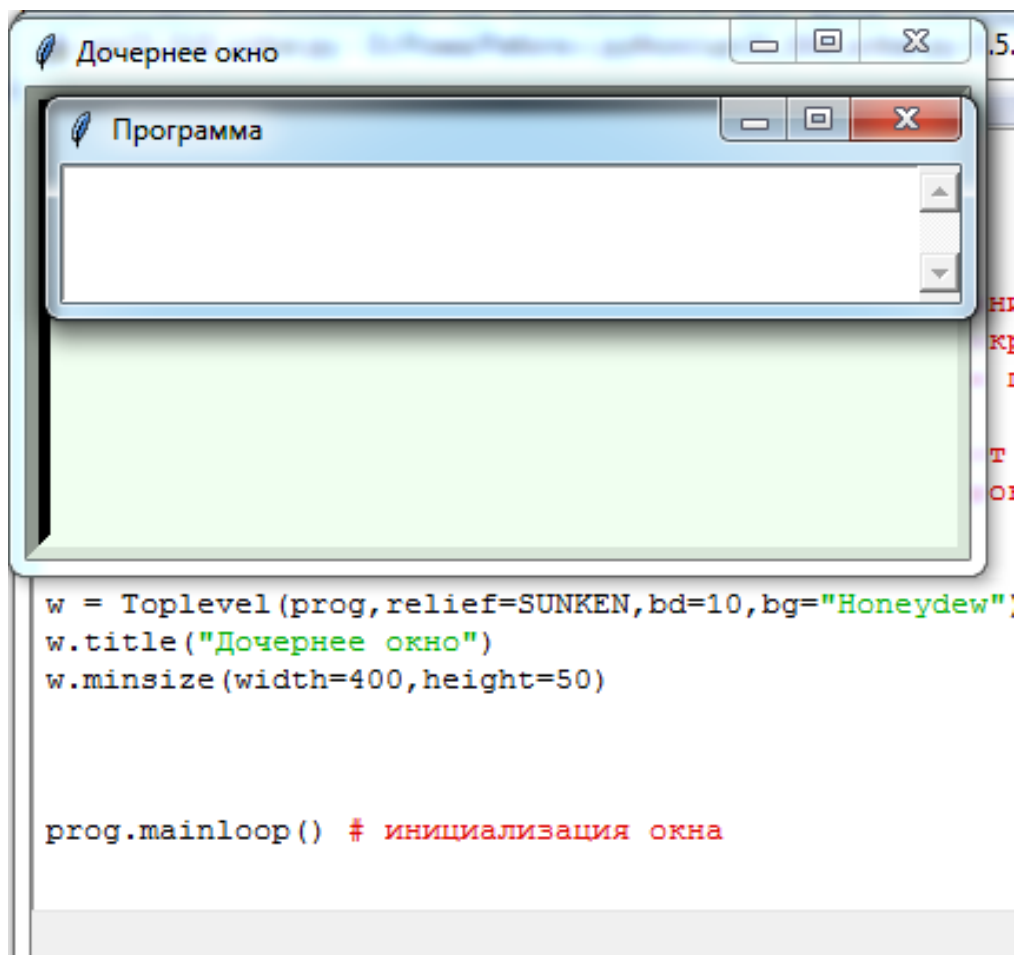


### *Полоса прокрутки*

### Упражнение 22

Используя опыт решения предыдущего упражнения, создайте приложение, состоящее из главного и двух дочерних окон. На каждом из трех окон должны располагаться один или два любых графических объекта.

Пример создания дочерних окон см. на рисунке ниже.



### Упражнение 23. Метод `bind`

1. Создайте приложение, в котором меняется размер фрейма в зависимости от того, какая из трех объектов-кнопок была нажата.
2. Напишите скрипт, генерирующий окно с меткой и текстовым полем. После ввода пользователем текста в поле и нажатия Enter, введенный текст должен отображаться в метке.

Пример связывания на рисунках ниже.

```

File Edit Format Run Options Window Help
from tkinter import * # импорт библиотеки
import string

def output(event):
    s = ent.get()
    if s == "1":
        tex.delete(1.0,END)
        tex.insert(END,"Обслуживание клиентов на втором этаже")
    elif s == "2":
        tex.delete(1.0,END)
        tex.insert(END,"Пластиковые карты выдают в соседнем здании")
    else:
        tex.delete(1.0,END)
        tex.insert(END,"Введите 1 или 2 в поле слева")

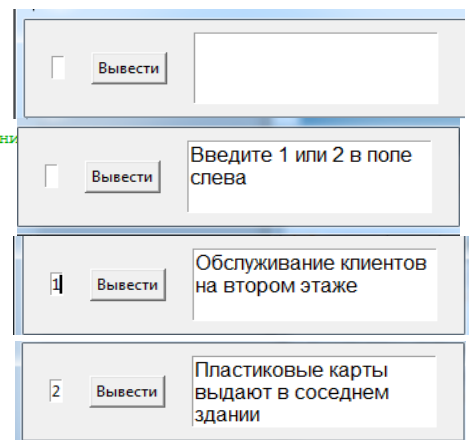
pr = Tk()
pr.title("Связывание функций")
ent = Entry(pr,width=1)
but = Button(pr,text="Вывести")
tex = Text(pr,width=20,height=3,font="12",wrap=WORD)

ent.grid(row=0,column=0,padx=20)
but.grid(row=0,column=1)
tex.grid(row=0,column=2,padx=20,pady=10)

but.bind("<Button-1>",output)

pr.mainloop()

```



Пример 1. `padx` и `pady` определяют количество пикселей от виджета до края рамки (или ячейки) по осям `x` и `y` соответственно.

```

File Edit Format Run Options Window Help
from tkinter import * # импорт библиотеки
import string
li = ["red","green"]
def color(event):
    fra.configure(bg=li[0])
    li[0],li[1] = li[1],li[0]

def outgo(event):
    pr.destroy()

pr = Tk()

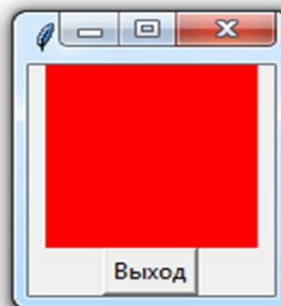
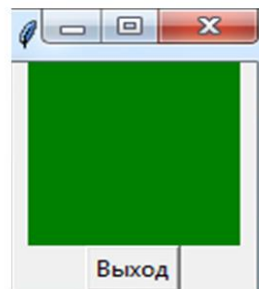
fra = Frame(pr,width=100,height=100,bd=20)
but = Button(pr,text="Выход")

fra.pack()
but.pack()

pr.bind("<Return>",color)
but.bind("<Button-1>",outgo)

pr.mainloop()

```



Пример 2

## Упражнение 23. События с мыши и клавиатуры

1. Напишите следующую программу. На главном окне находится несколько флажков и текстовое поле. При щелчке левой кнопкой мыши в пределах текстового поля в нем должны отображаться значения включенных флажки (появляться сообщение о том, какие флажки включены), при щелчке правой кнопкой мыши — значения выключенных флажков.
2. Напишите скрипт, генерирующий в окне два текстовых поля и рамку. Размер рамки можно менять с помощью вводимых значений в текстовые поля (определяют длину и ширину) и нажатии клавиши пробел на клавиатуре.

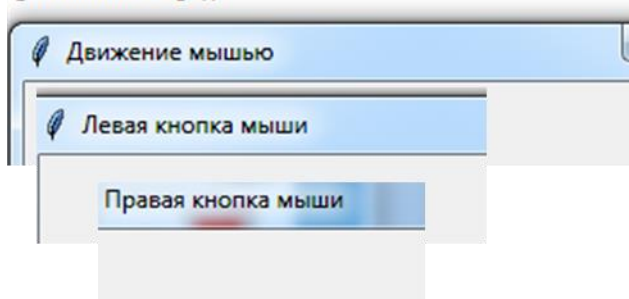
Пример кода представлен на рисунках ниже

```
from tkinter import * # импорт оконных
import string
def b1(event):
    pr.title("Левая кнопка мыши")
def b3(event):
    pr.title("Правая кнопка мыши")
def move(event):
    pr.title("Движение мышью")

pr = Tk()
pr.minsize(width = 400, height=100)

pr.bind('<Button-1>',b1)
pr.bind('<Button-3>',b3)
pr.bind('<Motion>',move)

pr.mainloop()
```



Пример событий с мыши



```

from tkinter import *

def exit_(event):
    root.destroy()
def caption(event):
    t = ent.get()
    lbl.configure(text = t)

root = Tk()

ent = Entry(root, width = 40)
lbl = Label(root, width = 80)

ent.pack()
lbl.pack()

ent.bind('<Return>',caption)
root.bind('<Escape>',exit_)

root.mainloop()

```

Пример событий с клавиатуры

## Упражнение 24. Работа с кнопками

1. Напишите программу с флажками, где значения ассоциированных переменных должны отображаться в метке (Label) через запятую.
2. Напишите программу, в которой пользователь может определить цвет рамки (Frame) с помощью шкалы (Scale).

Пример кода представлен наследующем рисунке

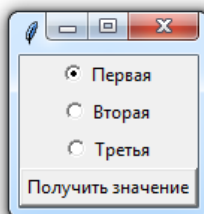
```

File Edit Format Run Options Window Help
from tkinter import * # импорт библиотеки
import string
root = Tk()
var=IntVar()
var.set(1)
rad0 = Radiobutton(root,text="Первая",variable=var,value=0)
rad1 = Radiobutton(root,text="Вторая",variable=var,value=1)
rad2 = Radiobutton(root,text="Третья",variable=var,value=2)
rad0.pack()
rad1.pack()
rad2.pack()

def display(event):
    v = var.get()
    if v == 0:
        print ("Включена первая кнопка")
    elif v == 1:
        print ("Включена вторая кнопка")
    elif v == 2:
        print ("Включена третья кнопка")

but = Button(root,text="Получить значение")
but.pack()
but.bind('<Button-1>',display)
root.mainloop()

```



## Радиокнопки

```

root = Tk()

var0=StringVar() # значение каждого флажка ...
var1=StringVar() # ... хранится в собственной переменной
var2=StringVar()
# если флажок установлен, то в ассоциированную переменную ...
# ... (var0, var1 или var2) заносится значение onvalue, ...
# ...если флажок снят, то - offvalue.
ch0 = Checkbutton(root,text="Окружность",variable=var0,
    onvalue="circle",offvalue="-")
ch1 = Checkbutton(root,text="Квадрат",variable=var1,
    onvalue="square",offvalue="-")
ch2 = Checkbutton(root,text="Треугольник",variable=var2,
    onvalue="triangle",offvalue="-")

lis = Listbox(root,height=3)
def result(event):
    v0 = var0.get()
    v1 = var1.get()
    v2 = var2.get()
    l = [v0,v1,v2] # значения переменных заносятся в список
    lis.delete(0,2) # предыдущее содержимое удаляется из Listbox
    for v in l: # содержимое списка l последовательно ...
        lis.insert(END,v) # ...вставляется в Listbox

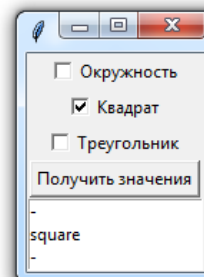
but = Button(root,text="Получить значения")
but.bind('<Button-1>',result)

ch0.deselect() # "по умолчанию" флажки сняты
ch1.deselect()
ch2.deselect()

ch0.pack()
ch1.pack()
ch2.pack()
but.pack()
lis.pack()

root.mainloop()

```



## Флажки

## Упражнение 25. Работа с меню

Задание 1. Создайте программу с меню, содержащим два пункта: Цвет и Размер. Пункт Цвет должен содержать три команды (Красный, Синий и Зеленый), меняющие цвет рамки на главном окне. Пункт Размер должен содержать две команды (500x500 и 700x400), изменяющие размер рамки.

Пример работы с меню и привязки функций представлен на рисунке.

```
from tkinter import * # импорт библиотеки
from tkinter.filedialog import *
import string
import fileinput
pr = Tk()
pr.title('Пример с меню')

def new_win():
    win = Toplevel(pr)

def close_win():
    pr.destroy()

def about():
    win = Toplevel(pr)
    lab = Label(win, text="Это просто программа- пример \n с меню в Tkinter")
    lab.pack()

def _open():
    op = askopenfilename()
    for l in fileinput.input(op):
        txt.insert(END, l)

def _save():
    sa = asksaveasfilename()
    letter = txt.get(1.0, END)
    f = open(sa, "w")
    f.write(letter)
    f.close()

men = Menu(pr) #создается объект Меню на главном окне
pr.config(menu=men) #окно конфигурируется с указанием меню для него

fmen = Menu(men) #создается пункт меню с размещением на основном меню (men)
men.add_cascade(label="Файл", menu=fmen) #пункту располагается на основном меню (men)
fmen.add_command(label="Новый", command=new_win) #формируется список команд пункта
fmen.add_command(label="Открыть", command=_open)
fmen.add_command(label="Сохранить", command=_save)
fmen.add_command(label="Выход", command=close_win)

sмен = Menu(men) #второй пункт меню
men.add_cascade(label="Помощь", menu=sмен)
```

*Первая часть*

```

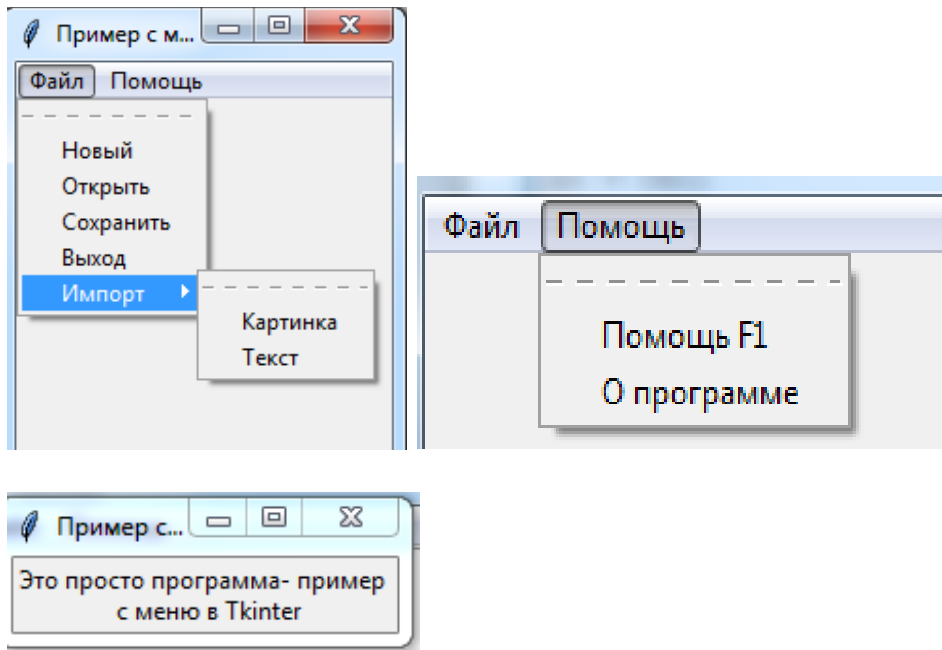
smen = Menu(men) #второй пункт меню
men.add_cascade(label="Помощь", menu=smen)
smen.add_command(label="Помощь F1")
smen.add_command(label="О программе", command=about)

nfmén = Menu(fmen)
fmen.add_cascade(label="Импорт", menu=nfmén)
nfmén.add_command(label="Картинка")
nfmén.add_command(label="Текст")

txt = Text(pr,width=40,height=15,font="12")
txt.pack()

```

*Продолжение предыдущего кода*



*Реализация кода*

Задание 2.

Пользуясь предыдущим примером, представленном на рисунках, добавьте условие после нажатия пункта Exit пользователю выводится окно с вопросом "сохранить или нет". В случае положительного ответа должна вызываться функция `_save` и только затем завершаться программа.

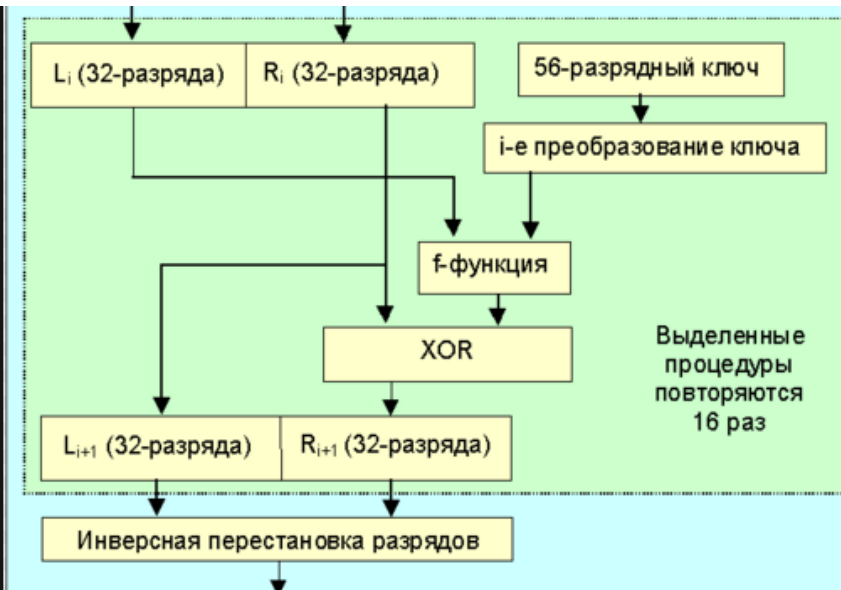
Если в текстовом поле что-то содержится, то при открытии файла оно не удаляется, а содержимое файла просто дописывается. Исправьте этот

недостаток (перед открытием файла содержимое текстового поля должно удаляться).

**Упражнение 26. Работа с графическими объектами Canvas (холст)**

Создайте изображения, представленные на рисунках

X	Y	X and Y	X or Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1



Пример кода представлен на рисунке

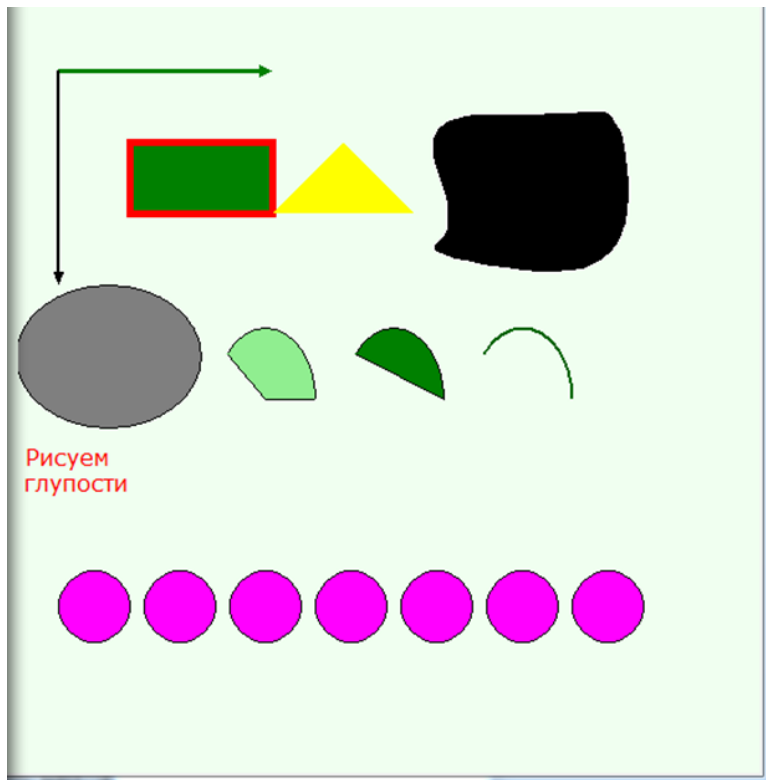
```

File Edit Format Run Options Window Help
from tkinter import *
myprog=Tk()

can = Canvas(myprog,width=500,height=500,bg="Honeydew",
             cursor="pencil", bd=20) # задание параметров виджета
can.create_line(50,50,200,50,width=3,fill="green", arrow=LAST) # линия
can.create_line(50,50,50,200,width=2,arrow=LAST)
x = 100
y = 100
can.create_rectangle(x,y,x+100,y+50,fill="green",outline="red",width=5)
# прямоугольник

can.create_polygon([250,100],[200,150],[300,150],fill="yellow")
can.create_polygon([300,80],[400,80],[450,75],[450,200],
                  [300,180],[330,160],outline="white",smooth=1)
# многоугольник
can.create_oval([20,200],[150,300],fill="gray50")
# эллипс задается прямоугольником
can.create_arc([160,230],[230,330],start=0,extent=140,fill="lightgreen") # сектор
can.create_arc([250,230],[320,330],start=0,extent=140,
               style=CHORD,fill="green") # сегмент
can.create_arc([340,230],[410,330],start=0,extent=140,
               style=ARC,outline="darkgreen",width=2) # дуга
can.create_text(20,330,text="Рисуем \n глупости",
               font="Verdana 12",anchor="w",justify=CENTER,fill="red")
x=50
while x < 450:
    can.create_oval(x,400,x+50,450, fill='magenta')
    x = x + 60
# anchor - привязка w - запад, варианты: n, ne, e, se, s, sw, w, nw.
can.pack()
myprog.mainloop()

```



**Упражнение 27. Работа с графическими объектами Canvas (холст).  
Модификация**

Создайте перемещение какого-либо объекта по холсту. Анимацию можно создать с помощью методов удаления и перемещения объектов, при этом можно воспользоваться циклом `while`, в теле которого с помощью метода `delete` удаляется старая фигура, а с помощью `move` рисуется такая же на новом месте.

Пример использования функций и тегов приведен на рисунке

```

File Edit Format Run Options Window Help
from tkinter import *
myprog=Tk()

def moove(event):
    can.move(rect,0,150) # модификатор
    can.itemconfig(trian,outline="red",width=3) # модификатор
    can.coords(oval,300,200,450,450) # модификатор

def color(event):
    can.itemconfig('group1',fill="yellow",width=5)

def clean(event):
    can.delete(oval)

can = Canvas(width=500,height=460,bg='Honeydew') # идентификатор

can.pack() # размещение

oval = can.create_oval(30,10,130,80,tag='group1') # идентификатор
rect = can.create_rectangle(180,10,280,80,tag='group1') # идентификатор
trian = can.create_polygon(330,80,380,10,430,80, fill='grey80', outline="black")
# идентификатор

can.bind('<Button-1>', moove) # реакция на событие
can.bind('<Button-3>', color) # реакция на событие
can.bind('<Double Button-1>', clean) # реакция на событие

myprog.mainloop() #инициация

```

Внимание, события привязывайте к кнопкам мыши. Можно сделать с трассировкой, оставляя предыдущие изображения.

Для выполнения анимации лучше использовать модуль [turtle](#).

## Упражнение 28. Построение графиков

Постройте следующие графики. Диапазон выберете произвольно.

$$1 + \sin^2(n) \cdot 3^3 + \arctg(8 \cdot \pi / 3)$$

$$\frac{\ln^2(n^3)}{1 + n^2} \bmod(2n) + 4 \cos^2(n) + \arctg(9 \cdot \pi / 2)$$

File Edit Format Run Options Window Help

```
# -*- coding: utf-8 -*-
from tkinter import *
import math

#
tk=Tk()
tk.title("Graph")
#
button=Button(tk,text='Заккрыть', command=quit)
button.pack()
#
canvas=Canvas(tk)
canvas["height"]=360
canvas["width"]=480
canvas["background"]="#eeeeff"
canvas["borderwidth"]=2
canvas.pack()
#
canvas.create_text(20,10,text="20,10")
canvas.create_text(460,350,text="460,350")
#
points=[]
ay=150
y0=150
x0=50
x1=470
dx=10
#
for n in range(x0,x1,dx):
    y=y0-ay*math.cos(n*dx)
    pp=(n,y)
    points.append(pp)
#
canvas.create_line(points,fill="blue",smooth=1)
#
y_ave=[]
yy=(x0,0)
y_ave.append(yy)
yy=(x0,y0+ay)
y_ave.append(yy)
canvas.create_line(y_ave,fill="black",width=2)
```

```
#
x_ave=[]
xx=(x0,y0)
x_ave.append(xx)
xx=(x1,y0)
x_ave.append(xx)
canvas.create_line(x_ave,fill="black",width=2)
#
tk.mainloop()
```

