Obiective

- Cunoașterea conceptelor fundamentale programării
- Introducere concepte de bază legate de ingineria software (design, arhitectură, implementare și întreținere)
- Înțelegerea instrumentelor software folosite în dezvoltarea de aplicații
- Învățarea limbajului Python și utilizarea lui pentru implementarea, testarea, rularea, depanarea de programe.
- Însușirea/Îmbunătățirea stilului de programare.

Conținut

Scripting
Python basics, file manipulation, unix/mac/windows

Desktop GUI applications
PyQt, Tkinter

Web Development
Django, Flask

Mobile Application
Kivy, PyQt, Toga

Data Science
NumPy, matplotlib,pandas, scikitlearn

Game Development
PyGame, Panda3d

Evaluare

Lab (30%) - o notă pe activitatea de laborator din timpul semestrului.
T (30%) - examen practic (în sesiune)
E (40%) - examen scris (în sesiune)

Pot participa la examen doar cei care au minim **12** prezente la laborator si minim **10** prezente la seminar

Pentru promovare trebuie să aveți cel puțin nota 5 la toate (Lab, T, E >= 5)
Toate activitățile sunt obligatorii
Dacă nu obțineți nota 5 la laborator nu puteți intra in examen in sesiunea normală.

Restanțe
În sesiunea de restanțe puteți preda laboratoare dar nota maximă este 5.
Se poate re-susține examenul practic
Se poate re-susține examenul scris

Data examen

Reguli / desfășurare

Să aveți la voi un act de identitate (carnet de student, buletin, pașaport)

Să aveți toate taxele plătite la zi (daca este cazul)

Se dă examenul scris apoi examen practic cu o pauză de aproximativ 1 oră intre ele

Să vă prezentați la examen înainte cu minim 10 minute de ora stabilită.

Nu copiați.
Dacă observați orice tentativă de fraudare / nereguli să le semnalați in timpul examenului.
Proiectele de la examenul practic se vor verifica pentru a identifica plagiatul. Nota se anulează in caz de fraudă.

Sugestii

Veniți odihniți la examen.
Aduceți cu voi o sticlă de apă pentru hidratare.
Sa mâncați ceva înainte de examen sau in pauza intre scris si practic.
Nu vă stresați excesiv. Tot timpul este o altă șansa sa mai dai examenul. Nu se termina lumea cu un examen ratat.

Elemente de bază Python:

- Instrucțiuni: =, if, while, for
- tipuri de date: integer, real, string, list, dictionary
- funcții: definiție, transmiterea de parameteri
- tipuri definite de utilizator: classe
- excepții

1) Care este rezultatul execuției pentru următorul cod Python:

```
def f(l):
    print ("A")
    if l==[]:
        raise ValueError()
    print ("B")

def start():
    l = []
    try:
        print ("A")
        f(l)
        print ("D")
    except ValueError:
        print ("C")
start()
```

2)

```
class A:
    def f(self, l,nr):
        l.append(nr)
class B:
    def g(self, l, nr):
        nr=nr-1
        l = l+[-2]

a = A()
b = B()
l = [1,2]

c = -1
a.f(l,6)
b.g(l,c)
print (l,c)
```

Algoritmi - specificații/test/implementare

Variante

Se dă specificația - implementați și testați

Se dă implementare – specificați și testați

Se dă o funcție de test: specificați și implementați

Exemple:

Implementați și testați funcția care are specificația

```
 """
    Calculeaza suma elementelor pare
    l - lista de numere
    return un numar, suma elementelor pare
    aruncă ValueError daca lista nu conține numere pare
"""
```

Specificați și testați funcția:

```
def f(n):
    d = 2
    while (d<n-1) and n%d>0: d+=1
    return d>=n-1
```

Complexitate

Se dă o funcție Pyton – se cere să analizați complexitatea ca timp si/sau spațiu de memorie

Exemple

Analizați complexitatea ca timp de execuție pentru următoarea funcție:

1)

```
def f(x):
    m = len(x)-1
    s =  0
    while m>=1:
        s += x[m]
        m = m//3
    return s
```

2)

```
def f(x):
    found = False
    n = len(x)-1
    while n!=0 and not found:
        if x[n]==7: found=True
        else: n=n-1
    return found
```

2)

```
def prel(x,i,j):
    if (i<j-1):
        k1 = (j-i+1)//3
        k2 = 2*k1
        prel(x,i,i+k1)
        prel(x,i+k1+1,i+k2)
        prel(x,i+k2+1,j)
        rez = 0
        for k in range(i,j):
            rez+=x[k]
        return rez
    else:
        return x[i]
```

Căutări/sortări complexitatea lor – subiect eliminatoriu

Nerezolvarea acestui subiect înseamnă automat ca ai picat examenul scris.

Implementați una din funcțiile de căutare/sortare studiate

- căutare:
  - secvențială
  - succesivă
  - binară
- sortări:
  - sortare prin selecție
  - sortare prin selecție directă
  - sortare prin inserție
  - metoda bulelor
  - quicksort
  - mergesort

Exemple:
1 Scrieți o funcție care sortează o listă de numere și are complexitatea ca timp de execuție în caz defavorabil n^2

2) O listă de cumpărături conține produse:

Produs = Product name, Product type, Price

Scrieți o funcție de sortare care permite sortarea listei de cumpărături:

- alfabetic după tip
- descrescător după preț

Technici de programare

- Backtracking
- Divide et. Impera
- Greedy
- Dynamic Programming

Se dă o problemă – se cere rezolvarea folosind o metoda dată

Alegeți cea mai potrivită metodă de rezolvare pentru o problemă dată

Se poate cere:

- indicați schematic soluția
  - backtracking: soluție candidat, valid, soluție
  - Divide et impera: recurența
  - Greedy: soluție candidat, valid, soluție, funcția de selecție greedy
  - Dynamic Programming: principiul optimalității, recurența care descrie algoritmul

implementare pentru elementele de bază sau implementare pentru tot (alg, elemente de bază)

Exemplu:

1 Se da o lista de numere, să se determine cel mai lung subșir cu elemente pare folosind programare dinamică.

2 Alegeți tehnica cea mai potrivită (Backtracking, Divide et. Impera, Greedy, Dynamic Programming) pentru a calcula suma numerelor pare într-o lista.

Examen practic

Scrieți o aplicație pentru gestiunea activităților de laborator

Aplicația permite cadrului didactic sa efectueze repetat operațiile:
- vizualizare listă studenți
-căutare student după id
-assignare laborator la un student
- aplicația semnalează (și nu permite) adăugarea de două probleme la același laborator (lab number)
- vizualizare toate laboratoare de la un student
- vizualizare studenți si laboratoare asignate pentru un laborator dat

Studenții și laboratoarele sunt stocate in 2 fișiere: “student.txt”, “labs.txt”

Arhitectura aplicației: