7. Z.D. Abbasov, Dördüncü tərtib xüsusi törəməli difernsial tənlik üçün bir qarışıq məsələ, Elmi Xəbərlər Jurnalı,Gəncə-2019(1), səh.16-19

СМЕЩЕНИЕ НЕОДНОРОДНОГО УПРУГОГО СТЕРЖНЯ ВЫТЯНУТОГО ВПРАВО ИЗ КОНЕЧНОГО ЧИСЛА ТОЧКИ

Аббасов Зафар Думан

Резюме

An example of the group of hyperbolic type equations is the solution to the problem under the conditions of additional parameters in order to define the dance process as one-valued. transverse oscillation of a string and an elastic rod. It is important to find a non-trivial
В работе рассматривалась задача определения перемещения неоднородного стержня длины l, соединенного концами по оси Ox, при колебательном движением под действием сосредоточенной силы[1,3].

DISPLACEMENT OF AN INHOMOGENEOUS ELASTIC ROD STRETSHEDD TO THE RIGHT FROM A FINITE NUMER OF POINTS

Abbasov Zafar Duman

Summary

An example of the group of hyperbolic type equations is the transverse oscillation of a string and an elastic rod. It is important to find a non-trivial solution to the problem under the conditions of additional parameters in order to define the dance process as one-valued.

İn the study, the problem of determining the displacement of a nonhomogeneus rod of length l which is connested at the ends along the Ox axis and consists of a combination of n uniform rods under the influence of a concentrated force, was considered[1,3].

Rəyçi dosent Oruc Hüseynov
Riyazi analiz kafedrasının 15 mart 2023-cü il tarixli iclasın 03 saylı protokoluy
Daxil olma tarixi 17 mart 2023-cü il

UOT:372.0:002

TKİNTER. PYTHONDA GUİ-NİN PROQRAMLAŞDIRMASI

Babayeva Şəkər Maarif qızı
Gəncə Dövlət Universiteti
shakarbabayeva@mail.ru

Xülasə: *Hal-hazırda, istifadəçi üçün yaradılan demək olar ki, bütün proqramlar GUI-yə malikdir. Python üçün bir çox GUI kitabxanaları yaradılmışdır. Python interpretatorunun quraşdırma faylı adətən standart kitabxananın bir hissəsi kimi tkinter paketini kompüterə yükləyir. Cari məqalə tkinter kitabxanasının istifadəsilə qrafik interfeysli proqramların yaradılmasına həsr edilmişdir.*

Qrafik istifadəçi interfeysi olan proqramlar hadisə yönümlü olur. Hadisə yönümlü proqramlaşdırma hadisəyə əsaslanır. Yəni proqram kodunun bu və ya digər hissəsi yalnız bu və ya digər hadisə baş verəndə icra olunmağa başlayır. Məqalədə qrafik interfeysli proqramın yaradılması mərhələləri şərh edilmişdir və bunlara üyğun bir məsələ üçün vidjetlərdən istifadə etməklə proqram tərtib edilmişdi. Sonda bir neçə və ya çoxlu oxşar qrup obyektlərin olmasını tələb edən məsələlər üçün proqramın tərtibində obyekt yönümlü yanaşmanın tətbiq edilməsi əlverli olduğu bir məsələ üzərində izah edilir.

Açar sözlər: qrafik istifadəçi interfeysi, sinif, obyekt, xassələr, metodlar, vidjetlər, hadisələr, emaledici funksiya **.**

Ключевые слова: графический интерфейс пользователя, класс, объект, свойства, методы, виджеты, события, функция-обработчик.

Key words: graphical user interface, class, oject, properties, methods, widgets, events, prosessing function **.**

Tkinter (ingiliscə Tk interface) Tk alətlərinə əsaslanan (GNU/Linux və digər UNIX-ə bənzər sistemlərdə geniş yayılmış, həmçinin Microsoft Windows-a köçürülmüş), kros-platformalı hadisə yönümlü qrafik kitabxanadır. Tkinter Stin Lamholt (Steen Lumholt) və Qvido van Rossum (Guido van Rossum) tərəfindən yazılmışdır və Python standart kitabxanasına daxildir. Tk kitabxanası Con Osterhut (John Ousterhout) tərəfindən yaradılmışdır.

Tkinter-Tk kitabxanası ilə işləmək üçün nəzərdə tutulmuş Python paketidir. Tk kitabxanasında qrafik istifadəçi interfeysi (GUI) komponentləri var. Bu kitabxana Tcl proqramlaşdırma dilində yazılmışdır. Kitabxana pəncərəli qrafik interfeysdən (GUI) istifadə edərək proqramda dialoqların təşkili üçün nəzərdə tutulmuşdur. Tkinter çox güclü bir kitabxanadır. Əgər sizdə artıq Python quraşdırılıbsa, siz Python ilə gələn inteqrasiya edilmiş IDE (Integrated Development Environment -İnteqrasiya edilmiş proqramlaşdırma mühiti) olan IDLE-dən istifadə edə bilərsiniz. Maraqlıdır ki, bu IDE özü də tkinterdən istifadə edərək yazılmışdır.

Qrafik istifadəçi interfeysi (GUI -Graphical User Interface)-klaviatura və siçan istifadə edərək qarşılıqlı əlaqədə olduğumuz proqramın örtüyüdür. GUİ dedikdə, proqramı açdığınız zaman ekranda gördüyünüz bütün pəncərələr, düymələr, mətn daxiletmə sahələri, sürüşdürmə düymələri, siyahılar, radio düymələr, təsdiq etmə (bayraqlar) qutuları və s. nəzərdə tutulur. Onların vasitəsilə siz proqramla qarşılıqlı əlaqədə olursunuz və ona nəzarət edirsiniz. Bütün bu interfeys elementləri vidjetlər (widgets) adlanır.

Bir çox GUI kitabxanaları (Kyvi, PyQT, PySide, WxPython və s.) var, onların arasında Tk ən populyar alət deyil (baxmayaraq ki, onunla bir çox layihələr yazılmışdır). Tk Python üçün standart olaraq seçilmişdir. Python interpretatorunun quraşdırma faylı adətən standart kitabxananın bir hissəsi kimi ***tkinter*** paketini kompüterinizə yükləyir (artıq tkinteri əlavə yükləməyə eytiyac olmur).

Tkinter Python-dan Tcl-ə tərcüməçi kimi düşünülə bilər. Siz Python-da proqram yazırsınız və ***tkinter*** modulu kod təlimatlarınızı Tk kitabxanasının başa düşdüyü Tcl dilinə çevirir.

Kitabxanada ümumi qrafik komponentlər (vidjetlər) var:

Toplevel/Tk – Üstsəviyyə pəncərəsi (kök vidjeti).

Frame – Çərçivə. Vidjetləri qruplaşdırmaq üçün istifadə edilir. Digər vizual komponentləri ehtiva edir.

Label – Etiket. Bəzi mətn və ya qrafikləri göstərir.

Entry – Giriş. Bir sətirli mətn daxiletmə sahəsi.

Text – Formatlaşdırılmış mətn daxiletmə sahəsi. Müxtəlif üslublardan istifadə edərək mətni göstərməyə, redaktə etməyə və formatlamağa, həmçinin mətnə şəkilləri və pəncərələri daxil etməyə imkan verir.

Canvas – Kətan. Düzbucaqlılar, ellipslər, xətlər, həmçinin mətn, şəkillər və pəncərələr kimi qrafik primitivləri göstərmək üçün istifadə edilə bilər.

Button – Düymə. Əmr və digər əməliyyatları yerinə yetirmək üçün sadə düymə.

Radiobutton – Radio düyməsi (dəyişdirici açar). Müəyyən dəyişənin alternativ qiymətlərindən birini təmsil edir. Adətən qrupda işləyir. İstifadəçi bir seçim etdikdə, eyni qrupda əvvəl edilmiş element seçimi ləğv edilir.

Checkbutton – Bayraq düyməsi. İki vəziyyətə malik olan düymə, basıldıqda vəziyyəti əksinə dəyişir. Radiobuttona bənzəyir, lakin çoxlu seçim etməyə imkan verir. Hər bir vidjet obyekti üçün ayrıca dəyişən təyin edir.

Scale – Sürgü ilə miqyaslama. Sürgünü hərəkət etdirməklə rəqəmsal qiymət təyin etməyə imkan verir.

Listbox – Siyahı qutusu. İstifadəçinin bir və ya bir neçə elementi seçə biləcəyi siyahını göstərir.

Scrollbar – Sürüşdürmə çubuğu. Bəzi digər komponentləri sürüşdürmək üçün onlarla birlikdə istifadə edilə bilər.

Menyu – Menyu. Üzə çıxan (popup) və açılan (pulldown) menyuların təşkili üçün xidmət edir.

Menubutton – Menyu düyməsi. Açılan menyunu təşkil etmək üçün istifadə olunur.

Message – Mesaj. Label-ə bənzəyir, lakin uzun sətrləri bükməyə və asanlıqla ölçüsünü dəyişməyə imkan verir.

Əlbəttə ki, bu vidjetlərin tam siyahısı deyil. Vidjet sinif oldiğuna görə hər vidjetin xassələri və metodları var. İixtiyari vidjetdən düzğün istifadə etmək üçün onun xassələri və metodları ilə tanış olmaq lazımdır (köməkçi məlumatı İnternetdə bu ünvanda almaq olar: https://ru.wikibooks.org/wiki/GUI_Help/Tkinter_book və
ya https://www.russianlutheran.org/python/life/life.htm).

Qrafik istifadəçi interfeysi olan proqramlar hadisə yönümlü olur. Hadisə yönümlü proqramlaşdırma hadisəyə əsaslanır. Yəni proqram kodunun bu və ya digər hissəsi yalnız bu və ya digər hadisə baş verəndə icra olunmağa başlayır.

Hadisə yönümlü proqramlaşdırma obyekt yönümlü və struktur proqramlaşdırmaya əsaslanır. Öz siniflərimizi və obyektlərimizi yaratmasaq da, quraşdırılmışlardan istifadə edirik. Bütün vidjetlər quraşdırılmış (daxili) siniflər tərəfindən yaradılan obyektlərdir.

Hadisələr fərqlidir: vaxt faktoru işlədi, kimsə siçanı kliklədi (düyməsini sıxdı) və ya Enter düyməsini basdı, yazmağa başladı, radio düymələrini dəyişdi, səhifəni aşağı sürüşdürdü və s. Belə bir hadisə baş verdikdə, müvafiq emaledici yaradılıbsa, proqramın müəyyən bir hissəsi işə salınır, bu da hansısa nəticəyə gətirib çıxarır.

Tkinter Python modulu üçün standart olan aşağıdakı üsullardan biri ilə idxal olunur:

• import tkinter
• from tkinter import *
• import tkinter as tk

Lazım gələrsə, TkVersion sabiti vasitəsilə Tk-nin hansı versiyası quraşdırılmış olduğunu tapa bilərik:

```
>>> from tkinter import *
>>> TkVersion
8.6
```

GUI proqramının yaradılması aşağıdakı ardıcılıqla yerinə yetirilir:

1. Əsas pəncərə yaradılır.
2. Vidjetlər yaradılır və onların xassələrinin konfiqurasiyası müəyyənləşdirilir.
3. Hadisələri, yəni proqramın nəyə cavab verəcəyini müəyyənləşdirilir.
4. Hadisə emalediciləri, yəni proqramın hadisəyə necə reaksiya verəcəyi təsvir edilir.
5. Vidcetlər əsas pəncərədə yerləşdirilir.
6. Hadisələr emaledicilərin dövrəsi başladılır.

Ardıcıllıq mütləq belə olmur, lakin ilk və sonuncu mərhələlər (addımlar) həmişə öz yerlərində qalır.

Müasir əməliyyat sistemlərində hər hansı bir istifadəçi proqramı əsas pəncərə adlandırıla bilən pəncərəyə daxil edilir, çünki bütün digər vidjetlər orada yerləşir. Üst

səviyyəli pəncərə obyekti tkinter modulunun Tk sinfindən yaradılmışdır. Obyektlə əlaqəli dəyişənə çox vaxt *root* (kök) deyilir:

```
root = Tk()
```

Düymə yaradılmasının sintaksisi:

```
name = Button(window)
```

name – düymənin adı, window – düymənin yerləşəcəyi pəncərənin adı.

Adətən mötərizələr arasında pəncərənin adından (window-dan) sonra düymənin xassələrinin adları (properties) və onların qiymətləri (options – ing. seçimlər) yazılır. Bunlar yazılmasa, susmaqla təyin olunan adlandırılmış parametrlər istifadə olunur. Məsələn:

```
bt = Button(root, text="ok")  # üstündə ok yazılmış düymə yaradılır.
button = Button(root, text="Press", fg=" red", bg="white") # üstündə qırmızı (düymə sıxılmamış olanda) rəngdə “Press” mətni yazılmış ağ (düymə sıxılmamış olanda) rəngdə düymə yaradılır.
b=Button(text='ok', width=10, height=3, activeforeground='green')  # üstündə yaşıl (düymə sıxılmış olanda) rəngdə “ok” mətni yazılmış, eni 10, hündürlüyü 3 piksel olan düymə yaradılır
```

Digər vidjetlərin (Entry, Label və s.) yaradılması sintaksisi oxşardır.

Tətbiq pəncərəsində mətn sahəsi, etiket və düymə yerləşdirək. Bu obyektlər tkinter modulunun Entry, Label və Button siniflərindən yaradılmışdır. Bu siniflərin konstruktorlarına arqumentlər ötürməklə onların bəzi xassələrini dərhal konfiqurasiya edəcəyik:

```
et = Entry(root, width=40)
bt = Button(root, text=" Əlifba sırası iə nizamla ")
lb = Label(root, width=40, bg='black', fg='white')
```

Vidcet konstruktorunun ilk arqumenti ana vidjetdir, yəni yaradılan vidjetin yerləşəcəyi vidjetdir. GUI elementləri birbaşa əsas pəncərəyə yerləşdirildikdə, əcdad (parent) göstərilmiyə bilər. Yəni, nümunəmizdə root-u silə bilərik:

```
et = Entry(width=40)
bt = Button(text=" Əlifba sırası iə nizamla ")
lb = Label(width=40, bg='black', fg='white')
```

Bununla belə, vidjetlər mütləq “kökdə” yerləşmir. Onlar digər vidjetlərdə yerləşə bilər, bu halda “kökün” obyekti (master) göstərilməlidir.

Tutaq ki, proqramda istifadəçinin mətn sahəsinə daxil etdiyi mətn düyməni basdıqda sözlər siyahısına bölünür, sözlər əlifba sırası ilə sıralanır və etiketdə göstərilir. Bütün bunları edən kod funksiyaya yerləşdirilməlidir:

```
def str_to_sort_list(event):
    s = et.get()
    s = s.split()
    s.sort()
    lb['text'] = ' '.join(s)
```

bind (bağlamaq) metodundan istifadə etməklə hadisə baş verdikdə çağırılan funksiyaların bir parametri olmalıdır, adətən buna ***event***, yəni “hadisə” deyilir. ***bind*** metodu vasitəsilə qrafik elementə funksionallığın əlavə edilməsinin ümumi şəkli:

```
Widget.bind(event,function)
```

Bizim funksiyamızda ***get*** metodundan istifadə etməklə mətn sahəsinə daxil olan mətn götürülür. **Split** (bölmə, ayırma) metodundan istifadə edərək sözlərdən ibarət olan mətn sözlər siyahısına çevrilir. Sonra siyahı sıralanır. Sonda etiketin ***text*** xassəsi dəyişdirilir. Ona, sətirin ***join*** (birləşmə) metodundan istifadə edərək siyahıdan əldə edilən sətir mənimsədilir.

İndi funksiya çağırışını hadisəyə bağlamalıyıq:

```
bt.bind('<Button-1>', str_to_sort_list)
```

Bu halda bu, ***bind*** metodundan istifadə etməklə həyata keçirilir. Hadisə və emal edici funksiyası ona ötürülür. Hadisə funksiyaya ötürüləcək və hadisə parametrinə təyin ediləcək. Burada hadisə ***'<Button-1>'*** sətri ilə işarələnən siçanın sol düyməsinin sıxılmasıdır. Əgər hadisə kimi siçanın sağ düyməsinin sıxılması və ya siçanın sol düyməsinin iki dəfə cəld sıxılması təyin olunur, onda o, uyğun olaraq ***'<Button-3>'*** sətri' və ya ***' <Double-Button-1>'*** sətri ilə işarələnir.

İstənilən proqramda vidjetlər təsadüfi olaraq pəncərənin ətrafına səpələnmir, onlar mütəşəkkil olurlar. İnterfeys ən xırda detallara qədər düşünülür və adətən müəyyən standartlara tabedir.

Pəncərənin səthində vidjetləri yerləşdirmək üçün ***location managers*** – ***yer menecerləri*** (vidjetlər üçün xüsusi yer göstəriciləri) istifadə olunur. Tkinter-də bunların üç növü var, hamısı vidcetləri fərqli şəkildə yerləşdirir: məsələn, ***grid*** (ing. şəbəkə, tor) adlanan menecer görünməz bir şəbəkə (sətirlər və sütunlardan ibarət) yaradır; ***place*** (ing. yer, yerləşdirmək) adlanan digəri ***x*** oxu və ***y*** oxu boyunca koordinatlardan istifadə edir; üçüncüsü – ***pack*** (ing. yığmaq, yerləşdirmək) adlanır və sadəcə olaraq, vidcetləri pəncərə çərçivəsinin yuxarı/aşağı/sol/sağ tərəflərinə "bağlayır".

Hələlik gəlin ən sadə tkinter yer menecerindən - ***pack*** metodundan istifadə edərək elementləri bir-birinin altında yerləşdirək:

```
et.pack()
bt.pack()
lb.pack()
```

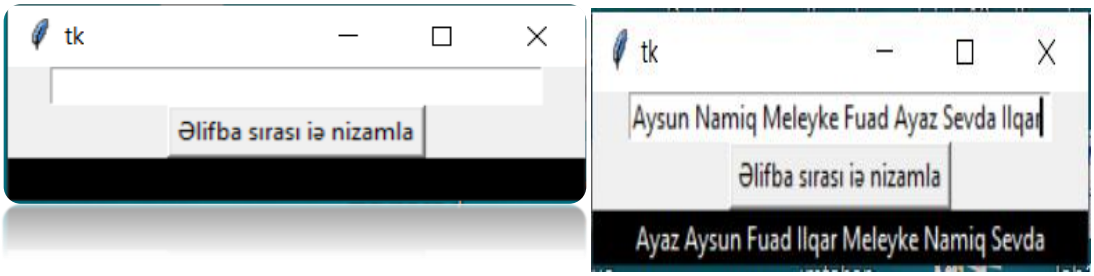
Tk sinfinin ***mainloop*** metodu əsas hadisələr dövrəsini başladır, bununla yanaşı, əsas pəncərənin onda yerləşdirilmiş bütün vidjetlər ilə göstərilməsinə səbəb olur:

```
root.mainloop()
```

Proqramın tam kodu belə olacaq:

```
from tkinter import *
def str_to_sort_list(event):
    s = et.get()
    s = s.split()
    s.sort()
    lb['text'] = ' '.join(s)
root = Tk()
et = Entry(width=40)
bt = Button(text="Əlifba sırası iə nizamla")
lb = Label(width=40, bg='black', fg='white')
bt.bind('<Button-1>', str_to_sort_list)
et.pack()
bt.pack()
lb.pack()
```

Bu skriptin icrası nəticəsində pəncərə görünür, bu pəncərənin mətn sahəsinə sözlərin siyahısını daxil edib düyməni sıxsanız sözlərin nizamlanmış versiyasını əldə edə biləcəksiiniz:



İndi isə proqramımızda obyekt yönümlü yanaşma tətbiq etməyə çalışaq. Belə yanaşma mütləq deyil, lakin çox vaxt rahat olur. Tutaq ki, etiket, düymə və mətn sahəsi qrupu Qrup sinfindən əldə edilən bir obyektdir (Qrup sinfinin nüsxəsidir). Onda proqramın əsas bölməsində əsas pəncərə, Qrup tipli obyekt və pəncərənin işə salınması əmri olacaq. Qrup əsas pəncərəyə bağlanmalı olduğundan, əcdad - pəncərəni sinif konstruktoruna ötürmək yaxşı olardı:

```
from tkinter import *
root = Tk()
qrup_1 = Qrup(root)
root.mainloop()
```

İndi Qrup sinfinin özünü yazaq:

```
class Qrup:
def __init__(self, master):
self.et = Entry(master, width=40)
        self.bt = Button(master, text="çevirmək")
        self.lb = Label(master, width=40, bg='black', fg='white')
        self.bt['command'] = self.str_to_sort
        self.et.pack()
        self.bt.pack()
        self.lb.pack()
    def str_to_sort(self):
        s = self.et.get()
        s = s.split()
        s.sort()
        self.lb['text'] = ' '.join(s)
```

Burada vidjetlər Qrup tipli obyektin sahələrinin qiymətləridir, düyməni basma hadisəsi idarəedici funksiyası ***bind*** metodundan deyil, düymənin ***command*** düyməsinin xassəsindən istifadə etməklə qurulur. Bu halda çağırılan funksiya hadisə parametrini tələb etmir. Biz metoda yalnız obyektin özünü ötürürük.

Lakin, belə məsələdə sinfin yaradılmasına ehtiyac yoxdur. Sinfin yaradılması o zaman əlverişli olacaq ki, bir neçə və ya çoxlu oxşar qrup obyektlərin olması tələb edilir.

Tutaq ki, bizə etiket, düymə, mətn sahəsindən ibarət bir neçə qrup lazımdır. Üstəlik, hər qrupun düyməsi öz click (düymənin sixilması) idarəedici funksiyasına malik olacaq.

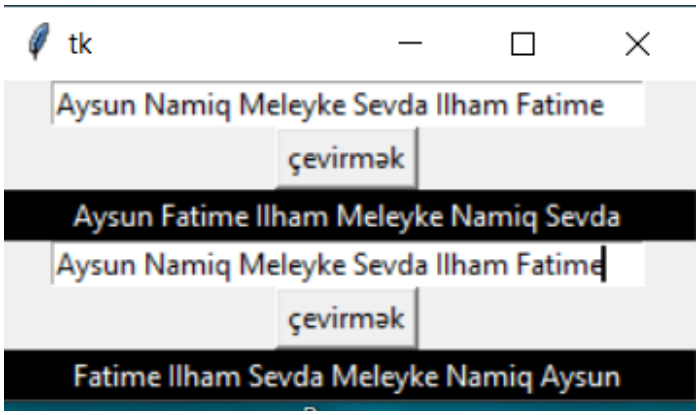
Onda ***command*** xassəsinin qiymətlərini (parametrləri) konstruktora ötürmək olar. Parametr kimi düyməyə bağlanılan hadisə idarəedici funksiyası ötürüləcək. Proqramın tam kodu:

```
from tkinter import *
class Qrup:
    def __init__(self, master, func):
        self.et = Entry(master, width=40)
        self.bt = Button(master, text="çevirmək")
        self.lb = Label(master, width=40, bg='black', fg='white')
        self.bt['command'] = getattr(self, func)
        self.et.pack()
        self.bt.pack()
        self.lb.pack()
    def str_to_sort(self):
        s = self.et.get()
        s = s.split()
        s.sort()    # siyahının elementlərini nizamlanmış qaydada yerləşdirir
```

```
            self.lb['text'] = ' '.join(s)
        def str_reverse(self):  .
            s = self.et.get()
            s = s.split()
            s.reverse()     # siyahının elementlərini tərs qaydada yerləşdirir
            self.lb['text'] = ' '.join(s)
    root = Tk()
    qrup_1= Qrup (root, 'str_to_sort')
    qrup_2= Qrup (root, 'str_reverse')
    root.mainloop()
```

'*str_to_sort*' və ya '*str_reverse*' sətirinin *func* üçün əvəz edildiyi ***getattr(self, func)*** ifadəsi ***self.str_to_sort*** və ya ***self.str_reverse*** ifadəsinə çevrilir.

Bu kod icra edildikdə, pəncərədə düymələri müxtəlif əməliyyatlari yerinə yetirən eyni tipli iki qrup görünəcəkdir:



Vidjet xassələrini sərt kodlaşdırmaqla deyil, qiymətləri konstruktora arqumentlər kimi ötürməklə və sonra obyektlər yaradarkən onları uyğun variantlara təyin etməklə sinif daha çevik edilə bilər.

Ədəbiyyat

1. Д. Ю. Федоров «Основы программирования на примере языка PYTHON». Учебное пособие, Санкт-Петербург, 2019
2. Сузи Р. А. « Язык программирования Python»: Учебное пособие. — М.: Интуит, Бином. Лаборатория знаний, 2006. — 328 с.
3. https://ru.wikibooks.org/wiki/GUI_Help/Tkinter_book
4. https://www.russianlutheran.org/python/life/life.htm Матт Конвей (Matt Conway) Спасательный круг для изучающих Tkinter.
5. https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter
6. https://metanit.com/python/tkinter/
7. http://it.kgsu.ru/Python_Tk/oglav.html

TKINTER. ПРОГРАММИРОВАНИЕ GUI НА PYTHON

Бабаева Шакяр Маарифовна

Резюме

В настоящее время почти все приложения, которые создаются для конечного пользователя, имеют GUI. Существует множество библиотек GUI для Python. Установочный файл интерпретатора Python обычно уже включает пакет tkinter в составе стандартной библиотеки. Настоящая статья посвящена созданию программ с графическим интерфейсом с использованием библиотеки tkinter.

Программы с графическим интерфейсом пользователя событийно-ориентированные. Событийно-ориентированное ориентировано на события. То есть та или иная часть программного кода начинает выполняться лишь тогда, когда случается то или иное событие. В статье были изложены этапы создания программы с графическим интерфейсом, а также разработана программа с использованием виджетов под соответствующую задачу. В завершении, на примере одной задачи поясняется, что объектно-ориентированный подход к проектированию программ удобно применять для задач, требующих наличия нескольких или многих подобных групп объектов.

TKINTER. PYTHON GUI PROGRAMMING

Babayeva Shakar Maarif

Summary

Nowadays, almost all applications that are created for the user have a GUI. There are many GUI libraries for Python. The Python interpreter installation file usually already includes the tkinter package as part of the standard library. This article is about creating programs with a graphical interface using the tkinter library.

Programs with a graphical user interface are event-driven. Event-driven programming is event-driven. That is, this or that part of the program code begins to be executed only when this or that event occurs. The article outlined the steps for creating a program with a graphical interface, and also developed a program using widgets for the corresponding task. In conclusion, using the example of one task, it is explained that the object-oriented approach to program design is convenient to apply to tasks that require the presence of several or many such groups of objects.

Rəyçi: Aslanov Ə.Ə.
Cəbr və İnformatika kafedrasının 31 yanvar 2023-cü il tarixli iclasın 01 saylı protokolu
Daxil oma tarixi 02 fevral 2023-cü il

UOT:53.

PYTHON PROQRAMLAŞDIRMA DİLİNDƏ GEOTƏTBİQLƏRİN EMAL EDİLMƏSİ

Qənbərova Sahibə İbrahim qızı, Əsgərova Tamilla Əhməd qızı
Gəncə Dövlət Universiteti
genberova.s63@gmail.com, asgerovatamilla998@mail.ru

Xülasə*:**Müasir dövrdə Python proqramlaşdırma dili ən geniş istifadə olunan yüksək proqramlaşdırma dillərindən biri hesab olunur. Veb sistemlərin yaradılması, stolüstü oyunlar, robototexnika, elmi, riyazi hesablamalarla yanaşı geoinformatika sahəsində də istifadə olunur. Geoverilənlərlə işin normal şəkildə təşkili və vizuallaşdırılmaı üçün geoinformasiya sistemləri yaradılmışdır. Bu sistemlər özündə geoverilənlərin saxlanması, idarə olunması, onların vizualllaşdırılması üçün lazım olan mürəkkəb hesablama sistemlərini özündə birləşdirir.*