(PTIA1201) Elemi programozás

Dr. Facskó Gábor, PhD
tudományos főmunkatárs
facskog@gamma.ttk.pte.hu

Pécsi Tudományegyetem, Természettudományi Kar, Matematikai és Informatikai Intézet, 7624 Pécs, Ifjúság utja 6.
Wigner Fizikai Kutatóközpont, Űrfizikai és Űrtechnikai Osztály, 1121 Budapest, Konkoly-Thege Miklós út 29-33.
`https://facsko.ttk.pte.hu`

2024. szeptember 13.

Kötelező és ajánlott irodalom

- Kötelező irodalom: TBD
- A kiadott tárgyismertető szerint
- Man page-ek, weblapok, fórumok, google, forráskódok, kollégák, oktatók

Követelmények

- Gyakorlati jegyet kapnak
  - Egy géptermi zh + pótzh $\rightarrow$ jegy
  - Időpontja: TBD
  - Pót zh időpontja: TBD
- Minden zh-mon mindent lehet használni, kivéve egymást. Nem a tárgyi tudást kérem számon
- Az előadásokat rögzítem, a diák elérhetőek lesznek a google drive-on
- A kódokat feltöltöm a GitHub-ra és a Moodle-ra
- Kérdezzenek bátran, órán, e-mailben, vagy fogadó órán: csütörtök 14:30-15:30 között, F/215-ös szoba

Python egyszerű adattípusok I

- Egyszerű adatípusok: int, float, string, logikai
- Konverziók
  - int(s): string $\rightarrow$ egész szám, pl. "123" $\rightarrow$ 123
  - float(x): string $\rightarrow$ valós szám, pl. "3.14" $\rightarrow$ 3.14
  - str(x): szám $\rightarrow$ string, pl. 123 $\rightarrow$ "123"
- Vezérlési szerkezetek?
  - Elágazás:
    if feltétel: utána kettőspont
        utasítás
    else: opcionális
        utasítás
  - While ciklus:
    while feltétel:
        utasítás

Python egyszerű adattípusok II

- For ciklus

```
For i in range(10):
    Print(i)
```

Python összetett adattípusok I

- Listák: pl.: szamok = [9.3, 7.5, 3.7, 0.1, 4.2]
- Az így létrehozott lista 5 elemű lesz. A sor elején álló, 0-s sorszámú elem a 9.3, a sor végén pedig a 4-es indexű elem áll, ez a 4.2.
- Lista hossza lekérdezhető a len() beépített függvénnyel: print(len(szamok)) # 5
- A listákat leggyakrabban ciklussal dolgozzuk fel. Ilyenkor figyelni kell arra, hogy a listaindexek tartománya 0-tól méret-1-ig-ig terjed. A lenti egy tipikus listás ciklus. Nullától indul az iterátor (ez a lista legelső eleme), és egyesével növekszik

```
i = 0
while i < len(szamok): # i = 0 ... i = 4
    print(szamok[i])
    i += 1
```

Python összetett adattípusok II

- Kiírásuk printtel:
  szamok = [1, 2, 3, 4, 5]
  print(szamok) # [1, 2, 3, 4, 5]
- Lista végéhez hozzá tudunk fűzni új elemet:
  szamok.append(6)
  print(szamok) # [1, 2, 3, 4, 5, 6]
- El tudjuk dobni a végén lévő elemet:
  szamok.pop()
  print(szamok) # [1, 2, 3, 4, 5]
- Rövidebben:
  for i in range(len(szamok)): # i = 0 . . . len(szamok)-1
      print(szamok[i])

Python összetett adattípusok III

- A legrövidebb megoldás:
  szamok = [9.3, 7.5, 3.7, 0.1, 4.2]
  for x in szamok: # x = 9.3, x = 7.5, ... print(x)

Utasítások, vezérlési szerkezetek I

- Skip/üres utasítás
  pass
- Értékadás
  a,b = b,a # megcseréli a-t és b-t
  c = []; d = [] # c és d két különböző üres lista
  c = d = [] # c és d ugyanazt az objektumot jelöli
- Szekvencia
- Elágazás
  if <kif> : <suite>
  (elif <kif> : <suite>)*
  [else : <suite>]
  Nincsen többszörös elágazás.
- Ciklus (for, while)

Utasítások, vezérlési szerkezetek II

- While:
  while <kif> : <suite>
  [else : <suite>]
- A while a hagyományos elöltesztelős ciklus. Az else-ága akkor fut le, ha a ciklusfeltétel nem teljesül (tehát a ciklus elhagyása után mindig, kivéve ha break utasítással hagyjuk el a ciklust).
- Kiugrási lehetőségek a ciklusból
  break - rögtön a ciklus utáni utasításra kerül a vezérlés
  continue - a ciklusfeltétel tesztelésére ugrik a vezérlés
- Nincs GOTO utasítás!
- Nincs lehetőség egyszerre több ciklusból való kiugrásra.

- For:
  for <target_list> in <kif_list> : <suite> [else : <suite>]

Utasítások, vezérlési szerkezetek III

- A <kif_list> kiértékelésekor egy sorozatot kell kapni. Ennek minden elemét hozzárendeli a <target_list>-hez és végrehajtja a <suite> részt. Ha a sorozat kiürült, akkor lefut az esetleges else ág.
- Ha a for ciklust a hagyományos értelemben szeretnénk használni, akkor jön jól a range() függvény:
  >>>range(10)
  [0,1,2,3,4,5,6,7,8,9]
  >>>range(5,10)
  [5,6,7,8,9]
  >>>range(0,10,3)
  [0,3,6,9] >>>range(-1,-10,-3) [-1,-4,-7]

Utasítások, vezérlési szerkezetek IV

- A magon belül módosíthatjuk a sorozatot, de ez csúnya hibákhoz vezethez, ugyanis egy számláló mutatja a következő feldolgozandó elemet, és egy beszúrás vagy törlés ebből a listából megkavarhatja a folyamatot. Megoldás: ideiglenes másolat készítése (pl. slice használata)
  for x in a[:]: if x < 0 : a.remove(x)
- A 2.0-es verzióban bevezették egy lista alapján egy másik lista kiszámítására a [for in] jelölést. Pl.:
  print [i**2 for i in range(4)] # [0, 1, 4, 9]
- A 2.4-es Pythonban új beépített függvény a reversed(seq), ami egy sorozatot vár, és visszaad egy iterátort, ami a sorozat elemein megy végig fordított sorrendben. pl:
  >>> for i in reversed(xrange(1,4)): print i

String kezelés I

- A String a Python típusrendszerében a Sorozat (Sequence) típusai közé tartozik, és fajtáját tekintve immutábilis (immutable). String literálokat a " vagy ' idézőjelek közé teszünk: "hello", 'hello' (az idézőjel választása tetszőleges). Literálokat írhatunk még tripla idézőjelek (""" vagy ''') közé, ekkor a literál tartalmazhat újsor karaktereket és idézőjeleket is.
  >>> str = """
  >>> print str
- A \ karaktert használjuk, ha escape-elni akarunk olyan karaktereket, melyeknek egyébként speciális jelentése van, mint például az újsor karakter, a \ karakter maga, vagy az idézőjelek.

String kezelés II

- A forráskódban megengedett több egymás utáni string literál, whitespace-el elválasztva (pl.: "hello" 'world'). Ez ekvivalens a két literál konkatenációjával ("helloworld"). Fontos tudni, hogy az e fajta összefűzés fordítási időben történik. Futási időben ez a konkatenáció nem működik, helyette a + operátor használandó.
- Mivel a Stringek immutábilis sorozatok, ezért az ezekre értelmezett műveletek természetesen használhatók a karakterláncokra is.
  - x in str Igaz, ha x strning része az str string-nek (substring)
  - s + t s és t stringek konkatenációja
  - s * n, n * s s replikálása n-szer, egymáshoz konkatenálva
  - s[i] s string i-edik karaktere (0 az első)
  - s[i:j] s string i-től j-ig tartó szelete
  - s[i:j:k] s i-től j-ig tartó szelete, k lépésközzel
  - len(s) s hossza
  - min(s) s-ben a legkisebb ASCII kódú karakter
  - max(s) s-ben a legnagyobb ASCII kódú karakter

Vége

Köszönöm a figyelmüket!