

# PYTHON FUNDAMENTALS

Curs interactiv de python

# STRUCTURA SEDINTA 7 : KIVY

▤ Tema sedinta anterioara

▤ Kivy

# TEMA

- Modificati programul Tema\_clasa1.py primit pe e-mail pentru a rula in consola o comanda de afisare a datei curente. Extrageți cu ajutorul RE anul curent.



Exemple Python

```
C:\Users\Fam.Popescu>date /t  
Fri 01/16/2015
```

# TEMA

```
# Program Tema_RE.py|  
# Explica functiile RE si os.popen  
# Ion Studentul - 1/26/13  
  
import os, re  
print "aplicam comanda date in consola"
```

# TEMA

- Programul 1: Realizati un program care sa extraga timpul de la serverul ntp <<2.ro.pool.ntp.org>>. Transformati timpul intr-un format prietenos cu ajutorul time.ctime(). Transmiteti prin e-mail acest timp catre adresa personala gmail/yahoo utilizand o adresa de e-mail gmail/yahoo. Utilizati transmiterea de mesaj ca sir de caractere.
- Dupa testare si reusita inlocuiti parola utilizata la adresa de email cu sirul de caractere <<\*\*\*\*>>.

# TEMA SEDINTA ANTERIOARA

```
NTP client_tema ✕  
1 # Program Tema 1  
2 # Aplica functiile ntplib si e-mail  
3 # Ion Studentul -- 1/26/13  
4  
5 #importam modulele utilizate  
6 import ntplib  
7 import time  
8 import ctypes
```



# TEMA

- Creati un program care sa se conecteze la telnet la adresa <<route-server.ip.att.net>> cu un debug level setat la zero(standard este zero) printr-o variabila definite la inceputul programului. Daca aceasta variabila e diferita de 0 atunci afisati si fiecare linie ce contine metoda
- read\_until() aplicata obiectului telnetlib. Astfel creati un debug level
- Conexiunea are nevoie de credentiale prin urmare ar trebui sa utilizeze
- userul: rviews
- parola: rviews
- Aplicati apoi comanda <<show route table inet.0>>
- Aceasta comanda returneaza tabelul de rutare. Extrageți doar pana la intalnirea expresiei <<---(more)--->>
- Cautati in textul returnat de comanda <<show route table inet.0>>
- sirul de caractere <<0.0.0.0>> cu ajutorul re.search() si afisati cu ajutorul metodei group(0).

# TEMA SEDINTA ANTERIOARA

telnet\_tema

```
1 # Program Telnet Client
2 # Explica functiile telnetlib
3 # Ion Studentul - 1/26/13
4
5 import telnetlib, re
6
7 HOST = "route-server.ip.att.net"
8 user = "rviews"
```



Kivy este un package ce ofera crearea de interfete grafice. Principalele avantaje sunt:

- Suport pentru lunga durata (proiect finantat cu angajati permanent)
- Suport pentru multi-touch
- Cross-Platform: Linux, Windows, OS X, Android
- concentrat în sensul în care putem scrie aplicatii în cateva linii de cod.
- Kivy este gratuit. Nu trebuie să platesti pentru kivy, chiar dacă tu faci bani de pe urma vanzarilor de aplicatii kivy.

Cat de rapid este kivy stiind ca este un pic mai lent decat C++?

- poti observa in diferite benchmark ca Python este mai lent decat C++.
- Utilizeaza Cython pentru compliare la nivelul limbajului C unde este critic cum ar fi dispecerizarea de evenimente și desenarea de grafice
- Desenarea de grafice Kivy se bazeaza pe GPU pt a maximiza performanta

Lista de dependente:

- Cython.

Alte module utilizate sunt(optional):

- OpenCV 2.0 – Camera de luat vederi.
- PIL – Afisarea de imagine și text.
- PyCairo – Afisare de text.
- PyEnchant – Corectura de text ortografic.
- PyGST – Redare audio/video și camera de luat vederi
- Pygame pt. redare audio/video

# KIVY: INSTALARE





Exista 3 tipuri de instalare pentru windows:

- Kivy ofera instalarea dintr-o arhiva - versiune portabila cu toate dependentele
- Instalare kivy manual
- Instalarea de kivy cu Python Extension Packages

# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Pasul 1: Copiaza ultima versiune de kivy accesand: <http://kivy.org/#download>

Operating System	File	Instructions	Size
 Windows 7, 8 (32/64 bit)	<a href="#">Kivy-1.8.0-py2.7-win32.zip (Mirror)</a> <a href="#">Kivy-1.8.0-py3.3-win32.zip (Mirror)</a>	<a href="#">Installation for Windows</a>	126MB 129MB
 Mac OS X 10.7, 10.8, 10.9 (requires Python 2.7)	<a href="#">Kivy-1.8.0-osx.dmg (Mirror)</a>	<a href="#">Installation for MacOSX</a>	34 Mb
 Linux (tested Ubuntu > 13.04, 32/64 bit, Maegia, Arch)	<a href="#">Kivy-1.8.0.tar.gz</a>	<a href="#">Installation for Ubuntu</a>	13 Mb
 Ubuntu PPA	<a href="#">Stable PPA</a> <a href="#">Daily PPA</a>	<a href="#">How to use software from PPA</a>	12 Mb

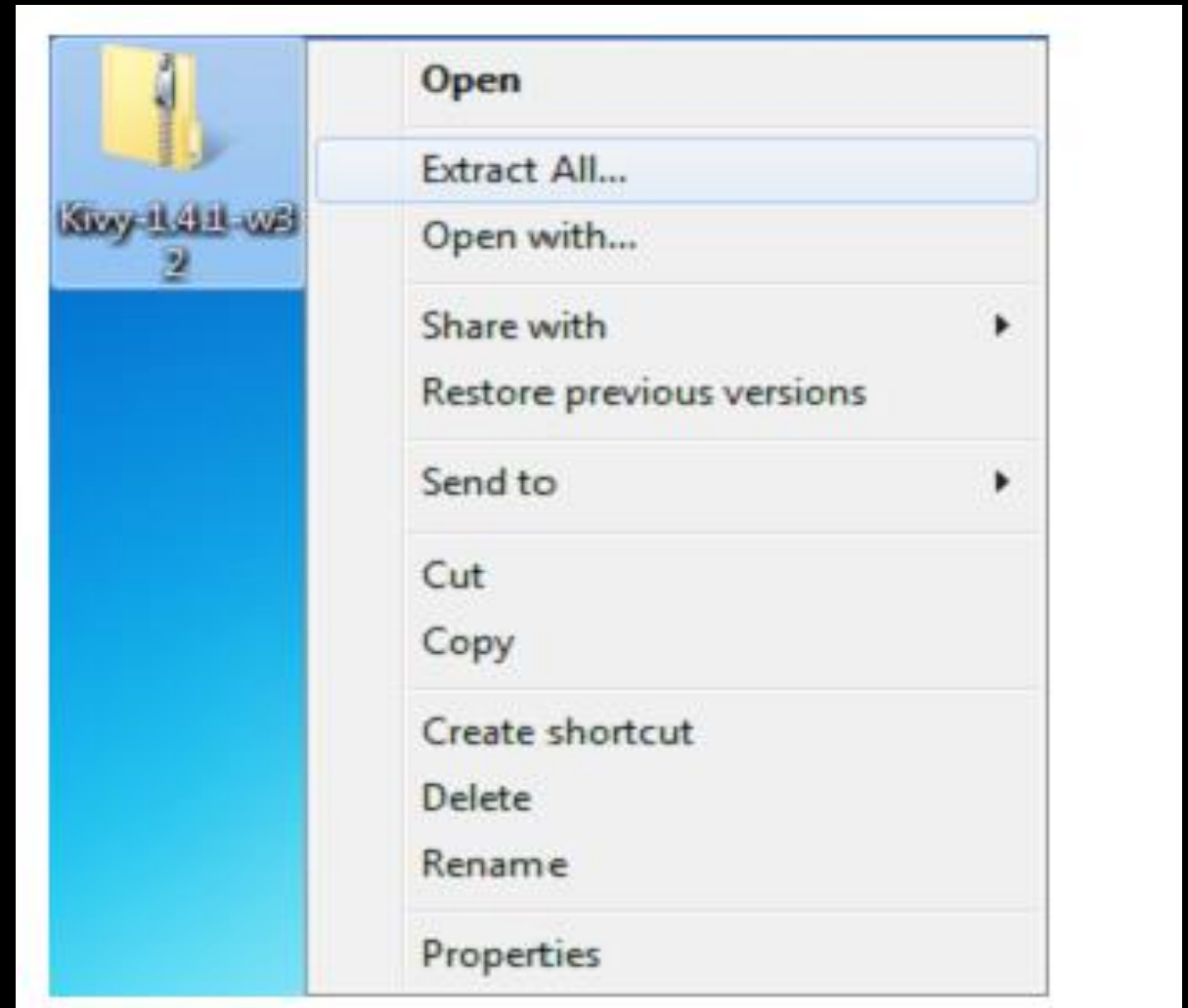
# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Pasul 2: Extrage fisierul copiat.

Va rog sa-l extrageti in calea  
C:\Python cu kivy\

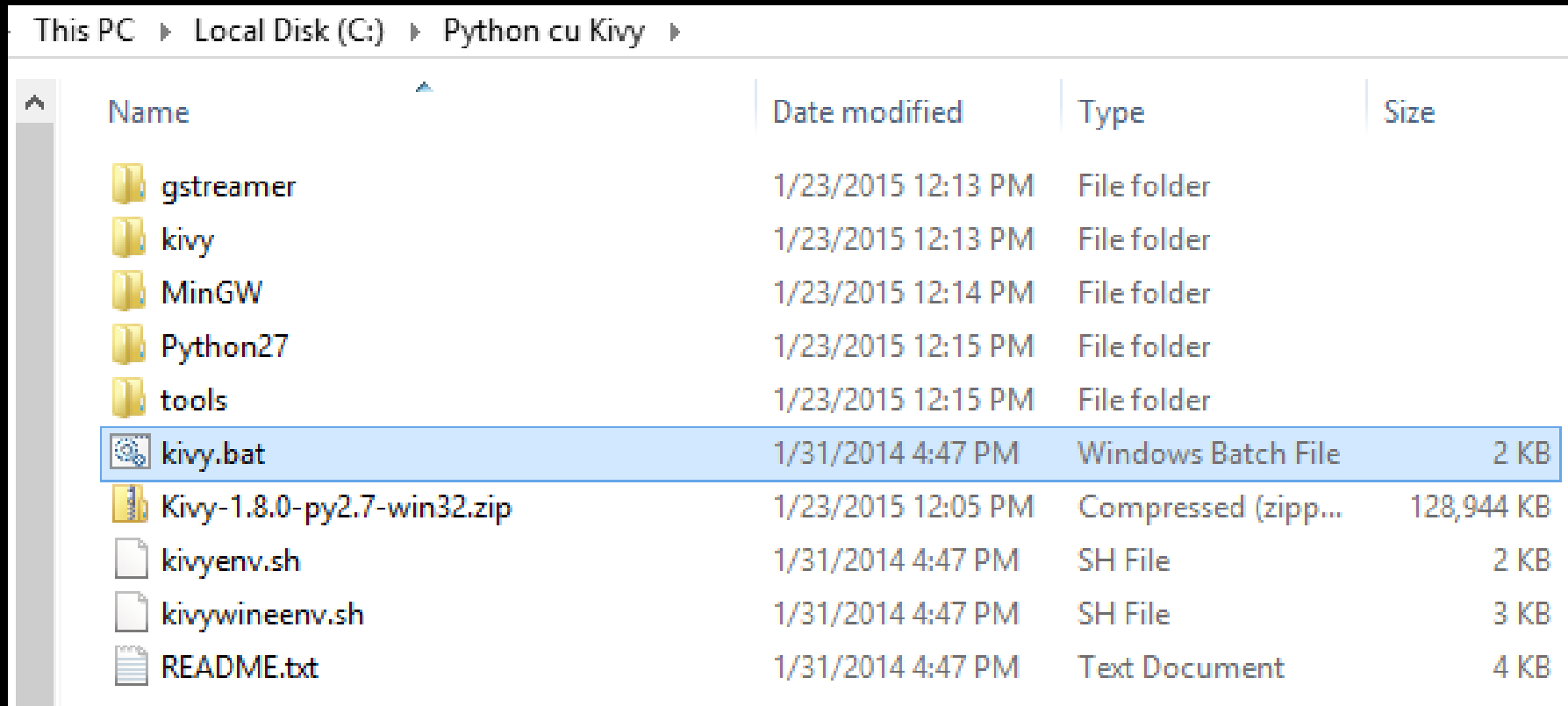
OBS. Aceasta arhiva vine cu Python  
preinstalat





# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

Pasul 3: In directorul unde ai dezarhivat fisierul zip copiat local la pasul 2 ai un fisier numit kivy.bat. Utilizeaza acest fisier pt. a lansa orice aplicatie kivy.

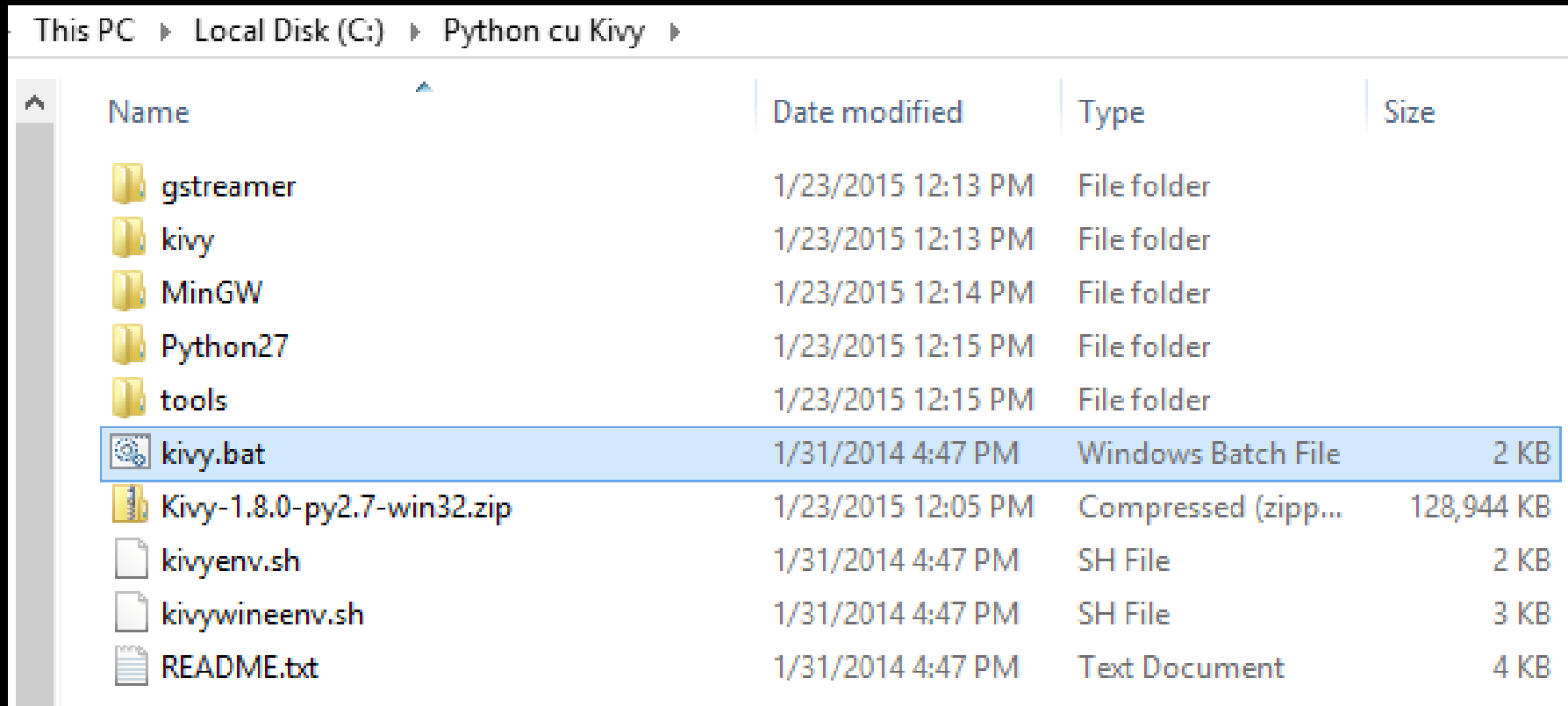


This PC > Local Disk (C:) > Python cu Kivy >				
Name	Date modified	Type	Size	
gstreamer	1/23/2015 12:13 PM	File folder		
kivy	1/23/2015 12:13 PM	File folder		
MinGW	1/23/2015 12:14 PM	File folder		
Python27	1/23/2015 12:15 PM	File folder		
tools	1/23/2015 12:15 PM	File folder		
kivy.bat	1/31/2014 4:47 PM	Windows Batch File	2 KB	
Kivy-1.8.0-py2.7-win32.zip	1/23/2015 12:05 PM	Compressed (zipp...	128,944 KB	
kivyenv.sh	1/31/2014 4:47 PM	SH File	2 KB	
kivywineenv.sh	1/31/2014 4:47 PM	SH File	3 KB	
README.txt	1/31/2014 4:47 PM	Text Document	4 KB	

# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Pentru a putea rula aplicatii kivy cu acest kivy.bat avem doua posibilitati. Una este să utilizam metoda send-to, iar cealalta este să utilizam metoda double-click..



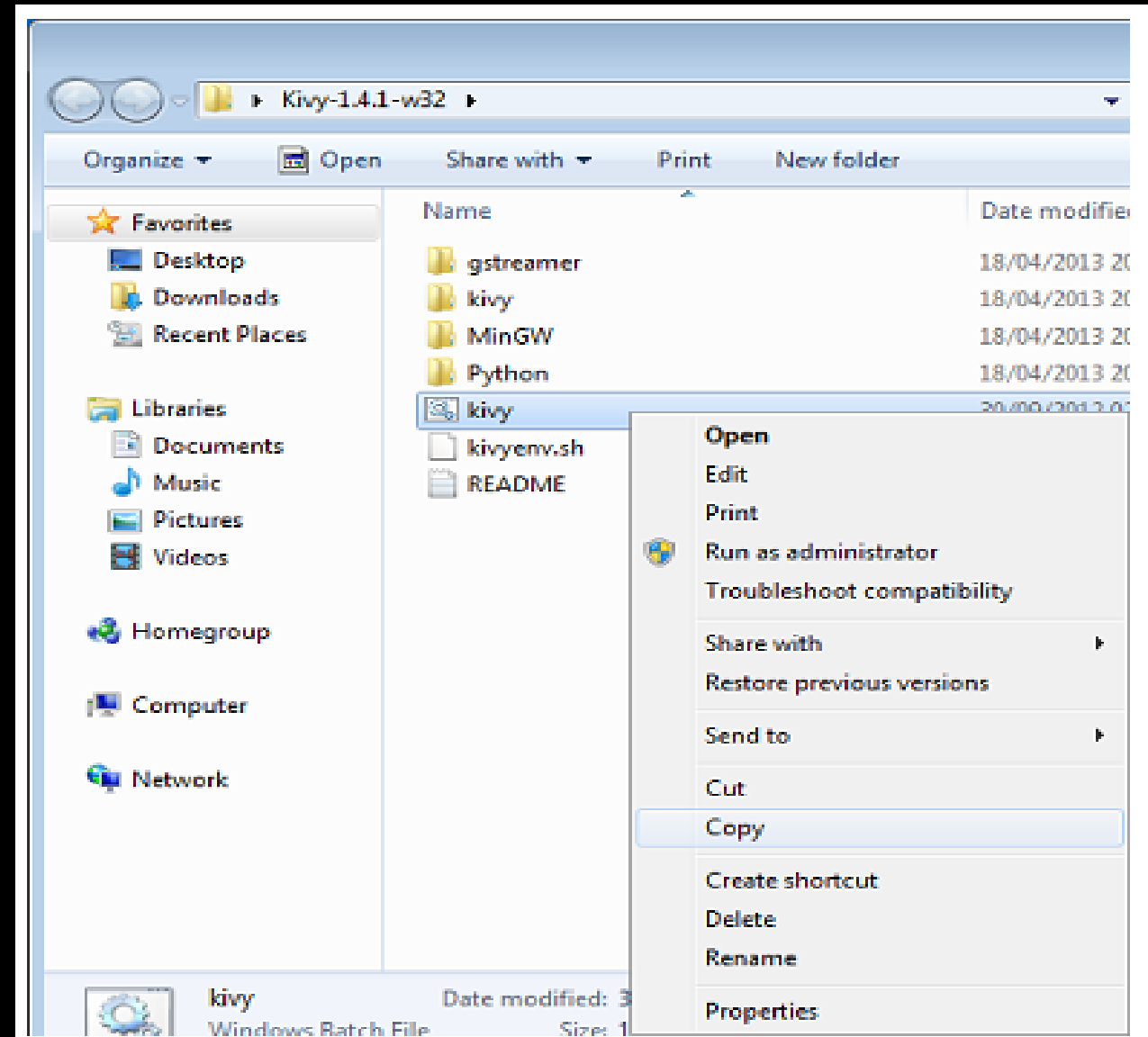
This PC > Local Disk (C:) > Python cu Kivy >				
Name	Date modified	Type	Size	
gstreamer	1/23/2015 12:13 PM	File folder		
kivy	1/23/2015 12:13 PM	File folder		
MinGW	1/23/2015 12:14 PM	File folder		
Python27	1/23/2015 12:15 PM	File folder		
tools	1/23/2015 12:15 PM	File folder		
kivy.bat	1/31/2014 4:47 PM	Windows Batch File	2 KB	
Kivy-1.8.0-py2.7-win32.zip	1/23/2015 12:05 PM	Compressed (zipp...	128,944 KB	
kivyenv.sh	1/31/2014 4:47 PM	SH File	2 KB	
kivywineenv.sh	1/31/2014 4:47 PM	SH File	3 KB	
README.txt	1/31/2014 4:47 PM	Text Document	4 KB	

# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Activarea kivi utilizand metoda send-to:

- Pasul 1: Copiaza fisierul kivy.bat în clipboard.

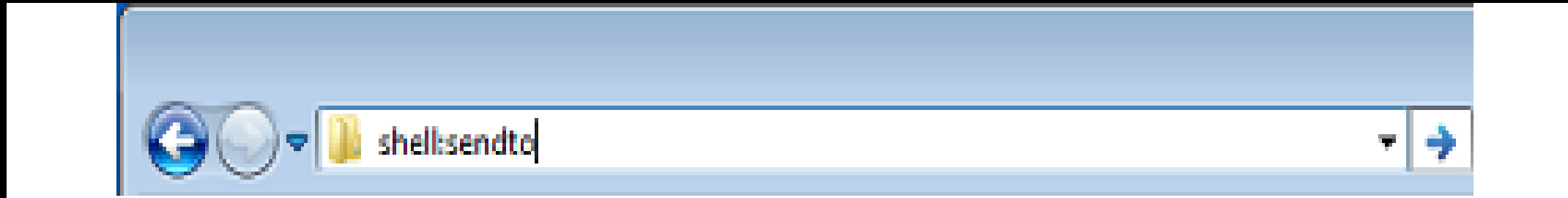


# NSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Activarea kivi utilizand metoda send-to:

- Pasul 2: Deschide Exploratorul de fisiere al windows-ului și scrie în address-bar <<shell:sendto>>

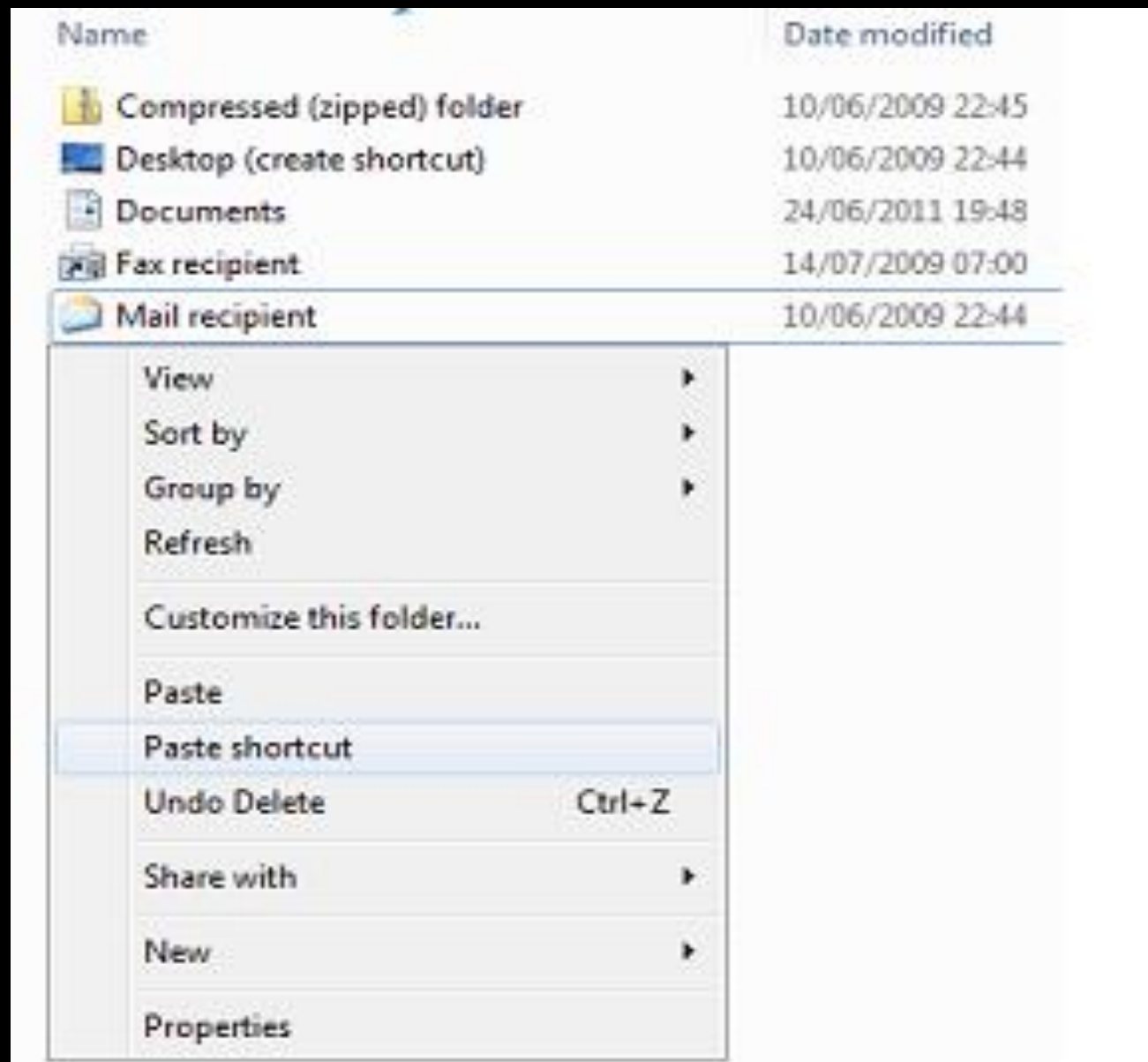


# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Activarea kivi utilizand metoda send-to:

- Pasul 3: Ar trebui să te trimita către un director special al windows-ului unde trebuie sa lipesti (paste) fisierul kivy.bat copiat anterior ca și shortcut(scurtatura):

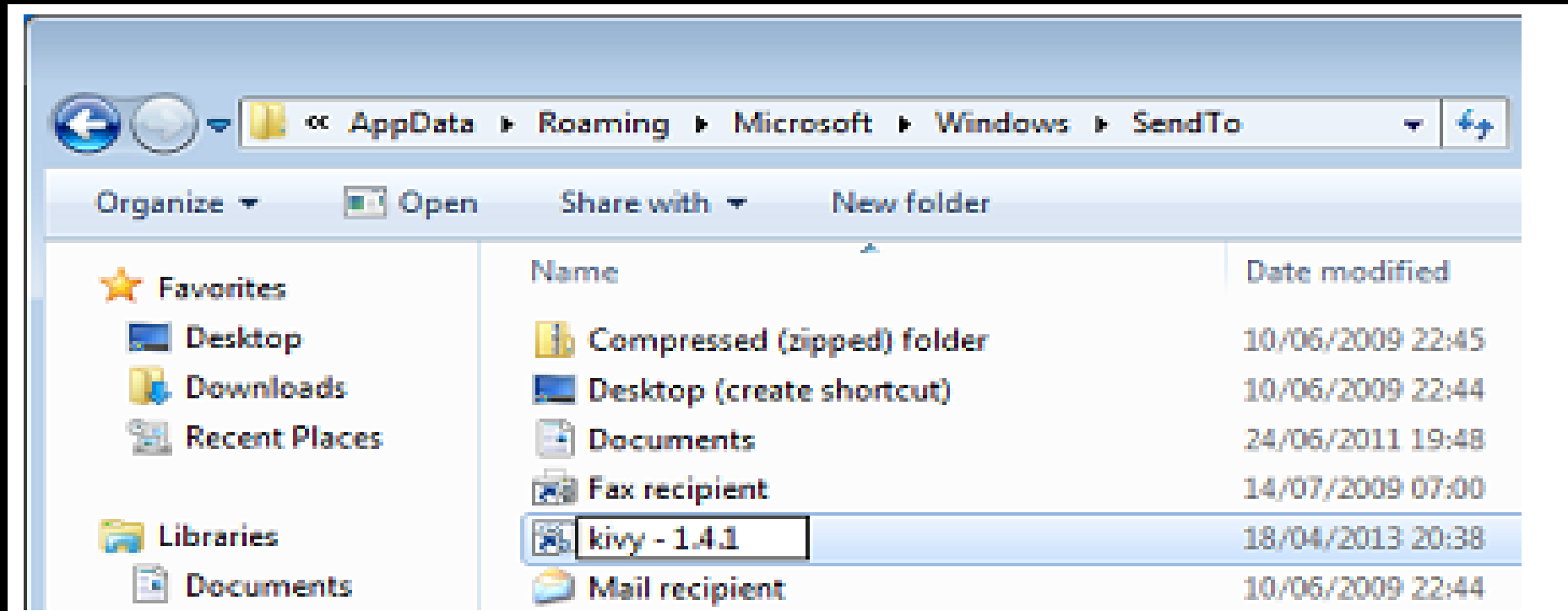


# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Activarea kivi utilizand metoda send-to:

- Pasul 4: Redenumeste shortcut-ul ca kivy versiune.

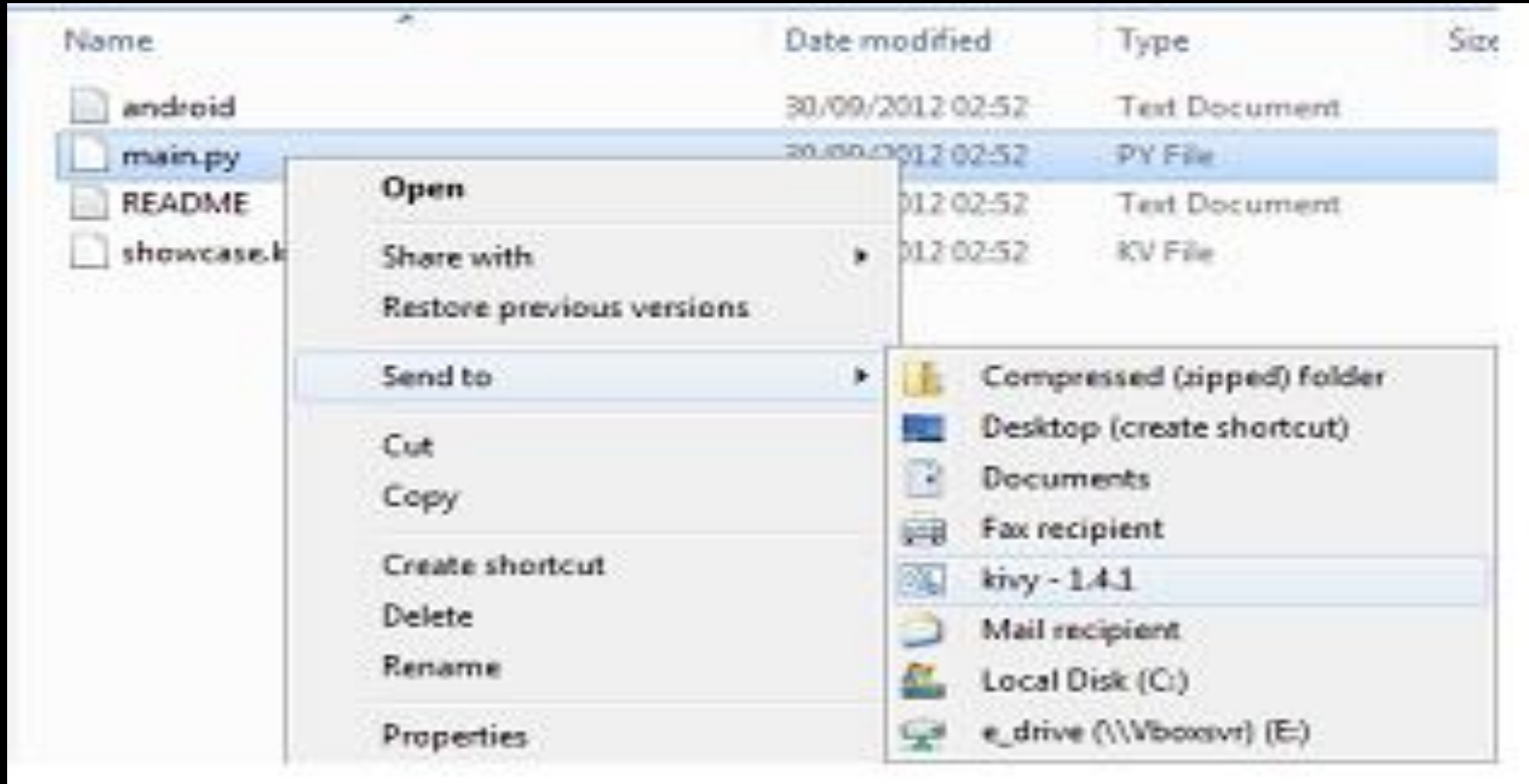




# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Acum poti rula kivy dand click pe aplicatia ta apoi apeland sendto și kivy.



# INSTALAREA DINTR-O ARHIVA - VERSIUNE PORTABILA

INFOACADEMY.NET

Activarea kivi utilizand metoda Click-DR:

- Da click dreapta pe un fisier .py
- Alege "Open with..." apoi "choose default program"
- Navigheaza catre kivy.bat din fisierul dezarhivat portabil și selecteaza-l.
- Apoi de fiecare data cand vei da dublu click pe tipul de fisier ".py" vei rula fisierul apeland kivy.bat

Pentru o compatibilitate cu modulele invatate in sedinta 6 e bine sa instalam modulele aditionale si in varianta kivy.

# INSTALAREA KIVY MANUAL

INFOACADEMY.NET

Aceasta metoda nu este recomandata deoarece de multe ori aceste dependente trebuiesc unite manual, altfel nu functioneaza:

- Glew 1.5.7: <http://sourceforge.net/projects/glew/files/glew/1.5.7/glew-1.5.7-win32.zip/download>
- Pygame 1.9.2: <http://pygame.org/ftp/pygame-1.9.2a0.win32-py2.7.msi>
- Cython 0.14 – necesita Visual Studio 2003/2008 pt. instalare  
<https://pypi.python.org/packages/source/C/Cython/Cython-0.14.tar.gz>
- MinGW: <http://sourceforge.net/projects/mingw/files/Installer/mingw-get-setup.exe/download>
- Gstreamer: <http://gstreamer.freedesktop.org/data/pkg/windows/1.4.1/gstreamer-1.0-x86-1.4.1.msi>
- Setuptools: [https://bootstrap.pypa.io/ez\\_setup.py](https://bootstrap.pypa.io/ez_setup.py)

# INSTALAREA CU PYTHON EXTENSION PACKAGES

INFOACADEMY.NET

O alta modalitate este instalarea de kivy cu Python Extension Packages realizata de Christoph Gohlke. Acestea sunt executabile de windows care instaleaza peste versiunea python curenta toate fisierele necesare pentru rulara modulelor dorite.

Aceste executabile sunt neoficiale si au scop de testare si evaluare.

- Pygame (varianta oficiala):

<http://pygame.org/ftp/pygame-1.9.2a0.win32-py2.7.msi>

- Aceste pachete Python vin cu extensia (whl). Prin urmare aveti nevoie de wheel pentru instalare:

<https://pypi.python.org/pypi/wheel#downloads>

- Kivy Python Extension Package - Christoph Gohlke extensia whl:

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#kivy>

# INSTALAREA CU PYTHON EXTENSION PACKAGES

INFOACADEMY.NET

Pentru a simplifica procedura in locul extensiei whl si a package-ului copiat de pe pagina lui Christoph Gohlke voi oferi pentru acest curs un executabil cu extensia \*.exe.

- Pygame (varianta oficiala):

<http://pygame.org/ftp/pygame-1.9.2a0.win32-py2.7.msi>

- Kivy Python Extension Package - Christoph Gohlke extensia exe:

Oferita prin prisma cursului Python.

Crearea unei aplicatii se realizeaza destul de simplu:

- In cadrul clasei trebuie să mostenim App()
- Trebuie să implementam metoda build() pentru a returna instanta widget (radacina tree-ului widget). Un root widget este acel obiect de tip graphic peste care se construiește toata aplicatia. Un widget este un element grafic cum ar fi un buton, o fereastră etc.

Build initializeaza aplicatia și poate fi aplelat doar o data. Dacă metoda returneaza un widget acesta va deveni root widget și toate celelate widget-uri trebuie să devina copii la root widget.

- Initializarea acestei clase și apelarea metodei run pentru rulare



Iata un exemplu simplu de aplicatie ce nu are nici un widget aplicat, deci nu are root:

```
# Program kivy0
```

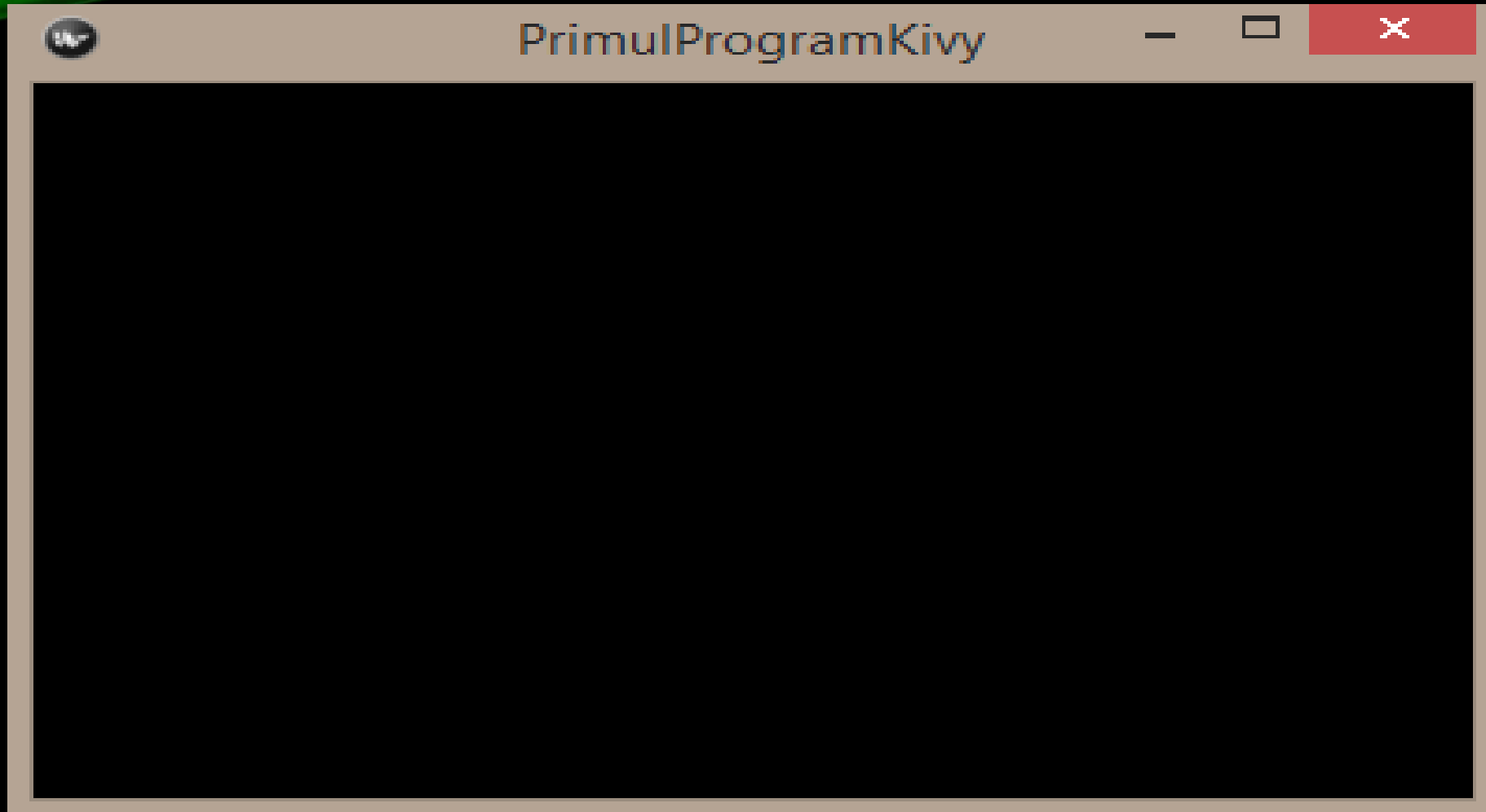
```
# Explica functiile kivy
```

```
# Ion Studentul - 1/26/13
```

```
from kivy.app import App
```

```
class PrimulProgramKivy(App):  
    pass
```

```
PrimulProgramKivy().run()
```



Puteti să salvati acest fisier sub un fisier .py și sa-l rulati. În urmatoarea sectiune vom explica cum functioneaza cel mai simplu program.

```
from kivy.app import App
```

- Este necesara operatia de mosterire, adica clasa de baza să mosteneasca clasa App. Acest modul app se regaseste in:

kivy\_installation\_dir/kivy/app.py.

```
class PrimulProgramKivy(App):
```

Acasta contine o modalitate de a defini clasa de baza pt. aplicatia kivy, singurul lucru care poate fi schimbat aici este numele aplicatiei din **PrimulProgramKivy** în cum doriti să se numeasca.

Pt ca aplicatia să ruleze efectiv avem urmatoarele linii:

```
    PrimulProgramKivy().run()
```

Aceste linii au rolul de a initializa clasa PrimulProgramKivy și de a rula metoda run()

# CREAREA UNEI APLICATII

INFOACADEMY.NET

Iata un alt exemplu simplu de aplicatie:

```
from kivy.app import App
from kivy.uix.label import Label

class MyApp(App):

    def build(self):
        return Label(text='Hello world')

if __name__ == '__main__':
    MyApp().run()
```

# CREAREA UNEI APLICATII

INFOACADEMY.NET



```
from kivy.app import App
```

Este necesar ca clasa de baza să mosteneasca clasa app. Aceasta se regaseste in kivy\_installation\_dir/kivy/app.py.

```
from kivy.uix.label import Label
```

La aceasta linie dorim să importam din kivy o anumita functie numita Label().

De retinut ca modulul uix mentine elementele de user interface precum butoane si alte widget-uri.

```
class MyApp(App):
```

Aceasta contine o modalitate de a define clasa de baza pt. aplicatia kivy, singurul lucru care poate fi schimbat aici este numele aplicatiei din MyApp în cum doriti să se numeasca.



```
def build(self):
```

Aceasta functie are scopul să initializeze și returneze Root Widget.

```
return Label(text='Hello world')
```

Returnarea Root Widget. Acest root Widget este initializat cu o eticheta (label) cu textul "Hello World" și returnează instanța

Pt ca aplicatia să ruleze efectiv avem următoarele linii:

```
if __name__ == '__main__':
```

```
    MyApp().run()
```

Aceste linii au rolul de a initializa clasa MyApp și de a rula metoda run()

Tot ceea ce vedeți pe ecran este oarecum desenat de un widget.

- Widget-ul are trei proprietati generale: un widget încapsulează date, definește interacțiunea utilizatorului cu acele date și desenează reprezentarea sa vizuală.
- Puteți construi apoi orice, de la o simpla interfata la interfețe complexe pt. utilizator prin nesting widget.
- widget-urile în kivy sunt organizate sub formă aborescenta. Aplicatia noastra are un root widget, care va avea copii. Fiecare copil poate avea copii la randul sau.
- Tree-ul widget poate fi manipulat cu urmatoarele metode:
  - `add_widget()`: adauga un widget ca un copil
  - `remove_widget()`: elimina un copil de pe lista de copii
  - `clear_widgets()`: elimina toti copii existenti de la un widget

Acesta se ocupa de modul cum sunt aranjate celelalte widget-uri în mod automat, deci toate celelalte widget-uri vor fi copii la un tip de layout. Am spus la un tip de layout deoarece nu putem utiliza modulul ca atare ci doar să mostenim un tip de layout și sa-l utilizam.

Layouts utilizeaza `size_hint` și `pos_hint` pt. a determina dimensiunea și pozitia copiilor.

- Module: [kivy.uix.layout](http://kivy.org/doc/stable/api-kivy.uix.layout.html)

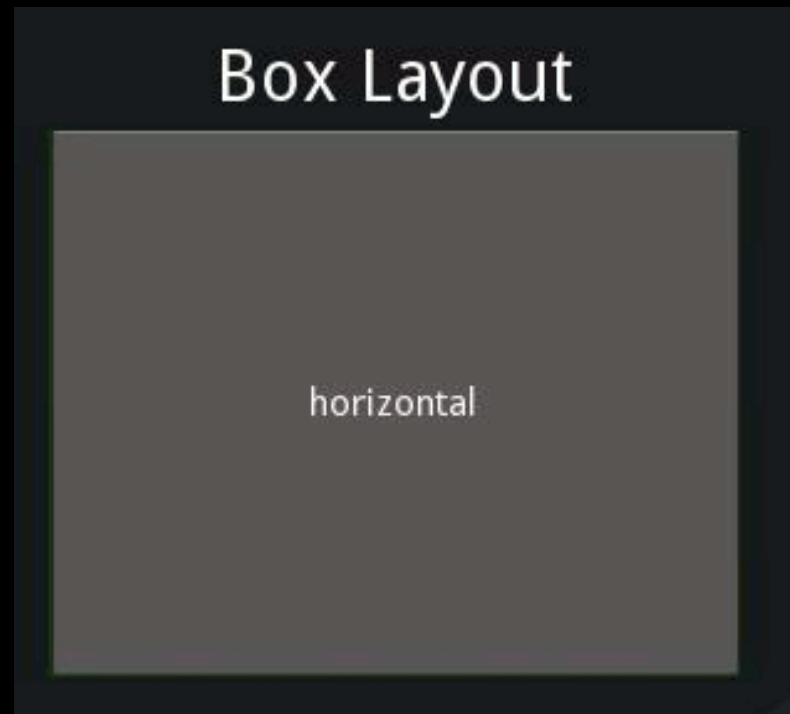
Anchor layout :

- Un layout simplu ce utilizeaza doar pozitiile copiilor. Permite să asezam copilul la o pozitie relativa fata de margimea layout-ului. Ancora `anchor_x` poate lua valorile `left`, `center` sau `right`. Ancora `anchor_y` poate lua valorile `top`, `center` sau `bottom`. Proprietatea `size_hint` a copilului nu va fi onorata.



- Box layout

Aranjeaza widget-urile intr-o maniera adiacenta, fie vertical, fie orizontal. Scopul este să umplem tot spatial layout-ului. Proprietatea `size_hint` a copilului poate fi utilizată pt. a schimba proportiile fiecarui copil sau să setam valori fixe pentru unele din ele.



## Float layout

Permite plasarea copiilor în locatii arbitrare și marimi arbitrare facand referire la valoarea absoluta sau relative a layout-ului. Marimea standard a layout-ului este `size_hint (1, 1)`. Aceasta va face fiecare copil de asemenea dimensiune cu layout-ul deci e recomandat să o schimbam dacă dorim să avem mai mult de un copil. Putem seta `size_hint` la `(None, None)` pentru a utiliza marimea absoluta setata cu `size`. Acest widget ia în calcul și `pos_hint`, parametru care dicteaza pozitia rel ative fata de layout.





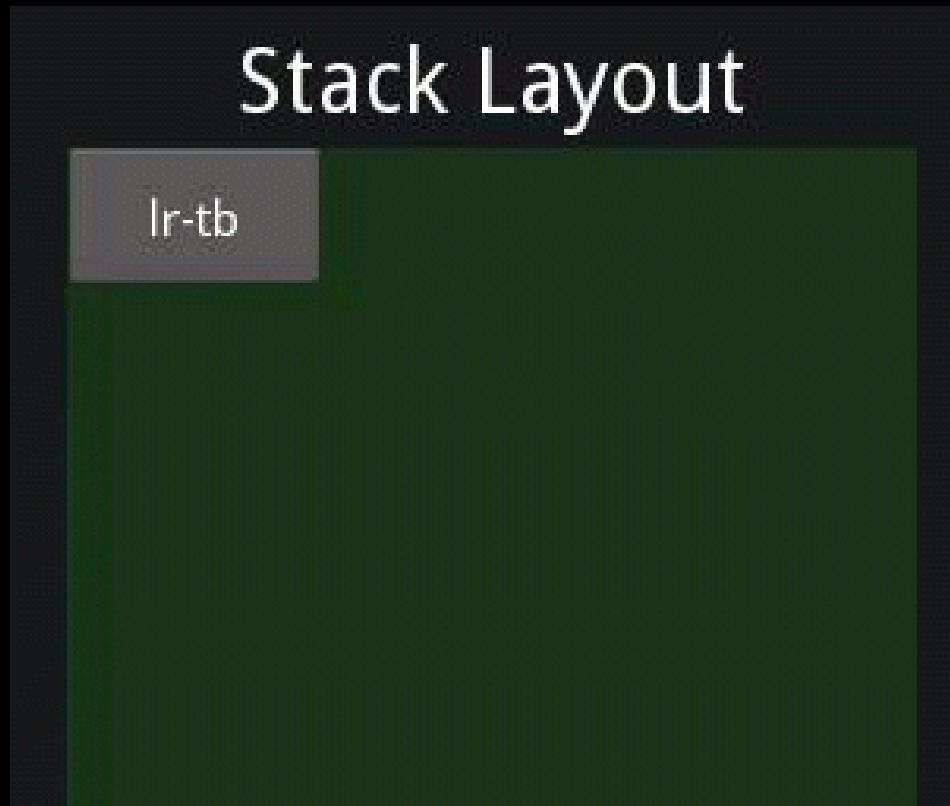
- Grid layout :

Aranjeaza widget-urile intr-o grila. Trebuie să specifici cel puțin o dimensiune a grilei pt. a putea să calculeze marimea elementelor și cum le aranjeaza.



- Stack layout :

Aranjeaza widget-urile adiacent unul fata de altul, dar cu o marime setata în una din dimensiuni fara a incerca să umple tot spatiul.



Fiecare layout are si o imagine gif animata care explica mai bine aceste caracteristici.

## Grid Layout

rows = 3

# EXEMPLU BOXLAYOUT

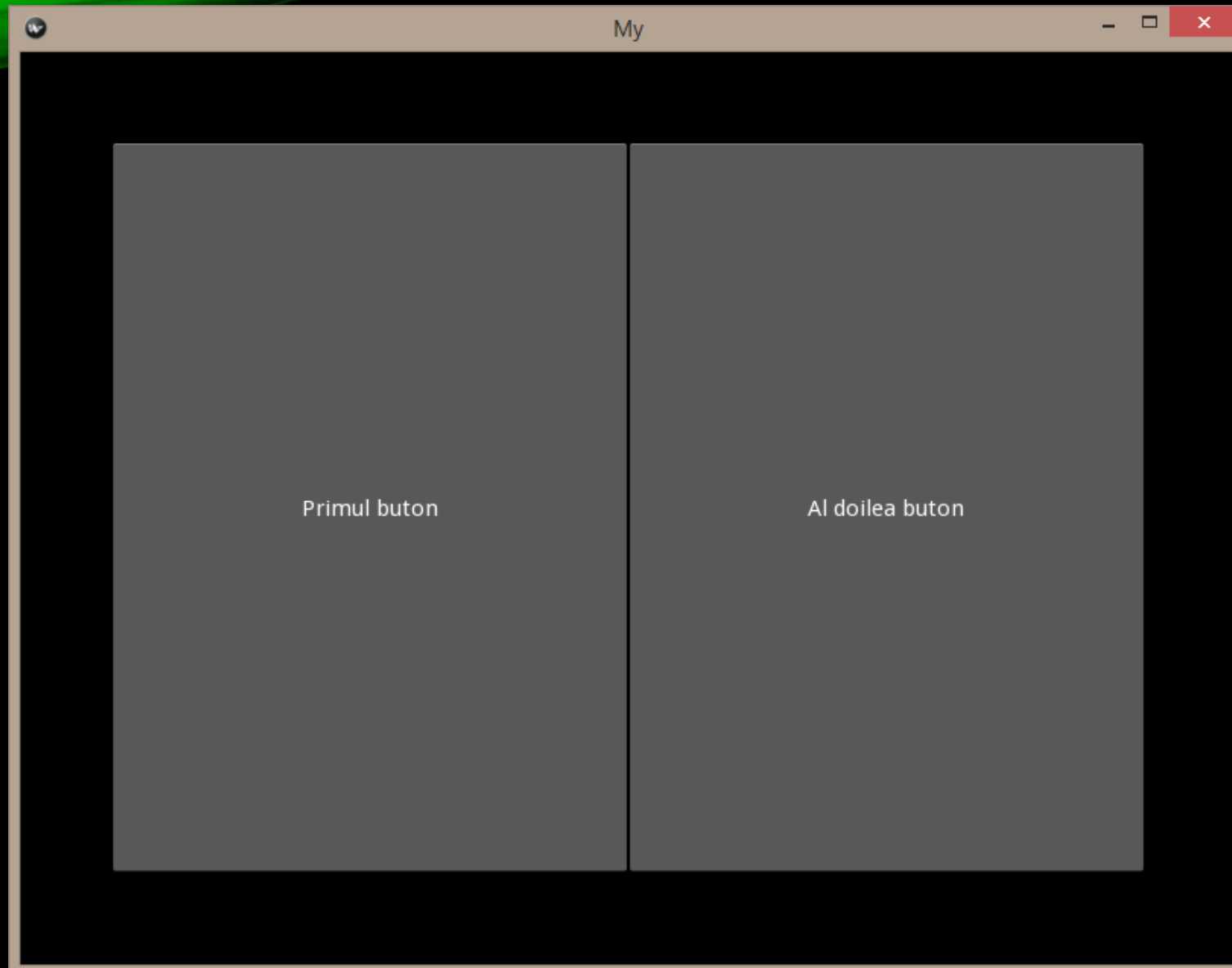
```
class LoginScreen(BoxLayout):  
    def __init__(self, **kwargs):  
        super(LoginScreen, self).__init__(**kwargs)  
        self.orientation='horizontal'  
        self.padding=60  
        self.buton1 = Button(text='Primul buton')  
        self.add_widget(self.buton1)  
        self.buton2 = Button(text='Al doilea buton')  
        self.add_widget(self.buton2)
```

```
class MyApp(App):  
    def build(self):  
        x=LoginScreen()  
        return x
```

```
if __name__ == '__main__':  
    MyApp().run()
```

# EXEMPLU BOXLAYOUT

INFOACADEMY.NET



# EXEMPLU BOXLAYOUT

La următoarea linie importam widget-ul de tip Button:

```
from kivy.uix.button import Button
```

La următoarea linie importam widget-ul de tip BoxLayout:

```
from kivy.uix.gridlayout import BoxLayout
```

Aceasta clasa este folosita ca o bază pentru Root Widget (LoginScreen):

```
class LoginScreen(BoxLayout):
```



Urmatoarele doua linii au rolul de a mosteni toate attributele pe care le are boxlayout-ul. Este o scriere standard in Kivy. Clasa LoginScreen, va suprascrie metoda `__init__()` pentru a adăuga widgeturi și pentru a defini comportamentul lor.

```
def __init__(self, **kwargs):  
    super(LoginScreen, self).__init__(**kwargs)
```

Nu trebuie să uităm să apelăm `super` în scopul de a implementa funcționalitatea clasei inițiale suprascrise. De asemenea, rețineți că aceasta este o bună practică să nu omită `kwargs **`. Acest `kwargs` are rolul de a accepta argumente alterioare standard, deci fara aceasta parte nu va functiona.

```
self.orientation='horizontal'
```

Aceasta linie de mai sus are rolul de a seta orientarea Boxlayout-ului. Default este horizontal.

```
self.padding=60
```

Aceasta linie de mai sus are rolul de a seta un spatiu dintre marginea Boxlayout-ului si copii. Default este zero.

```
self.buton1 = Button(text='Primul buton')
```

Aceasta linie de mai sus are rolul de a crea un widget . Acesta va afisa textul <<Primul buton>>.

```
self.add_widget(self.buton1)
```

Aceasta linie de mai sus are rolul de a adauga la root widgetul self.buton1 creat anterior.

```
self.buton2 = Button(text='Al doilea buton')
```

Aceasta linie de mai sus are rolul de a crea un widget . Acesta va afisa textul <<Al doilea buton>>.

```
self.add_widget(self.buton2)
```

Aceasta linie de mai sus are rolul de a adauga la root widgetul self.buton2 creat anterior.

# EXEMPLU BOXLAYOUT

INFOACADEMY.NET

```
class MyApp(App):  
    def build(self):  
        x=LoginScreen()  
        return x
```

```
if __name__ == '__main__':  
    MyApp().run()
```

Codul de mai sus exista pt. a crea si rula aplicatia kivy.

```
self.spacing=20
```

Spacing reprezinta un atribut al layout-ului ce indica spatiul dintre widget-urile child.

```
self.size_hint_x =0.5
```

```
self.size_hint_y =0.3
```

Size\_hint este un tuplu de doua valori ce au scopul de a indica o marime ce nu este impusa, ci doar daca poate fi aplicata.

# EXEMPLU STACKLAYOUT

INFOACADEMY.NET

```
StackLayout  [icon]  
1 # Program kivy7  
2 # Explica functiile kivy -- Stacklayout  
3 # Ion Studentul -- 1/26/13  
4  
5 import kivy  
6  
7 from kivy.app import App  
8 from kivy.uix.button import Button  
9 from kivy.uix.stacklayout import StackLayout  
10  
11 class Stack Layout(StackLayout):
```



```

>>> f = open("D:\\x.txt", "r")
>>> print f.read()
TOSHIBA Hardware Setup                                4.08.06.00
Microsoft Visual C++ 2010 x64 Redistributable - 10.0.40219 10.0.40219
Microsoft Visual C++ 2010 x86 Redistributable - 10.0.40219 10.0.40219
Python 2.7.8                                           2.7.8150
TOSHIBA ReelTime                                       1.7.17.64
Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.21005 12.0.21005
Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.21005 12.0.21005
>>> f
<open file 'D:\\x.txt', mode 'r' at 0x02A3FBD0>
>>> f.close()
>>> f
<closed file 'D:\\x.txt', mode 'r' at 0x02A3FBD0>
>>> with open("D:\\x.txt", "r") as f:
    data = f.read()

>>> print data
TOSHIBA Hardware Setup                                4.08.06.00
Microsoft Visual C++ 2010 x64 Redistributable - 10.0.40219 10.0.40219
Microsoft Visual C++ 2010 x86 Redistributable - 10.0.40219 10.0.40219
Python 2.7.8                                           2.7.8150
TOSHIBA ReelTime                                       1.7.17.64
Microsoft Visual C++ 2013 x86 Minimum Runtime - 12.0.21005 12.0.21005
Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.21005 12.0.21005
>>> f
<closed file 'D:\\x.txt', mode 'r' at 0x02A3FC80>

```

WITH

# CREAREA UNUI FUNDAL PENTRU WIDGET-UL DE TIP LAYOUT

INFOACADEMY.NET

```
BoxLayout2 ⌵  
1 # Program kivy6.py  
2 # Explica functiile kivy -- BoxLayout.py  
3 # Ion Studentul -- 1/26/13.py  
4  
5 import kivy  
6  
7 from kivy.app import App  
8 from kivy.uix.button import Button  
9 from kivy.uix.boxlayout import BoxLayout  
10 from kivy.graphics import Color, Rectangle
```

# CREAREA UNUI FUNDAL PENTRU [INFOACADEMY.NET](http://INFOACADEMY.NET)

## WIDGET-UL DE TIP LAYOUT

Layout-urile, prin natura lor, nu au o reprezentare vizuala deoarece nu au instructiuni de tip canvas(grafice). Totusi putem adauga instructiuni vizuale (canvas) destul de usor cum ar fi adaugarea de culoare ca fundal. Astfel:

Importam din modulul grafice color(eng. culoare) și rectangle(dreptunghi):

```
from kivy.graphics import Color, Rectangle
```

Adaugarea de canvas la instanta layout. Culoarele sunt exprimate ca fiind (R,G,B,A) cu valori de la 0 la 1. Adica R reprezinta rosu(aici 0.4), G reprezinta verde de la eng. green (aici 0.5) , B reprezinta albastru de la eng. Blue (aici 0.6) și A reprezinta luminozitatea de la eng alpha(aici 1). Dacă alpha este 0 atunci culoarea este negru indiferent de celelalte valori. Dacă toate sunt 1 atunci avem alb.

```
with layout_instance.canvas.before:
```

```
    Color(0.4, 0.5, 0.6, 1) # blue
```

```
    self.rect = Rectangle(size=layout_instance.size,  
                           pos=layout_instance.pos)
```

# CREAREA UNUI FUNDAL PENTRU [INFOACADEMY.NET](http://INFOACADEMY.NET)

## WIDGET-UL DE TIP LAYOUT

Vom desena un dreptunghi colorat doar la desenarea initiala a layout-ului. Pentru a fi siguri ca `self.rect` este desenat de fiecare data cand layout-ul isi modifica dimensiunea trebuie să ascultam după schimbări și să luăm o acțiune. Așa cum am văzut anterior, acest lucru este posibil cu un eveniment, deci `bind`. Apoi vom rula de fiecare dată când poziția sau dimensiunea se modifică `update_rect`:

```
# listen to size and position changes
```

```
layout_instance.bind(pos=update_rect, size=update_rect)
```

```
with layout_instance.canvas.before:
```

```
    Color(0, 1, 0, 1) # green; colors range from 0-1 instead of 0-255
```

```
    self.rect = Rectangle(size=layout_instance.size,  
                           pos=layout_instance.pos)
```

```
def update_rect(instance, value):
```

```
    instance.rect.pos = instance.pos
```

```
    instance.rect.size = instance.size
```

# AFISAREA UNUI TEXT

INFOACADEMY.NET

Pentru a schimba afisarea unui text putem utiliza Text Markup. Acestea sunt similare cu BBcore (Bulletin Board Code) utilizate în html.

```
from kivy.app import App
from kivy.uix.label import Label
```

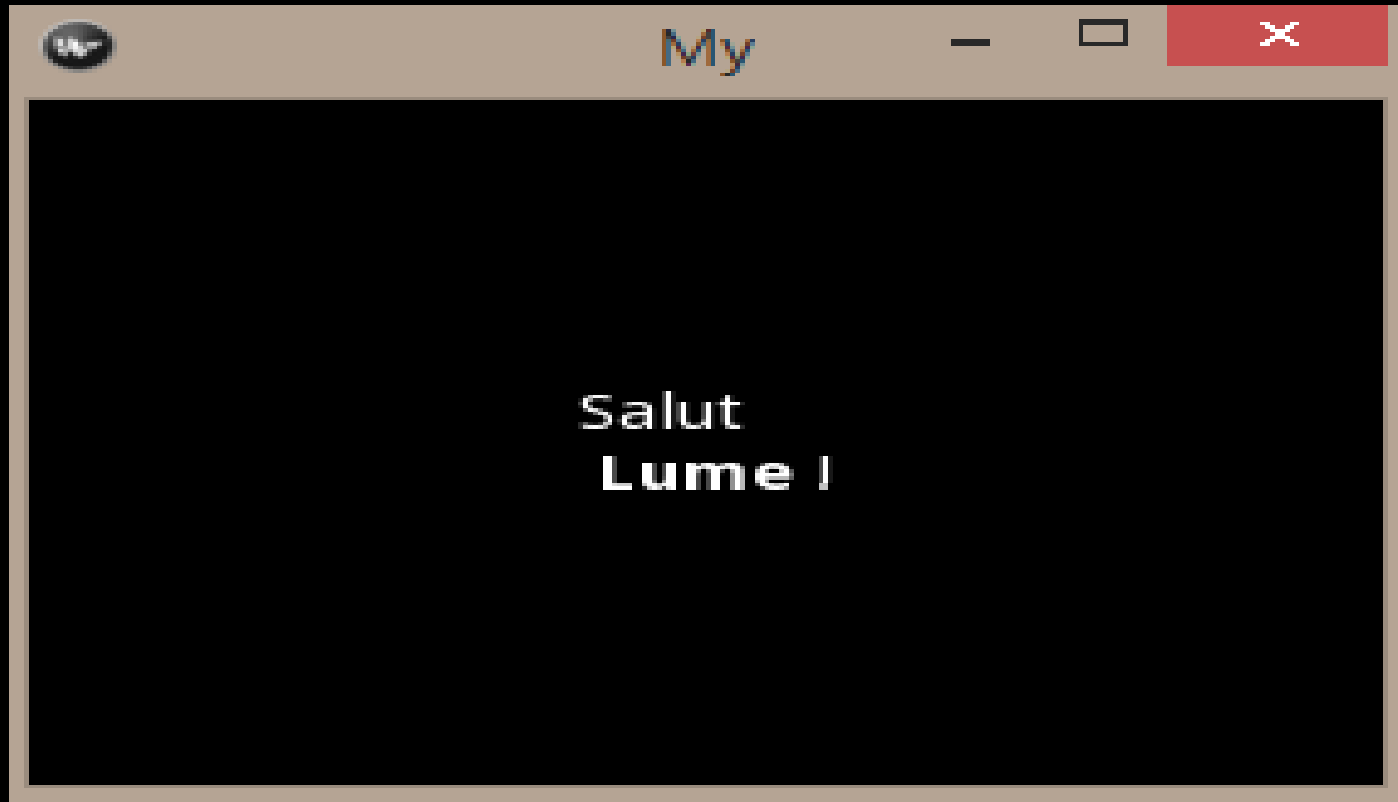
```
class MyApp(App):
    def build(self):
        return Label(text='Salut \n [b]Lume[/b] !', markup=True )
```

```
if __name__ == '__main__':
    MyApp().run()
```

# WIDGET-UL LABEL INFOACADEMY.NET

## AFISAREA UNUI TEXT

Pentru a schimba afisarea unui text putem utiliza [Text Markup](#). Acestea sunt similare cu BBcore (Bulletin Board Code) utilizate în html.





# WIDGET-UL LABEL

## AFISAREA UNUI TEXT

Pentru a schimba afisarea unui text putem utiliza Text Markup. Acestea sunt similare cu BBcore (Bulletin Board Code) utilizate în html.

```
from kivy.app import App
from kivy.uix.label import Label

class MyApp(App):
    def build(self):
        return Label(text='Salut \n [size=32]Lume![/size] ', markup=True )

if __name__ == '__main__':
    MyApp().run()
```

# WIDGET-UL LABEL INFOACADEMY.NET

## AFISAREA UNUI TEXT

Pentru a schimba afisarea unui text putem utiliza [Text Markup](#). Acestea sunt similare cu BBcore (Bulletin Board Code) utilizate în html.



# WIDGET-UL TEXT INPUT

Evenimente :

- `on_text_validate` adica la apasarea enter.
- `focus` – selectarea widget-ului.

Exemplu:

```
textinput = TextInput()
```

```
textinput.bind(on_text_validate=metoda_aplicata)
```

# WIDGET-UL TEXT INPUT

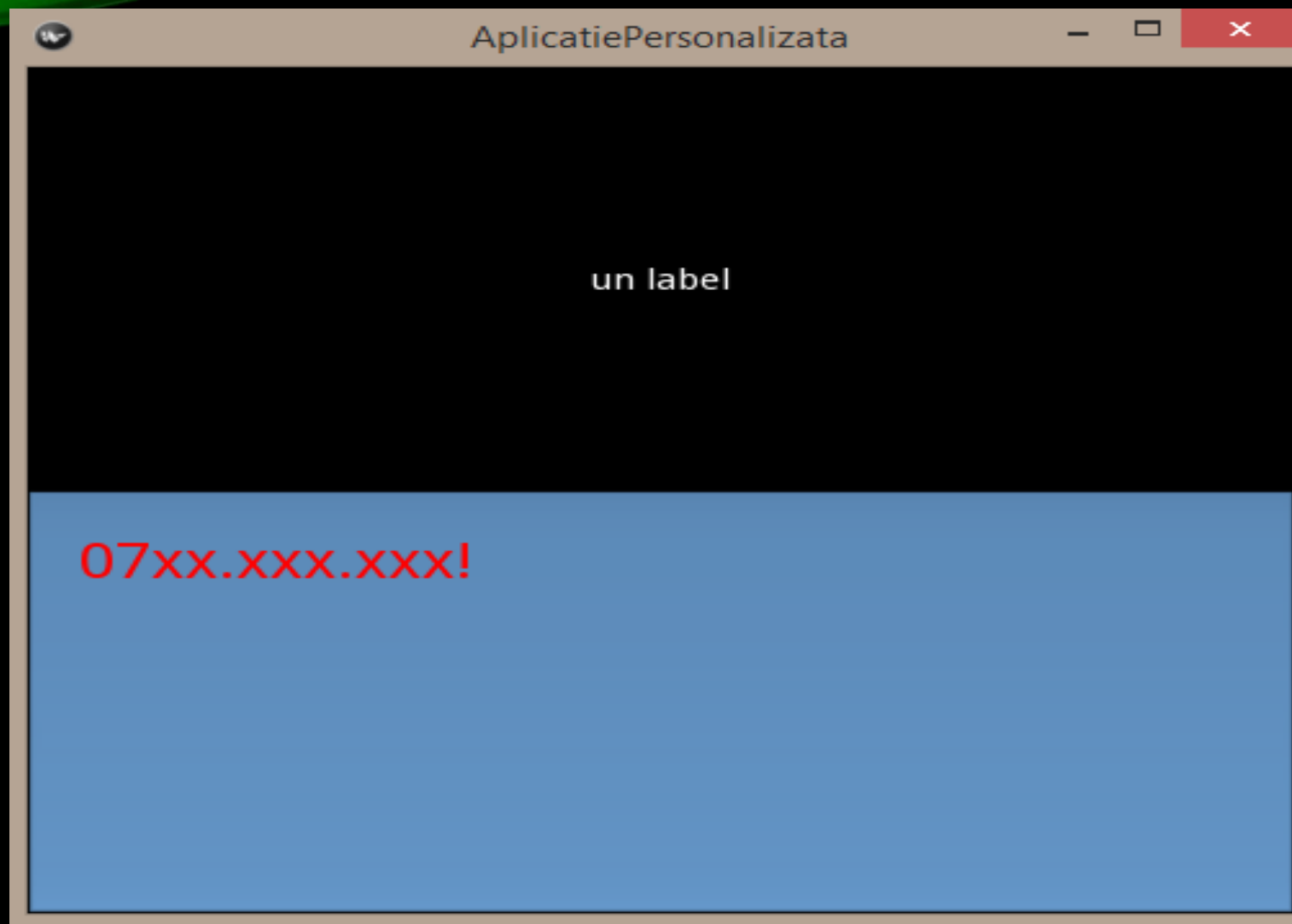
## Proprietati:

- multiline – accepta o singura linie (multiline = False) sau multiline (multiline = True)  
.Standard multiline= True
- Text-ul widget-ului textinput este stocat în obiect.text
- font\_size - marimea font-ului
- foreground\_color – culoarea textului
- background\_color – culoarea fundalului
- hint\_text\_color – culoarea hint text
- hint\_text – Afiseaza un text inainte să dai click sau să scrii ceva.
- padding - reprezinta spatierea dintre marginea label-ului și text. Poate avea și padding\_x și padding\_y.

# WIDGET-UL TEXT INPUT

```
TextInput1 ✖
1 # Program kivy9.py
2 # Explica functiile kivy -- text input.py
3 # Ion Studentul -- 1/26/13.py
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.textinput import TextInput
9
```

# WIDGET-UL TEXT INPUT



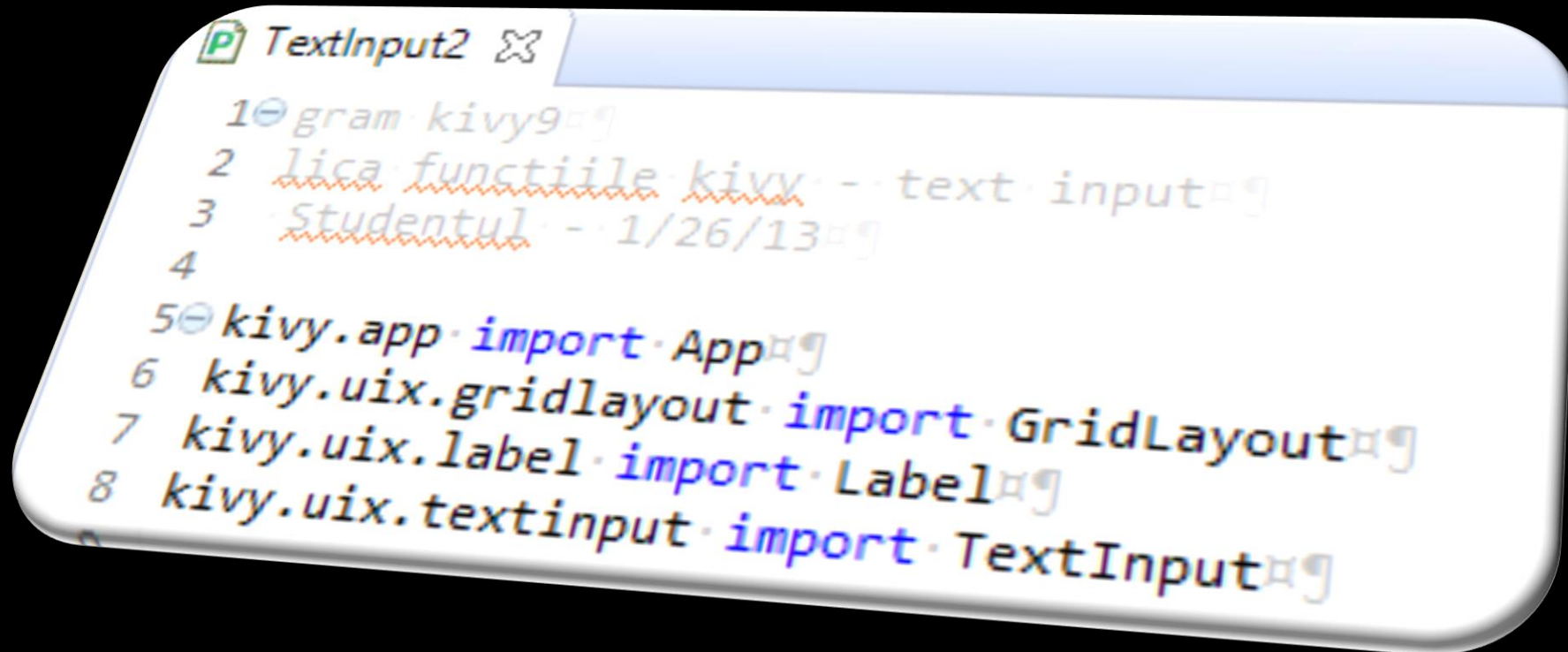
# WIDGET-UL TEXT INPUT

Alte caracteristici ale text input:

- password – textul introdus de utilizator va fi de forma \*\*\*\* (stelute)
- background\_normal – aceasta optiune permite adaugarea unei poze de fundal în loc de cea clasica. Starea este neselectata
- background\_active – aceasta optiune permite adaugarea unei poze de fundal în loc de cea clasica. Starea este selectata



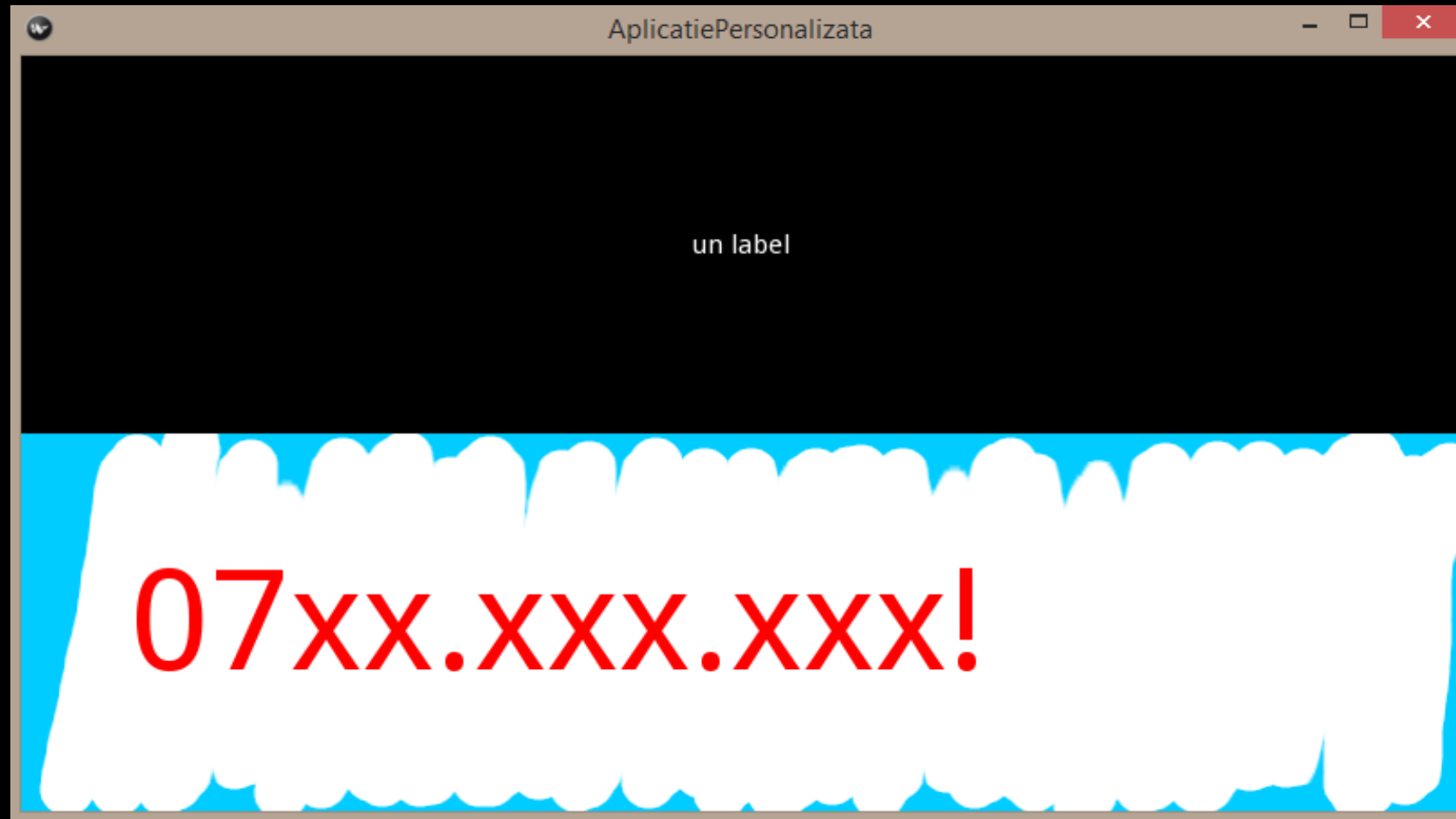
# WIDGET-UL TEXT INPUT



```
1 gram.kivy9
2 lica functiile kivy -- text input
3 Studentul -- 1/26/13
4
5 kivy.app import App
6 kivy.uix.gridlayout import GridLayout
7 kivy.uix.label import Label
8 kivy.uix.textinput import TextInput
```

# WIDGET-UL TEXT INPUT

INFOACADEMY.NET



- In cele ce urmeaza vom discuta despre checkbox. Acest widget poate fi utilizat ca checkbox sau ca buton radio. Mai jos regasim primul exemplu unde vom utiliza un checkbox. Ca eveniment vom folosi active. Acesta se activeaza de fiecare data cand schimbam starea butonului. De asemenea vom tine o inregistrare a starii prin variabila globala check1 pentru a sti starea actuala.

# WIDGET-UL CHECKBOX

```
checkbox1 ✖
1 # Program kivy9
2 # Explica functiile kivy -- text input
3 # Ion Studentul -- 1/26/13
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.checkbox import CheckBox
9 check1=0
10
```

# WIDGET-UL CHECKBOX

- Urmeaza exemplul doi în care utilizam checkbox-ul ca buton radio. *Obiectele de tip checkbox trebuie sa faca parte din acealasi grup. Aici group = "test"*

```
checkbox2 ✕
1 # Program kivy9
2 # Explica functiile kivy -- text input
3 # Ion Studentul -- 1/26/13
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.checkbox import CheckBox
9 check1=0
10
```



# WIDGET-UL CHECKBOX

- Similar cu widget-ul checkbox , dar mai apropiat de un buton normal este Toggle Button. Acesta are în plus fata de checkbox și posibilitatea de a avea un label asociat cu o descriere a acestei stari. Poate fi descris ca un buton care va fi în starea selectat sau deselectat. Toggle Button se poate utiliza ca un buton normal, avand acelasi proprietati și evenimente ca un buton normal.

```
Toggle_button1
```

```
1 # Program kivy9.py
2 # Explica functiile kivy -- Toggle button.py
3 # Ion Studentul -- 1/26/13.py
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.togglebutton import ToggleButton
```

# WIDGET-UL TOGGLE BUTTON

- Mai jos regasim un al doilea exemplu în care folosim Toggle button ca un buton radio. Obiectele de tip Toggle button trebuie sa faca parte din acealasi grup.
- Aici groupul este "Eu cu cine votez?"

```
Tuggle_button2 ✖  
3 # Ion Studentul -- 1/26/13  
4  
5 from kivy.app import App  
6 from kivy.uix.gridlayout import GridLayout  
7 from kivy.uix.label import Label  
8 from kivy.uix.togglebutton import ToggleButton  
9 check1=0  
10  
11
```



- Asa cum se poate vedea putem selecta una din aceste stari. Starea default este reprezentata de state="down". Doar un singur buton poate fi state down pe o perioada de timp dacă face parte din acelasi grup. Aici grupul este "Eu cu cine votez?". De asemenea, s-au utilizat proprietatile bold = true, am schimbat culoarea toggle button în rosu.
- Asa cum o să vedem și în cazul butoanelor, dar similar cu checkbox widget putem schimba imaginea clasica a unui toggle button. Astfel mai jos regasim un exemplu în care utilizam schimbarea imaginii default a unui toggle buton.

- Asa cum se poate vedea putem selecta una din aceste stari. Starea default este reprezentata de state="down". Doar un singur buton poate fi state down pe o perioada de timp dacă face parte din acelasi grup. Aici grupul este "Eu cu cine votez?". De asemenea, s-au utilizat proprietatile bold = true, am schimbat culoarea toggle button în rosu.
- Asa cum o să vedem și în cazul butoanelor, dar similar cu checkbox widget putem schimba imaginea clasica a unui toggle button. Astfel mai jos regasim un exemplu în care utilizam schimbarea imaginii default a unui toggle buton.
- cu ajutorul proprietatilor background\_normal și background\_down putem incarca doua imagini ce au ca rol crearea celor doua stari neapasat (normal) și apasat (down).

# WIDGET-UL TOGGLE BUTTON

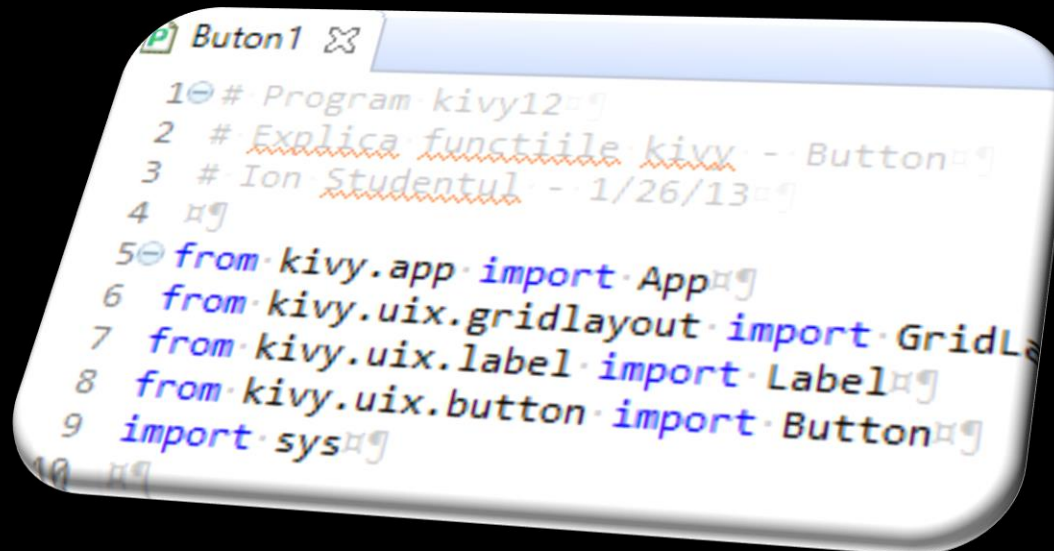
INFOACADEMY.NET

Tuggle\_button3

```
1 # Program kivy11.py
2 # Explica functiile kivy -- toggle button.py
3 # Ion Studentul -- 1/26/13.py
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.togglebutton import ToggleButton
9 check1=0
10 check2=0
```

# WIDGET-UL BUTTON

- Ultimul subiect propus azi este widget-ul button. Acesta se prezinta ca un widget ce are tot doua stari ca toggle cu diferenta ca nu ramane apasat. Mai jos regasim un exemplu complex ce surprinde și widget-ul button în actiune. Astfel putem adauga doua poze pentru cele doua stari ale butonului normal și down(apasat).
- Deci, cu ajutorul proprietatilor background\_normal și background\_down putem incarca doua imagini ce au ca rol crearea celor doua stari neapasat (normal) și apasat (down).



```
1 # Program kivy12.py
2 # Explica functiile kivy -- Button.py
3 # Ion Studentul -- 1/26/13.py
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.button import Button
9 import sys
```

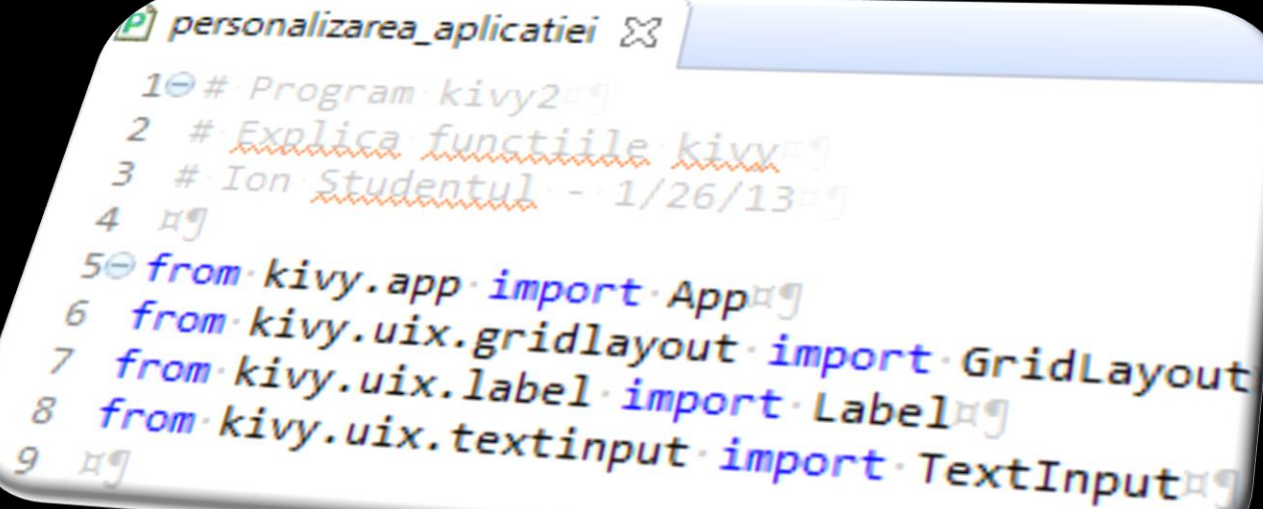


# PERSONALIZAREA UNEI APLICATII

În cele ce urmează vom crea o aplicație ce are ca scop autentificarea la o aplicație.

În prima fază cream un program care doar creează cadrul necesar autentificării. În a doua etapă vom crea un mic program ce va realiza și o autentificare.

Încercăm să manipulem un program pentru a arăta flexibilitatea pe care o deținem, dar și să înțelegem cum realizăm un program. Pentru prima fază programul propus este următorul:



```
1 # Program kivy2.py
2 # Explica functiile kivy.py
3 # Ion Studentul - 1/26/13.py
4
5 from kivy.app import App
6 from kivy.uix.gridlayout import GridLayout
7 from kivy.uix.label import Label
8 from kivy.uix.textinput import TextInput
9
```

# PERSONALIZAREA UNEI APLICATII

In următoarea linie importam GridLayout:

```
from kivy.uix.gridlayout import GridLayout
```

Aceasta clasa este folosita ca o bază pentru Root Widget (LoginScreen) :

```
class LoginScreen(GridLayout):
```

In clasa LoginScreen, va suprascrie \_\_ metoda \_\_init () pentru a adăuga widgeturi și pentru a defini comportamentul lor:

```
def __init__(self, **kwargs):  
    super(LoginScreen, self).__init__(**kwargs)
```

# PERSONALIZAREA UNEI APLICATII

```
self.cols = 2  
self.add_widget(Label(text='Utilizator '))  
self.username = TextInput(multiline=False)  
self.add_widget(self.username)  
self.add_widget(Label(text='Parola'))  
self.password = TextInput(password=True, multiline=False)  
self.add_widget(self.password)
```

În secțiunea de mai sus, cerem GridLayout să gestioneze copii săi în două coloane. Pe fiecare rând se adaugă o etichetă și un TextInput pentru numele de utilizator și parola.



# PERSONALIZAREA UNEI APLICATII

Etapă a doua presupune unirea programului învățat anterior în care afișăm o fereastră cu un label pentru care cream o protecție de securitate de tip user și parola.

```
personalizarea_aplicatiei2 ✕  
8 from kivy.uix.textinput import TextInput  
9  
10  
11 class LoginScreen(GridLayout):  
12  
13     def __init__(self, **kwargs):  
14         super(LoginScreen, self).__init__(**kw  
15         self.cols = 2
```

# PERSONALIZAREA UNEI APLICATII

Elementele noi în acest program se regasesc mai jos:

```
self.password.bind(on_text_validate= self.verific_user_si_parola)
```

Aceasta linie are rolul de a lega un eveniment (aici enter) de o metoda. Prin urmare, actiunea enter data de utilizator în widget-ul text input numit `self.password` va activa rulara metodei `verific_user_si_parola`.

*on\_text\_validate* este eclansat doar dacă *multiline* este False și utilizatorul tasteaza 'enter'

# PERSONALIZAREA UNEI APLICATII

Urmatoarea linie are scopul de a crea o noua metoda ce va fi apelata cand vom da enter în campul password.

```
def verific_user_si_parola(self,t):
```

Liniile de mai jos au rolul de a extrage textul introdus de utilizator

```
    text_user = self.username.text
```

```
    text_pass = self.password.text
```

Urmeaza o verificare a acestor valori extrase. Astfel, dacă user-ul introdus și parola introdusa au ambele valoarea test, vom sterge toate widget-urile anterioare apoi vom crea un alt widget cu un label.

```
    if (text_user == "test" and text_pass=="test"):
```

```
        self.clear_widgets()
```

```
        self.add_widget(Label(text='Bine ai venit!'))
```

# PERSONALIZAREA UNEI APLICATII

În cazul în care nu sunt îndeplinite criteriile cerute atunci vom selecta textul introdus de utilizator în câmpul username și îl vom șterge. La fel vom proceda și cu textul introdus în câmpul parola.

else:

```
self.username.select_all()
```

```
self.username.delete_selection()
```

```
self.password.select_all()
```

```
self.password.delete_selection()
```

# TEMA CLASA BOXLAYOUT

INFOACADEMY.NET

Exercitiu pe baza programului Boxlayout0.py:

- Sa modificati padding la 20
- Sa modificati orientarea la “vertical”
- Sa adaugati spacing cu valoarea 20
- Sa adaugati size\_hint\_x cu valoarea 0.5 și size\_hint\_y cu valoarea 0.3, apoi să rulați

Creati un program kivy cu un layout de tip boxlayout care sa afiseze 2 widget-uri de tip buton cu textele:

- <<Am plecat>>
- <<Am venit>>

Acestea sa aiba culoarea de background albastra. La apasare (evenimentul `object.on_press`) sa printeze texul butonului.

Creati un program kivy cu un layout de tip boxlayout care sa afiseze 2 widget-uri de tip buton cu textele:

- <<Am plecat>>
- <<Am venit>>

Acestea sa aiba culoarea de background albastra. La apasarea butonului cu textul <<Am plecat>> (evenimentul `obiect.on_press`) sa schimbam culoarea scrisului butoanelor in rosu (`self.buton.color=(1,0,0,1)`)

La apasarea butonului cu textul <<Am venit>> (evenimentul `obiect.on_press`) sa schimbam culoarea scrisului butoanelor in verde(`self.buton.color=(0,1,0,1)`)



VA MULTUMESC PENTRU PARTICIPARE

**La revedere!**