root.mainloop()

В практической части работы будут подробно рассмотены различные компоненты и функции данного модуля.

Оборудование и материалы.

Персональный компьютер, среда разработки Python.

Указания по технике безопасности:

Соответствуют технике безопасности по работе с компьютерной техникой.

Задания

Гораздо более серьёзными возможностями, чем модуль turtle, обладает модуль Tkinter. Основное предназначение этого модуля — создание графических интерфейсов (GUI — Graphical User Interface) для программ на Python. Но благодаря наличию элемента графического интерфейса (или, как говорят, "виджета") canvas ("холст") Tkinter можно применять для рисования на основании координат, рассчитанных по формулам, используя доступные элементы векторной графики — кривые, дуги, эллипсы, прямоугольники и пр.

Рассмотрим задачу построения графика некоторой функции по вычисляемым точкам с помощью Tkinter.

Поскольку Tkinter позволяет работать с элементами GUI, создадим окно заданного размера, установим для него заголовок и цвет фона "холста", а также снабдим окно программной "кнопкой". На "холсте" определим систему координат и нарисуем "косинусоиду".

```
importtkinter
importmath
#
tk=tkinter.Tk()
tk.title("Sample")
#
button=tkinter.Button(tk)
button["text"]="Закрыть"
button["command"]=tk.quit
button.pack()
#
canvas=tkinter.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_axe=[]
yy=(x0,0)
y_axe.append(yy)
yy=(x0, y0+ay)
y_axe.append(yy)
canvas.create_line(y_axe, fill="black", width=2)
#
x_axe=[]
xx=(x0,y0)
x_axe.append(xx)
xx=(x1,y0)
x_axe.append(xx)
canvas.create_line(x_axe, fill="black", width=2)
#
tk.mainloop()
```

Посмотрим на результат (рис. 15.1), и разберём текст примера.

Итак, первые две строки программы понятны —подключаются библиотеки. Поскольку в примере должны использоваться тригонометрические функции, необходимо подключить модуль math.

Затем создаётся так называемое "корневое" окно — говоря научным языком, "экземпляр интерфейсного объекта Tk", который представляет собой просто окно без содержимого. Для этого окна устанавливается значение свойства title (создаётся заголовок).

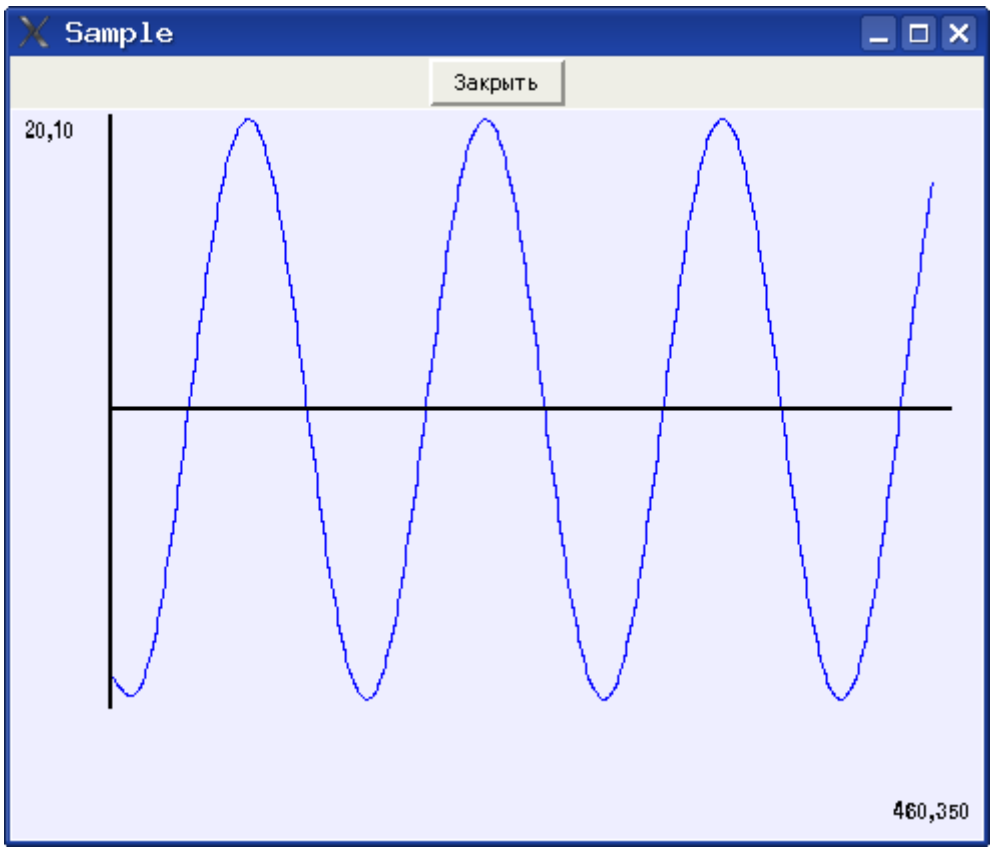

Рис. 15.1. Окно Tkinter с кнопкой и графиком

Далее начинается заполнение окна интерфейсными объектами ("виджетами" — widgets). В данном примере используется два объекта — кнопка и "холст". Для размещения объекта в окне используется метод pack(), а порядок объектов определяется порядком выполнения этой функции.

Кнопка создаётся как экземпляр объекта Button модуля Tkinter, связанный с "корневым" окном. Для кнопки можно установить текст надписи (свойство text) и связать кнопку с выполнением какой-либо команды (функции или процедуры), установив значение свойства command.

В приведённом примере кнопка связана с командой закрытия окна и прекращения работы интерпретатора, однако ничто не мешает также закрывать окно нашего "приложения" обычным образом — с помощью стандартной кнопки закрытия окна в верхнем правом углу.

После создания и размещения кнопки создаётся холст. Для элемента (объекта) canvas указываются высота, ширина, цвет фона и отступ от границ окна (таким образом, размеры окна получаются несколько больше, чем размеры объекта canvas). Размеры окна автоматически подстраиваются так, чтобы обеспечить размещение всех объектов (элементов интерфейса).

Прежде чем приступать к рисованию, исследуем систему координат. Поскольку размеры окна уже нами заданы, полезно определить, где находится точка с координатами (0, 0). Как видно из попыток вывести значения координат с помощью метода canvas.create_text(), начало координат находится в верхнем левом углу холста.

Теперь, определившись с координатами, можно выбрать масштабные коэффициенты и сдвиги и сформировать координаты точек для рисования кривой.

При использовании метода canvas. create_line() в качестве координат требуется список пар точек (кортежей) (x,y). Этот список формируется в цикле с шагом dx.

Для линии графика устанавливаются цвет и режим сглаживания. Сглаживание обеспечивает некоторую "плавность" кривой. Если его убрать, линия будет состоять из отрезков прямых. Кроме того, для линий можно устанавливать толщину, как это показано на примере осей координат.

Моделирование математических функций

Пусть требуется построить график функции, выбираемой из заданного списка. Здесь потребуется уже использование дополнительных интерфейсных элементов модуля Tkinter, а также создание собственных (пользовательских) процедур или функций для облегчения понимания кода.

Результат решения задачи (вариант внешнего вида "приложения") показан на рис. 3.5.

Для выбора вида математической функции используется раскрывающийся список, после выбора вида функции и нажатия на кнопку "Нарисовать" на "холсте" схематически изображается график этой функции. Кнопка "Закрыть" закрывает наше "приложение". Теперь посмотрим на код.

```
# -*- coding: utf-8 -*-
 import Tkinter
 import math
#
# Пользовательские процедуры
 def plot_x_axe(x0, y0, x1):
  x_axe=[]
  xx=(x0, y0)
  x_axe.append(xx)
  xx=(x1, y0)
  x_axe.append(xx)
  canvas.create_line(x_axe, fill="black", width=2)
#
 def plot_y_axe(x0, y0, y1):
  y_axe=[]
  yy=(x0, y1)
  y_axe.append(yy)
  yy=(x0, y0)
  y_axe.append(yy)
  canvas.create_line(y_axe, fill="black", width=2)
#
 def plot_func0(x0, x1, dx, y0, y1):
  x0i=int(x0)
  x1i=int(x1)
  y0i=int(y0)
  y1i=int(y1)
  a=y1
  b=(y0-y1) / (x1-x0)
  points=[]
          for x in range(x0i, x1i, dx):
             y=int(a+b*x)
             pp=(x, y)
             points.append(pp)
             #
             canvas.create_line(points, fill="blue", smooth=1)
             plot_y_axe(x0i, y0i, y1i)
             plot_x_axe(x0i, y0i, x1i)
             #
 def plot_func1(x0, x1, dx, y0, y1):
```

```
    x0i=int(x0)
    x1i=int(x1)
    y0i=int(y0)
    y1i=int(y1)
    a=y0
    b=y0-y1
    points=[]
            for x in range(x0i, x1i, dx):
                y=int(a-y1i*b/ x)
                pp=(x, y)
                points.append(pp)
    #
    canvas.create_line(points, fill="blue", smooth=1)
    plot_y_axe(x0i, y0i, y1i)
    plot_x_axe(x0i, y0i, x1i)
#
  def plot_ func2(x0, x1, dx, y0, y1):
    x0i=int(x0)
    x1i=int(x1)
    y0i=int(y0)
    y1i=int(y1)
    a=(y0-y1)/(15*x1)
    b=1+((y0-y1)/(x1-x0))
    points=[]
            for x in range(x0i, x1i, dx):
                y=y0i-int(a*(x-x0i)**b)
                pp=(x, y)
                points.append(pp)
    #
    canvas.create_line(points, fill="blue", smooth=1)
    plot_y_axe(x0i, y0i, y1i)
    plot_x_axe(x0i, y0i, x1i)
#
  def plot_ func3(x0, x1, dx, y0, y1):
    x0i=int(x0)
    x1i=int(x1)
    y0i=int(y0)
    y1i=int(y1)
    ay=150
    y0i=150
    points=[]
            for x in range(x0i, x1i, dx):
                y=y0i-ay*math.cos(x*dx)
                pp=(x, y)
                points.append(pp)
    #
    canvas.create_line(points, fill="blue", smooth=1)
    plot_y_axe(x0i, 0, y0i+ay)
    plot_x_axe(x0i, y0i, x1i)
#
  def DrawGraph():
    fn=func.get()
    f=fn[0]
    x0=50.0
    y0=250.0
    x1=450.0
    y1=50.0
```

```
   dx=10
#
            if f=="0":
               canvas.delete("all")
               plot_ func0(x0, x1, dx, y0, y1)
            elif f=="1":
               canvas.delete("all")
               plot_ func1(x0, x1, dx, y0, y1)
            elif f=="2":
               canvas.delete("all")
               plot_ func2(x0, x1, dx, y0, y1)
            else:
               canvas.delete("all")
               plot_ func3(x0, x1, dx, y0, y1)
#
# Основная часть
tk=Tkinter.Tk()
tk.title(" Sample ")
# Верхняя часть окна со списком и кнопками
menuframe=Tkinter.Frame(tk)
menuframe.pack({"side":"top", "fill":"x"})
# Надписьдлясписка
lbl=Tkinter.Label(menuframe)
lbl["text"]="Выбор:"
lbl.pack({"side":"left"})
# Инициализация и формирование списка
func=Tkinter.StrigVar(tk)
func.set('0 y=Ax+B')
#
fspis=Tkinter.OptionMenu(menuframe, func,
   '0 y=Ax+B',
   '1 y=A+B/ x',
   '2 y=Ax^B',
   '3 y=A*cos(Bx)')
fspis.pack({"side":"left"})
# Кнопка управления рисованием
btnOk=Tkinter.Button(menuframe)
btnOk["text"]="Нарисовать"
btnOk["command"]=DrawGraph
btnOk.pack({"side":"left"})
# Кнопказакрытияприложения
button=Tkinter.Button(menuframe)
button["text"]="Закрыть"
button["command"]=tk.quit
button.pack({"side":"right"})
# Областьрисования (холст)
canvas=Tkinter.Canvas(tk)
canvas["height"]=360
canvas["width"]=480
canvas["background"]="#eeeeff"
canvas["borderwidth"]=2
canvas.pack({"side":"bottom"})
tk.mainloop()
```

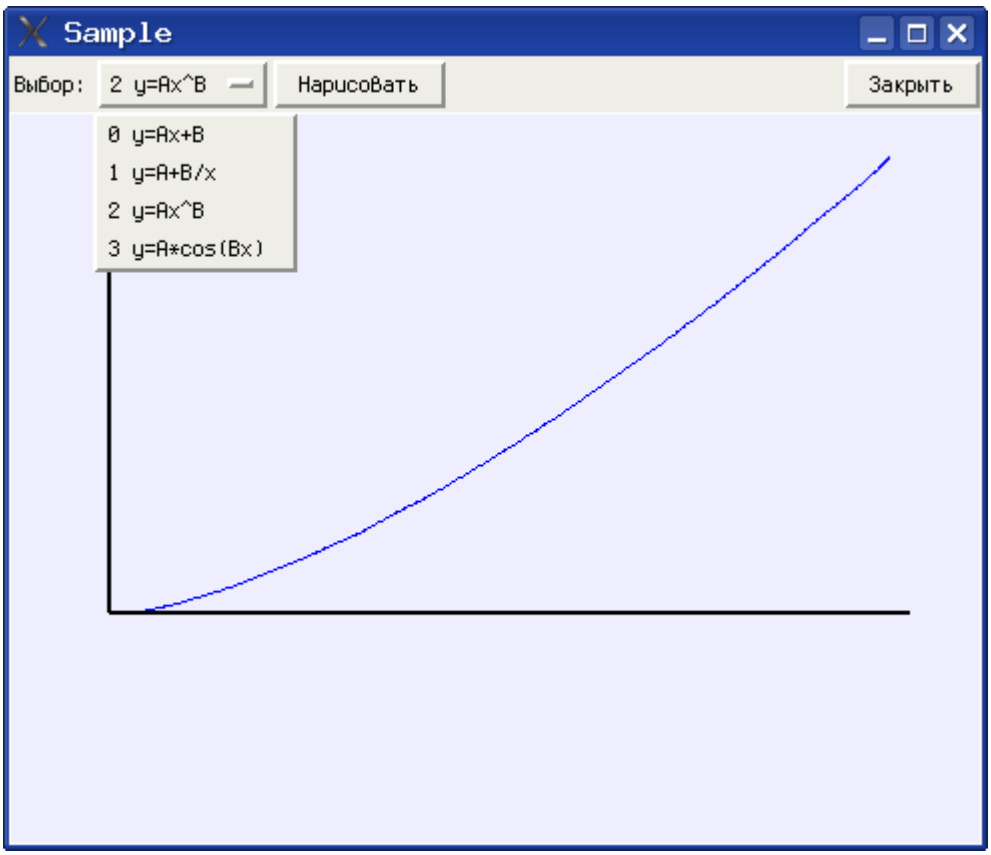

Рис. 15.2. Построение графика для функции, выбираемой из списка

Основная часть программы (интерфейсная) начинается с момента создания корневого окна (инструкция tk=Tkinter.Tk()). В этом окне располагаются два интерфейсных элемента — рамка (Frame) и холст (canvas). Рамка является "контейнером" для остальных интерфейсных элементов — текстовой надписи (метки — Label), раскрывающегося списка вариантов (OptionMenu) и двух кнопок. Как видно, кнопка закрытия стала объектом рамки, а не корневого окна, но по-прежнему закрывает всё окно.

Для получения нужного расположения элементов метод pack() используется с указанием, как именно размещать элементы интерфейса (к какой стороне элемента-контейнера их нужно "прижимать").

Есть некоторые тонкости в создании раскрывающегося списка. Для успешного выполнения этой операции нужно предварительно сформировать строку (а точнее, объект Tkinter.StringVar()) и определить для этого объекта значение по умолчанию (это значение будет показано в только что запущенном приложении). Затем при определении объекта OptionMenu() список значений дополняется. При выборе элемента списка изменяется значение именно этой строки и для дальнейшей работы нужно его анализировать, что и делается в процедуре DrawGraph().

"Вычислительная" часть, а именно, все процедуры и функции, обеспечивающие вычисления координат точек и рисование линий, вынесена в начало текста программы.

Определение каждой пользовательской подпрограммы обеспечивается составным оператором def. Поскольку эти подпрограммы занимаются только рисованием, они не возвращают никаких значений (т.е. результаты выполнения этих подпрограмм не присваиваются никаким переменным).

Собственно подпрограмма рисования графика DrawGraph() вызывается при нажатии кнопки "Нарисовать", и имя этой подпрограммы является командой, которая сопоставляется кнопке.

Эта подпрограмма берёт значение из списка (метод get()), выбирает первый символ получившейся строки и в зависимости от этого символа вызывает другие подпрограммы для построения конкретных графиков с установленными масштабными коэффициентами.

Перед рисованием следующего графика математической функции холст очищается командой canvas.delete("all").

Для построения графика каждой функции вычисляются собственные масштабные коэффициенты, поэтому их вычисление включено в код соответствующей подпрограммы. Кроме того, для графика нужны целые значения координат, поэтому в каждой подпрограмме выполняются соответствующие преобразования с помощью функции int().

Для каждого графика требуется нарисовать оси, и действия по рисованию осей также вынесены в отдельные подпрограммы.

Таким образом, оказывается, что программу нужно читать "с конца", и писать тоже.

Контрольные вопросы

1. Импорт библиотеки tkinter.
2. Создание главного окна.
3. Создание виджетов, установка их свойств, определение событий, определение обработчиков событий.
4. Расположение виджетов на главном окне, отображение главного окна.

Список литературы, рекомендуемый к использованию по данной теме:

1. Орлов, С. А. Программная инженерия. Технологии разработки программного обеспечения : учебник / С.А. Орлов. - 5-е изд., обновл. и доп. - СПб. : Питер, 2017. - 640 с.
2. Гагарина, Л. Г. Современные проблемы информатики и вычислительной техники : [учеб. пособие] / Л.Г. Гагарина, А.А. Петров. - М. : Форум, 2016. - 368 с.
3. Шкляр, М. Ф. Основы научных исследований : учеб. пособие / М.Ф. Шкляр. - 6-е изд. - М. : Дашков и Ко, 2016. - 208 с.
4. Михеева, Е. В. Информационные технологии в профессиональной деятельности : учеб. пособие / Е.В. Михеева. - 14-е изд., стер. - М. : Академия, 2016. - 384 с.
5. Любанович Билл. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.

Лабораторная работа 16. Моделирование физического явления.

Цель работы:

Изучить понятия моделирования физических процессов. Применение физического моделирования для выявления закономерностей изучаемого явления, проверки правильности и границ применимости эксперимента. Модели тела, брошенного под углом у горизонту. Поиск угла для достижения максимальной дальности на модели. Верификация моделей.

Компетенции:

Код	Формулировка:
ОК-5	быть готовым работать с информацией в различных формах, использовать для ее получения, обработки, передачи, хранения и защиты современные компьютерные технологии