UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică

ALGORITMI
ŞI
PROGRAMARE

Cursul 2

Programare Procedurală

Ionescu Vlad
vlad.ionescu@ubbcluj.ro

Întrebarea +feedback prima săptămână

- Adresați întrebarea de început
- Feedback

Programarea procedurală

- Este o **paradigmă** de programare

- Paradigmă:
  - O colecție de concepte, șabloane, teorii, metode, abordări, standarde

- Programarea procedurală:
  - Organizarea programelor în proceduri, funcții

Funcții

- Un bloc de instrucțiuni care se vor executa doar la apelul funcției
- Au parametri (date de intrare) și returnează valori (date de ieșire)
- Trebuie să aibă **nume sugestive**
- Trebuie să aibă **specificații**
- Ar fi bine să respecte PEP 8: https://www.python.org/dev/peps/pep-0008/

Funcții – parametri și return **Exemplu**

- O funcție returnează una sau mai multe valori prin instrucțiunea **return**

- O funcție fără nicio instrucțiune **return** va returna valoarea specială **None**

Funcții – nume sugestive **Exemplu**

- Alegeți un stil consistent de denumire dintre:
  - snake_case – cel mai des întâlnit în Python
  - camelCase
  - PascalCase
- Evitați denumiri de un singur caracter, prescurtări, acronime
- Numele ar trebui să sugereze ce face funcția
- > Lab4: fără warning-uri PEP 8 in PyCharm

Funcții – specificații **Exemplu**

- Descriu **CE** face funcția
- Descriu datele de intrare și tipul lor
- Descriu datele de ieșire
- Pot oferi alte informații utile, de genul:
  - Excepții ridicate
  - Complexitatea
  - Algoritmi folosiți
- Nu conțin detalii de implementare

Funcții – vizibilitatea variabilelor **Exemplu**

- **Scope** – locul în care o variabilă este vizibilă

- În Python avem următoarele scope-uri:
  - Funcții
  - Module
  - Clase
  - Alte expresii speciale

- Nu există **block scope**

Funcții – variabile globale **Exemplu**

- Se referă la variabile cu scope de modul / fișier
- Orice funcție poate citi variabile globale
- Dar nu le poate modifica dacă nu specifică explicit acest lucru
- **Dar dacă face oricare dintre astea, nu e bine!**

Funcții – transmiterea parametrilor **Exemplu**

- Parametri formali vs parametri actuali
- Transmiterea parametrilor este **Call-by-Object**
- "Referințe transmise prin valoare"

Funcții – efecte secundare **Exemplu**

- O funcție are efecte secundare dacă modifică mediul apelant:
  - Modifică parametrii actuali
  - Modifică fișiere

- Evitarea efectelor secundare este considerată o bună practică:
  - Acestea sunt greu de urmărit și de gestionat
  - Cresc șansele de a introduce bug-uri
  - Sunt greu de depanat
  - Aproape întotdeauna există alternative mai bune

Funcții – alte elemente **Exemplu**

- Valori default
- Supraîncarcare
- Număr variabil de argumente: *args, **kw_args
- Unpacking

Dezvoltarea de software

- Roluri în ingineria soft
  - Programator/dezvoltator
    - Scrie/dezvoltă programe pentru utilizatori
  - Client
    - Cel interesat/afectat de rezultatele unui proiect
  - Utilizator
    - Rulează programe pe computer

- Procesul dezvoltării unui soft
  - include construirea, lansarea și întreținerea unui soft
  - indică pașii care trebuie efectuați și ordinea lor

Dezvoltarea dirijată de funcționalități

- Pași în rezolvarea unei probleme
  - Enunț pentru definirea problemei
  - Cerințe
  - Scenariu de utilizare a aplicației
  - Stabilirea funcționalităților și împărțirea lor pe iterații
  - Identificarea de activităţi (ale fiecărei funcţionalităţi) şi descrierea lor

Dezvoltarea dirijată de funcționalități

- Enunț pentru definirea problemei
  - Surtă descriere a problemei
  - *Un profesor (client) are nevoie de o aplicație pentru studenții (utilizatorii) care învață să găsească cel mai mic număr prim mai mare decât un număr natural n dat.*

- Cerințe
  - Definesc în detaliu ceea ce este necesar din perspectiva clientului, respectiv ce trebuie să facă aplicația
    - Ce dorește clientul?
    - Ce trebuie să includă sistemul pentru a satisface cerințele clientului?
  - Stabilirea informaților de intrare și ieșire ale aplicației
  - *Date de intrare: **n** – număr natural*
  - *Date de ieşire: cel mai mic număr prim mai mare decât **n***
  - Scenariu de utilizare a aplicației

Dezvoltarea dirijată de funcționalități

- Cerințele bine formulate asigură funcționarea sistemului așa cum dorește clientul
- Accent pe lista de funcționalități pe care trebuie să le execute sistemul
- Lista de funcționalități trebuie să clarifice ambiguitățile din enunțul problemei

Dezvoltarea dirijată de funcționalități

- Stabilirea funcționalităților și planificarea iterațiilor
  - Funcționalitatea
    - definită ca o funcție client
    - exprimată în forma acțiune rezultat obiect
      - Acțiunea – o funcție pe care aplicația trebuie să o furnizeze; a nu se confunda cu funcțiile din Python!
      - Rezultatul – este obținut în urma execuției funcției
      - Obiect – o entitate în care aplicația implementează funcția
    - poate fi implementată în câteva ore – complexitate redusă
    - *F1: găsirea celui mai mic număr prim mai mic decât un număr natural n dat*.
  - Iterația
    - O perioadă de timp în cadrul căreia se realizează o versiune stabilă și executabilă a unui produs, împreună cu documentația suport
    - Ajută la planificarea temporală a ansamblului de funcționalități
    - O colecție de funcționalități
    - *I1 = F1, [F2, F3, …]*

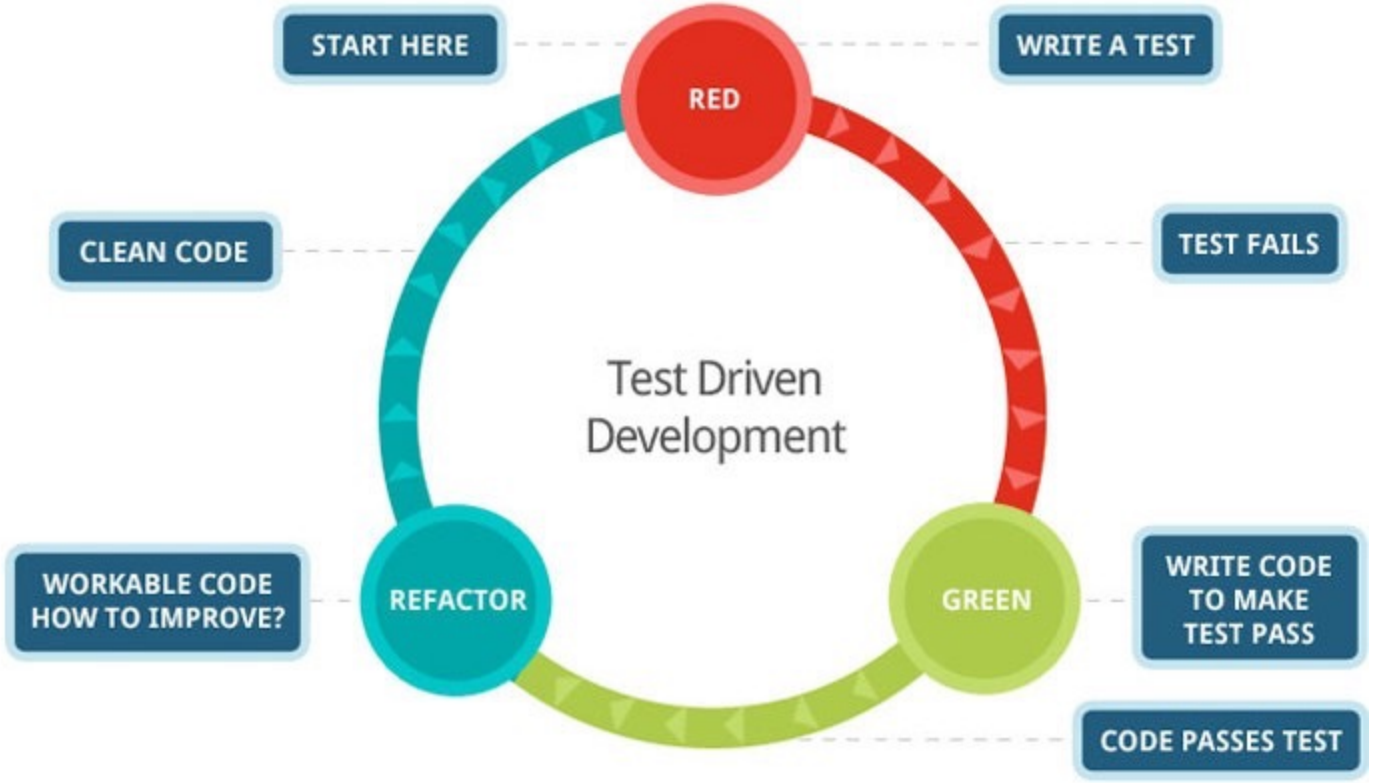
Dezvoltarea dirijată de funcționalități

- Listă de activităţi (ale fiecărei funcţionalităţi) şi descrierea lor
  - Recomandări:
    - Definirea unei activități pentru fiecare operație
    - Definirea unei activități pentru interacțiunea Utilizator – Program (Interfața utilizator – *user interface, UI*)
    - Definirea unei activități pentru operațiile UI
    - **Determinarea dependențelor între activități**
  - *A1: verificarea calității de număr prim pentru o valoare dată*
  - *A2: găsirea celui mai mic număr prim mai mic decât un număr natural n dat*
  - *A3: implementarea inițializării unui număr, căutării celui mai mic nr prim mai mic decât n și furnizarea rezultatului*
  - *A4: implementarea UI*

Dezvoltarea dirijată de teste – TDD

Exemplu

- Scrierea testelor pe baza specificațiilor, înainte de implementare



https://blog.usejournal.com/test-driven-development-understanding-the-business-better-9c4cae4cb990

TDD – refactorizarea **Exemplu**

- Restructurarea codului fără a-i modifica funcționalitatea

- Are rolul de a îmbunătăți calitatea codului, de obicei prin eliminarea unor **code smells**:
  - Cod duplicat
  - Funcții prea mari
  - Nume nesugestive
  - Cod ineficient

TDD – ce cod poate fi testat? **Exemplu**

- În general: funcțiile fără efecte secundare:
  - Funcții care nu așteaptă date de la utilizator
  - Funcții care nu afișează pe ecran
  - Funcțiile care fac parte din partea **logică** / **de calcule** a aplicației
- Funcțiile din partea de **UI** le apelează pe cele de logică, nu invers!
- Acest lucru contribuie și la **generalitatea** codului.

Q & A