UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică

ALGORITMI ŞI PROGRAMARE

Introducere în git și în programare prin github respectiv limbajul Python

Ionescu Vlad
vlad.ionescu@ubbcluj.ro

Git

- Git este un VCS – Version Control System.
- Un VCS ne permite să gestionăm modificări ale fișierelor de cod, lucru esențial în proiecte mari.
- Instalați git pe sistemul vostru: https://git-scm.com/downloads
- Citiți și urmați cartea pro git de pe site-ul oficial: https://git-scm.com/book/en/v2. La această materie vom folosi doar funcționalități descrise până la 2.5 inclusiv.
- Comenzile necesare vor fi prezentate și în cadrul laboratoarelor.

Github

- Github este un site care găzduiește repository-uri de git și oferă diverse funcționalități de colaborare.
- Noi vom folosi github classroom pentru încărcarea temelor de laborator și a examenelor.
- Vom folosi assignments pentru temele de laborator. Fiți foarte atenți la instrucțiunilor din assignments, deoarece folosim evaluatoare automate pentru unele cerințe.
- Acceptați assignmentul pentru fiecare laborator. Dacă vi se cere să vă asociați contul și nu vă găsiți în listă, cereți să fiți adăugat de către profesorul de laborator.
- Verificați pagina de classroom la fiecare sfârșit de laborator pentru următorul assignment.

Comunicarea în timpul semestrului

- Niciun curs nu va începe până nu adresați o întrebare
- Îmi puteți scrie orice întrebări, nelămuriri, nemulțumiri și reclamații pe Teams.
  - Dacă doriți să rămâneți anonim, contactați șeful de grupă / de an, care îmi va transmite informația.

Limbajul Python

- Ce este Programarea?
- Ce este Python?
  - Un limbaj high level
  - Multi-paradigm
  - Strongly typed
  - Dynamically typed
  - Interpreted
  - Garbage collected
  - Accent pe înțelegerea codului: *code readability*
- Instalați o versiune >= 3.7:
  - https://www.python.org/
  - Tot aici găsiți documentație și un tutorial

Limbajul Python

- The Zen of Python:
  - https://www.python.org/dev/peps/pep-0020/
- Vom pune accent pe scrierea codului într-un anumit mod, nu pe eficiență.
- *"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. Code for readability."* – John Woods, 1991.
- Vom începe cu editorul IDLE sau cu programe rulate de la consolă.
- Instalați PyCharm sau alt editor cât mai rapid: obligatoriu din săptămâna 2.

Programarea în general

- Ce este programarea?
  - Hardware
    - Computere (desktop, laptop, etc) și dispozitive înrudite
  - Software
    - Programe sau sisteme care rulează pe un hardware
  - Limbaj de programare
    - Un limbaj prin care scriem Software
  - Python
    - Un limbaj de programare de nivel înalt
  - Interpretatorul Python
    - Un program care permite rularea/interpretarea instrucțiunilor Python
  - Bibliotecile Python
    - Subprograme și tipuri de date deja definite de alți programatori

Elemente de bază Python

- Exemplu

```
# takes two integers and prints the sum of them
a = 2000
b = 19
c = a + b
print("The sum of ", a, " + ", b, " is ", c)
```

- Elemente lexicale
  - Un program Python poate fi alcătuit din mai multe linii de cod
  - Comentarii
    - încep cu # și țin până la sfârșitul liniei
    - încep cu ’’’ și țin mai multe rânduri, până la un nou ’’’
  - Identificatori
    - secvențe de caractere (litere, cifre, _) care încep cu o literă sau cu _
  - Literali
    - notații pentru valorile constante sau pentru tipuri definite de utilizator

Elemente de bază Python

- Modelul de date
  - Toate datele unui program Python – obiecte
  - Un obiect are
    - o identitate – adresa lui în memorie
    - un tip
    - o valoare
  - Odată creat, identitatea și tipul obiectului nu mai pot fi modificate
  - Valoarile unor obiecte se pot modifica
    - Obiecte mutabile
    - Obiecte ne-mutabile

Elemente de bază Python

- Tipuri de date
  - Domeniu: mulțimea valorilor posibile
  - Operații posibile
- Tipuri de **date numerice** - ne-mutabile
  - int
    - Domeniu: întregi pozitivi și negativi
    - Operații: +, -, *, /, **, //
    - Literali: 1, 2
  - bool
    - Domeniu: True sau False
    - Operații: logice (and, or, not,...)
    - Literali: True, False, 1, 0
  - float
    - Domeniu: numere reale în dublă precizie
    - Operații: +, -, *, /, **, //
    - Literali: 3.14, -0.25

Elemente de bază Python

- Tipuri de **date secvențiale**
  - Stringuri – ne-mutabile
    - Domeniu: șiruri de caractere
    - Operații: concatenare, căutare
    - Literali: ”abc”

```
#concatenate
a = "mate"
b = "info"
c = a + b
print(c)

#search
n = c.find("at")
print(n)
m = c.rfind("info")
print(m)
```

Elemente de bază Python

- Tipuri de **date secvențiale**
  - Liste – mutabile
    - Domeniu: secvențe de elemente (similare sau diferite ca tip) separate prin „,” și încadrate de „[ ]”
    - Operații
      - Creare (manuală, *range*)
      - Accesare (index, *len*) și modificare elemente
      - Eliminare (*pop*) și inserție (*insert*) de elemente
      - Slicing și încapsulare
      - Utilizare ca stive (*append, pop*)

```
# create
a = [1, 2, 'a']
print(a)
x, y, z = a
print(x, y, z)
print(a[0]) # indices: 0, 1, ..., len(a) - 1
print('last element = ', a[len(a)-1])
a[1] = 3 # lists are mutable
print(a)
```

Elemente de bază Python

- Tipuri de **date secvențiale**
  - Tuple – ne-mutabile: “liste nemutabile”

```
# Tuples are immutable sequences
# A tuple consists of a number of
# values separated by commas
# tuple packing
t = 12, 21, 'ab'
print(t[0])
# empty tuple (0 items)
empty = ()

# sequence unpacking
x, y, z = t
print(x, y, z)
```

```
# tuple with one item
singleton = (12,)
print (singleton)
print (len(singleton))
#tuple in a for
t = 1,2,3
for el in t:
    print(el)

# Tuples may be nested
u = t, (23, 32)
print(u)
```

Elemente de bază Python

- Tipuri de **date secvențiale**
  - Dicționare – mutabile
    - Domeniu: mulțimi ne-ordonate de perechi (cheie, valoare) cu chei unice
    - Operații:
      - Creare
      - Accesarea valorii pentru o cheie dată
      - Adăugarea/modificarea/eliminarea unei perechi (cheie, valoare)
      - Verificarea existenței unei chei

```
#create a dictionary
a = {'num': 1, 'denom': 2}
print(a)
#get a value for a key
print(a['num'])

#delete a key value pair
del a['num']
print(a)
```

```
#set a value for a key
a['num'] = 3
print(a)
print(a['num'])

#check for a key
if 'denom' in a:
    print('denom = ', a['denom'])
if 'num' in a:
    print('num = ', a['num'])
```

Elemente de bază Python

- Funcții în Python:
  - Elimină codul duplicat, facilitând refolosirea codului
  - Cod mai puțin și mai bine organizat
  - Fiecare funcție trebuie să fie responsabilă de un singur lucru
  - Vom începe fiecare funcție prin a-i scrie **specificațiile**:
    - O scurtă descriere a ce face funcția
    - Listarea datelor de intrare (parametrii), a tipurilor acestora și a rolurilor acestora
    - Listarea datelor de ieșire (returnate) sau a efectelor funcției
  - Vom scrie **teste cu aserțiuni** pentru fiecare funcție

Elemente de bază Python

- Specificațiile funcțiilor:
  - Specificațiile descriu **CE** face o funcție, **nu CUM** face acel lucru.
  - **NU** se fac referiri la variabile locale, la structuri de date, la cate for-uri se folosesc etc.

- Testarea funcțiilor:
  - Testele cu assert automatizează verificarea funcțiilor pe anumite date de intrare.
  - Nu orice funcție poate fi testată în acest mod. Vezi cursul 2 pentru mai multe detalii.

Elemente de bază Python

- Funcții în Python:

```
def f(x, y):
    '''
    Determina suma puterilor lui 2 din x si y.
    :param x: int, primul numar.
    :param y: int, al doilea numar.
    :return: la ce putere apare 2 in x
             plus la ce putere apare 2 in y
    '''
    num_2_x = 0
    num_2_y = 0
    while x % 2 == 0:
        num_2_x += 1
        x //= 2
    while y % 2 == 0:
        num_2_y += 1
        y //= 2
    return num_2_x + num_2_y

assert f(4, 2) == 3, 'f(4, 2) returned != 3'
assert f(5, 1) == 0, 'f(5, 1) returned != 0'
assert f(7, 6) == 1, `f(7, 6) returned != 1'
```

Elemente de bază Python

Q & A