Aplicații Multimedia

Laborator 1

Interfețe Multimedia cu Utilizatorul

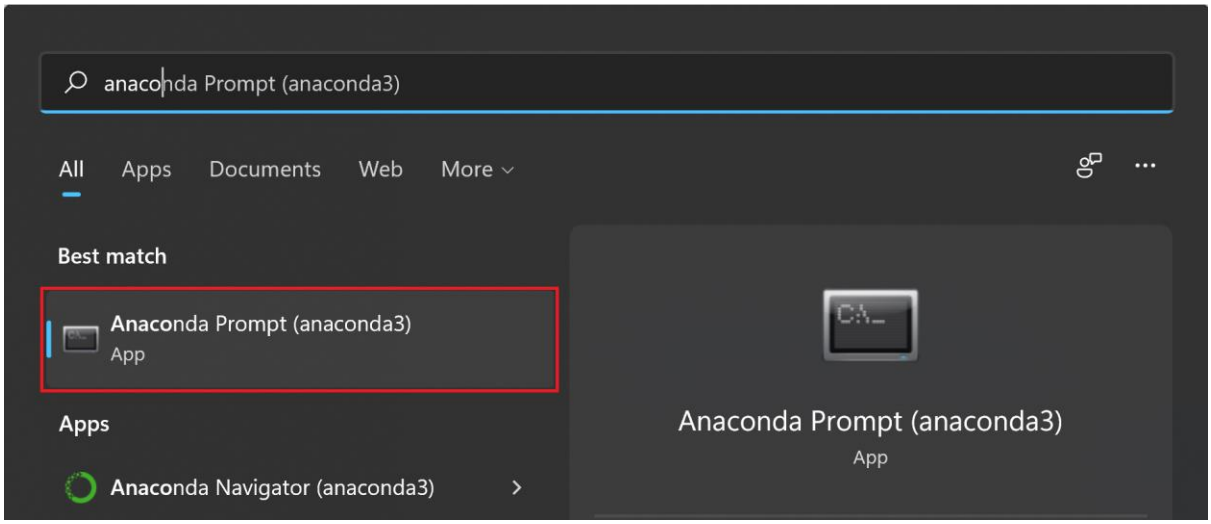
Cuprins

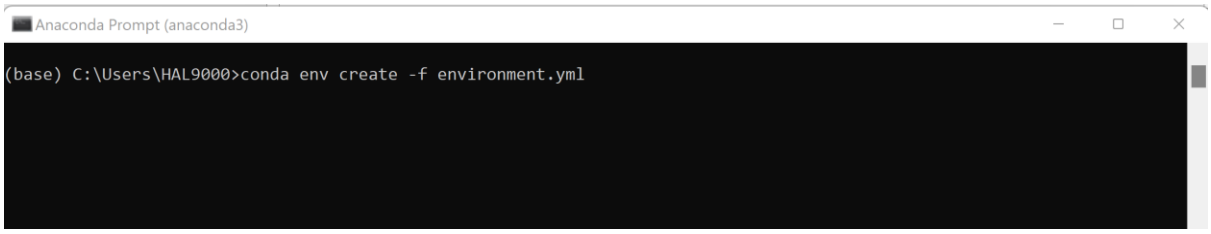
Instalarea suitei software

1. Instalare Anaconda, program ce gestionează pachetele și medii virtuale de Python
   Se poate descărca de la adresa: https://www.anaconda.com/products/individual

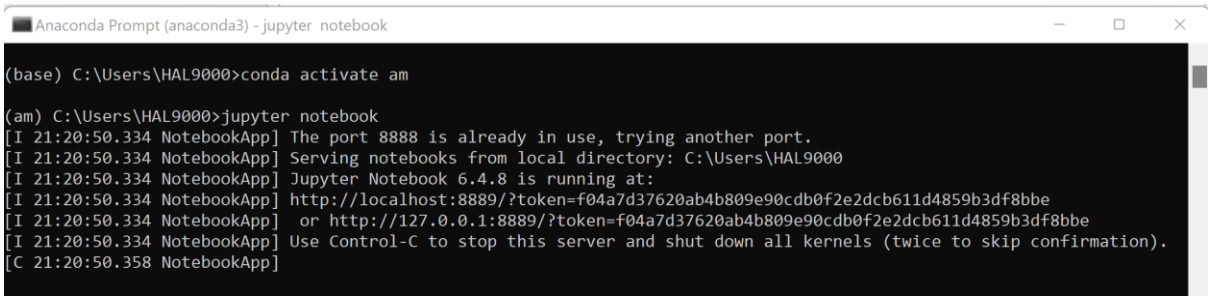
Notă: Se poate folosi orice program sau mediu care permite rularea de cod Python și instalarea de pachete.

2. Crearea unui environment folosind fișierul environment.yml.
   Se deschide consola Anaconda Prompt și se rulează comanda ***conda env create -f environment.yml***.





3. După finalizare, se activează mediul virtual ***am***, folosind comanda ***conda activate am***.
4. Pentru folosirea notebook-ului suport, ***Laborator 1.ipynb***, se rulează comanda ***jupyter notebook***.

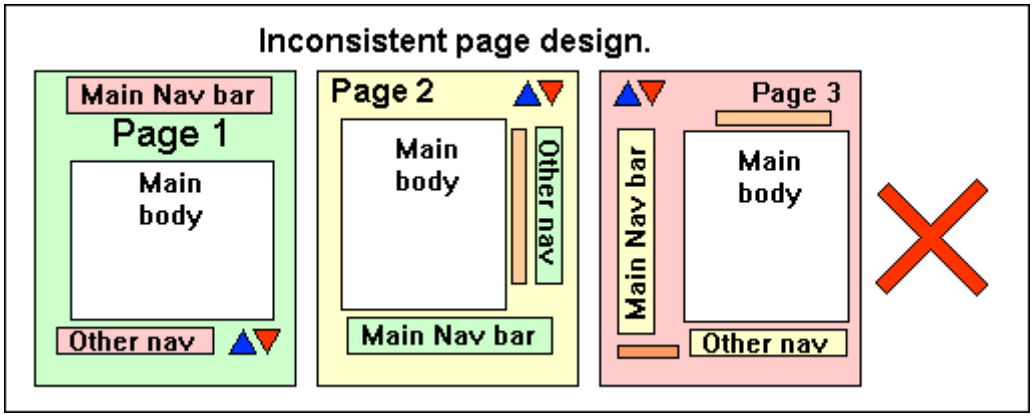


Generalități interfețe multimedia cu utilizatorul

O interfață multimedia cu utilizatorul este o interfață pe un sistem de calcul care comunică cu un utilizator folosind numeroase medii, acestea fiind elementul care dictează cât de bine este realizată interacțiunea. Interfețele multimedia pot îngloba scris, elemente vizuale (imagini, video), elemente auditive. În cadrul acestui laborator ne vom concentra pe interfețele grafice pentru aplicații.

Când vine vorba despre interfețe grafice, se poate discuta despre principii de design, care au legătură cu modul în care informația este structurată și prezentată pe o pagină. Câteva dintre aceste principii sunt:

- Consistența
- Claritatea
- Ușurința de a naviga (introducerea de trimiteri si referințe)
- Flexibilitatea (nu poate fi utilizată numai de o categorie de utilizatori cu un set de cunoștințe specializat)
- Accesibilitatea pentru mai multe categorii de utilizatori



Exemplu de design de interfață cu mai multe pagini care nu respectă regula de consistență.

Sursă

Crearea de aplicații în Python folosind framework-ul Qt

Qt este un framework scris în C++ care permite crearea de aplicații pe mai multe platforme, cum ar fi Linux, OS X, Windows, Android, iOS, etc. Printre altele, acesta facilitează realizarea de interfețe grafice, având elemente precum butoane, meniuri, grafice și altele, ce pot fi create programatic. În același timp, există și un program grafic interactiv, numit Qt Designer, prin intermediul căruia se pot genera secvențele de cod necesare pentru elementele de interfață, acesta nefăcând însă scopul laboratorului curent.

PyQt este un binding de Python pentru framework-ul Qt, versiunea folosită în aplicațiile de laborator fiind PyQt5.

Semnale și Sloturi

Un aspect important în Qt atunci când sunt folosite elemente precum butoane, la a căror apăsare să se producă o schimbare într-un alt element din aplicație (de cele mai multe ori schimbarea fiind necesar să se producă pe un alt obiect decât cel care conține butonul) este reprezentată de conceptul de semnale și sloturi, acestea fiind o alternativă la callback-uri

Un semnal este emis atunci când s-a produs un eveniment (a fost apăsat un buton), iar un slot este o funcție care este apelată ca răspuns la emiterea unui semnal. În codul de laborator, variabila ***cameraOption*** din clasa ***MainWindow*** este un semnal, iar metoda ***set_option*** din clasa ***Thread*** este un slot. De menționat este faptul că semnătura unui semnal trebuie să se potrivească cu semnătura unui slot, ca în imaginea de mai jos:

```
# Semnal
cameraOption = pyqtSignal(str)

# Slot
@pyqtSlot(str)
    def set_option(self, option):
        [...]
```

Aplicații și thread-uri de execuție

În cadrul laboratorului, vom lucra cu o aplicație care are o fereastră principală (instanță a clasei ***MainWindow***), pe care dorim să afișăm un stream video de la o cameră, în diverse stiluri, cum ar fi în format color, în tonuri de gri, doar conturul obiectelor, etc, stilul afișat fiind selectat prin butoane din interfața grafică.

Deoarece aplicația și fereastra principală se ocupă cu afișarea informației, desenarea butoanelor și ascultarea de evenimente pentru ele, vom avea nevoie să creăm un *thread* de execuție separat care să citească datele de la cameră și să aplice eventualele stilizări alese. Putem să ne gândim la un *thread* de execuție ca la un proces lightweight, care există doar în cadrul unui proces. Acesta nu are resurse proprii, ci le folosește pe ale procesului (memorie, acces la fișiere, semnale, etc.). Toate *thread*-urile unu proces îi folosesc resursele în comun.

Comunicația dintre aplicația principală și *thread*-ul asociat camerei se realizează prin intermediul a două perechi semnal-slot, ca în figura de mai jos: