1. Gyakorló feladatsor

Python mérnöki alkalmazásai (*Monori Bence*)

1 Bevezető feladatok

1.1 "Hello world" - Mechatronikai rendszerek

Marika néni az utóbbi időben nem érzi magát biztonságban a pesterzsébeti (XX. kerületi) lakásában. Attól tart, hogy illetéktelenek bejuthatnak az ingatlanjába, ezért megkéri kedves unokáját, *Petikét*, hogy fejlesszen ki egy arcfelismerő rendszert, amely jelzi, ha idegenek próbálnak bejutni a lakásába. Ehhez Petike rendel egy IP kamerát, melynek jelét feldolgozva közvetlen értesítést tud küldeni Marika néni telefonjára, ha gyanús alak van a ház előtt.

1. Nevezzük meg az arcfelismerő rendszerhez kapcsolódó *szenzor-* és *aktuátoroldali* rendszereket!
2. Az előzőek alapján nevezzük meg az arcfelismerő rendszer *bemenetét* és *kimenetét*!
3. Adjunk meg néhány lehetőséget, melyeken keresztül a rendszerek *kommunikálni tudnak* egymással!

1.2 Változók és aritmetikai műveletek - Számtani és mértani közép

Hozzunk létre két (`a` és `b`) változót, melyekben két egész számot tárolunk el. Határozzuk meg ezen két szám *számtani* és *mértani* átalát! Ellenőrizzük a számtani és mértani közepek közti egyenlőtlenséget!

1.3 Változók és aritmetikai műveletek - Azonosságok

Hozzunk létre két (`x` és `y`) változót, melyekben két valós számot tárolunk el. (A két valós szám két szög nagyságát reprezentálja.) Ellenőrizzük az alábbi trigonometrikus és logaritmikus azonosságokat:

$$\sin^2(x) + \cos^2(x) = 1$$
$$\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y)$$
$$\ln(x^y) = y \cdot \ln(x)$$

1.4 Tömbök - Maximumkeresés

Hozzunk létre egy tetszőleges `homerseklet[]` tömböt, melyben tároljunk el (legalább 2) `float` típusú számot. Keressük meg és írjuk ki a tömb maximumát és azt, hogy hányas indexű ez az elem!

1.5 Tömbök - Gyümölcsök

Hozzunk létre két (`gyumolcs_1` és `gyumolcs_2`) változót, melyekben két *string* adatot tárolunk el. (Például `gyumolcs_1 = "alma"` és `gyumolcs_2 = "korte"`.) Fűzzük össze a két stringet úgy, hogy a `gyumolcs_1` változó minden karaktere nagybetűvel szedett, míg a `gyumolcs_2` változó tartalmát megfordítjuk! (Azaz az eredmény itt `"ALMAetrok"`.)

2 Gyakorló feladatok

Útmutató: a szövegben `ezen betűtípussal szedett szavak` olyan változókat jelölnek, melyeket mindig a felhasználó (azaz Te) definiál(sz)! Ezért javaslom, hogy mindig ezen változók megadásával kezdjétek a feladatokat!

2.1 Gyorsulások

Adott a térben egy `m` tömegű, pontszerűnek tekinthető test, amelyre adott `F_1`, `F_2` és `F_3` erők hatnak. Számítsuk ki az egyes a_x, a_y és a_z gyorsuláskomponenseket! Adott `dt` idő alatt mennyivel változik meg a test sebessége?

2.2 Relativisztikus gyorsulások

Ismeretes, hogy a *fénysebességgel összemérhető* sebességeken a Newtoni mechanika axiómái elérik a határaikat, és figyelembe kell vennünk a relativisztikus hatásokat. Az egyik fontos relativisztikus hatás az úgynevezett *relativisztikus tömegváltozás*, amely azt mondja ki, hogy ha a test nyugalmi tömege `m_0`, sebessége pedig a megfigyelőhöz képest `v`, akkor a megfigyelő rendszerében a test tömege:

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}}$$

Számoljuk ki a fénysebesség 10%, 50%, 90% és 99%-nál a relativisztikus tömegeket! Az egyes sebességeken hány százalékkal tér el a relativisztikus tömeg a nyugalmi tömegtől? *Gondolkodjunk el azon, hogy mekkora eltérés esetén érezzük úgy, hogy még eltekinthetünk a relativisztikus hatásoktól?*

2.3 Főtéri felújítások

Kübekcsakra főterének alapja egy szabályos `m`×`n`-es téglalap. A kübekcsakrai polgármesteri hivatal egy EU-s pályázat keretén belül fix összegű támogatást nyert el a főtér felújítására, melynek során fix `a`×`a` méretű gránitlapokkal fogják lefedni a teret. A lapokat se elvágni, se egymásra helyezni nem lehet, illetve a tér oldalaival párhuzamosan kell elhelyezni az összeset. A teljes teret le kell fedni, viszont nem probléma, ha annál nagyobb területet fednek le a lapok. Összesen hány lapra van szükség a felújításhoz?

2.4 Web Scraping

A stringkezelés egyik nagy népszerűségnek örvendő alkalmazása az úgynevezett *Web Scraping*. Ennek célja, hogy a scraping bot minél több adatot nyerjen ki, miközben véletlenszerűen járja a honlapokat.

Adott az alábbi (fiktív) URL: `URL="https:##idokep.hu#archive#2012#august"`. Ez a hivatkozás sajnos érvénytelen, ezért az a feladatunk, hogy kijavítsuk:

1. Cseréljük le a `#` karaktereket `/`-re, hogy a link formátuma érvényes legyen!
2. A scraping bot célja, hogy minden éven és hónapon végigmenjen! Írjuk át a linket úgy, hogy a `2023 september` fülre menjen!
3. A könnyebb módosításért keressük meg, hogy a linkben hanyadik indexű elemtől kezdődik az évszám.

3 Ajánlott feladatok

3.1 Tornyok

Petike nagyon szeret tornyokat építeni, ezért egy `N` darabból álló készletet kapott mamájától. Az építőelemek magassága bármilyen pozitív egész szám lehet, azaz `1, 2, 3, ...` és minden méretből tetszőlegesen sok lehet neki. Ezeket tároljuk el a `epitoelemek[]` tömbben. Pistike meg szeretne építeni *minden* olyan tornyot úgy, hogy az *azonos méretű* elemeket pakolja egymásra.

1. Határozzuk meg összesen hány tornyot tud építeni Pistike! *Segítség: Ismerünk-e olyan adatstruktúrát, amely nem tartalmazhat duplikált elemeket?*
2. Határozzuk meg melyik toronyban van a legtöbb elem!

Példa 1

Bemenet:
`epitoelemek = [1, 2, 3]`
Kimenet:
`"Tornyok szama:  3, legtobb elemu torony:  1"`

Példa 2

Bemenet:
`epitoelemek = [1, 4, 4, 4, 5, 7, 7]`
Kimenet:
`"Tornyok szama:  4, legtobb elemu torony:  3"`

Az egyik fontos megjegyzés, hogy az egyetlen elemből álló tornyokt is toronynak tekintjük. A másik fontos észrevétel, hogy nem az a kérdés, hogy fizikailag milyen magas a legnagyobb torony (ezért nem $7 + 7 = 14$ a válasz)! Azt kell meagdnunk, hogy hány elemű a legtöbb elemből álló torony: `[4, 4, 4]:  3`

3.2 Túl hosszú szavak...

A mérnökök köztudottan szeretnek *energiahatékonyak* lenni és jelentős fájdalmat okoz számukra, ha *gyakran le kell írniuk túl hosszú szavakat és kifejezéseket*, mint például a `"zsugorkötés"`, vagy a `"centrifugál regulátor"`. Ezért bevezetik a következő rövidítést: a *legalább 10 karakter* hosszú kifejezéseket átírják úgy, hogy csak az első és utolsó karaktereket tartjuk meg és a köztes karaktereket helyettesítjük a karakterek számával:

- `"cica"` → `"cica"` (Mert csak 4 karakter hosszú.)
- `"centrifugal regulator"` → `"c19r"` (Mert 19 karakter van a `c` és az `r` között.)

Írjunk egy programot, mely szükség esetén rövidíti a kifejezéseket!

3.3 Jegyzettömb+-

Petike nemrég készített egy `Jegyzettömb+-` alkalmazást, amelyben a gyors és hatékony gépelésre törekedett. Az alkalmazás két különböző funkciót támogat:

1. A szavakat begépelhetjük "hagyományosan" azaz *egy adott lépésben egy betűt*;
2. De lehetőségünk van egy adott lépésben a *szó végére másolni a korábban begépelt szöveg egy tetszőlegesen hosszú, folytonos substringjét.*

Tegyük fel, hogy a begépelni kívánt `s` string csupán `n` darab, kisbetűs latin karakterből áll. Írjunk egy programot, mely eldönti, hogy ezen `s` string begépelhető-e szigorúan kevesebb, mint `n` lépés alatt!

Példa 1

Bemenet:
`n = 10; s = "centrifuga"`
Kimenet: `"NEM"`
Itt az egyetlen lehetőségünk, hogy betűnként gépeljük be a szót, így `n` lépésre van szükség.

Példa 2

Bemenet:
`n = 6; s = "banana"`
Kimenet: `"IGEN"`
Itt megtehetjük, hogy begépeljük a `"ban"` részt, másoljuk az `"an"` substringet és végül leírjuk az `"a"`-t, ami összesen $3 + 1 + 1 = 5 < 6$ lépés.

Megoldások

Az alábbiakban találhatóak megoldások a *bevezető* és a *gyakorló* példákhoz. Kérlek szánjatok kellő időt és figyelmet a megoldások tanulmányozására, próbáljátok ki őket, futtassátok le, kísérletezzetek! Illetve hasonlítsátok össze a saját elgondolásaitokkal, gondoljátok végig melyik miért lehet célravezetőbb!

1.1 Mechatornikai rendszerek

1. A szenzoroldalon az *IP kamera* található, míg az aktuátoroldalon *Marika néni telefonja* (illetve arra telepített alkalmazás) áll.

2. Az arcfelismerő rendszer bemenete egy adott *felbontás*ú és *képfrissítési frekvenciá*jú *videófelvétel*[1], míg a rendszer kimenete az, hogy éppen van gyanús alak a láthatáron `True`, vagy nincs `False`.

3. Ez több különböző csatornán történhet: legegyszerűbben *Ethernet* vagy *WiFi*, esetleg Bluetooth segítségével.

1.2 Számtani és mértani közép

Két szám A számtani és G mértani átlaga definiálható, mint:

$$A = \frac{a+b}{2} \qquad G = \sqrt{a \cdot b} = (a \cdot b)^{\frac{1}{2}}$$

Ezek alapján az implementáció:

```
# Definiáljuk a változókat
a = 2
b = 5

# Kiszámítjuk a számtani és mértani összegeket
arithmeticMean = (a+b)/2
geometricMean = (a*b)**(1/2)
print(f"A {a} és {b} számtani közepe: {arithmeticMean}")
print(f"A {a} és {b} mértani közepe: {geometricMean}")
```

Kicsit tömörebben:

```
a, b = 2, 5
aMean, gMean = (a+b)/2, (a*b)**(1/2)
print(f"Az {a} és {b} számtani és mértani átalaga: {aMean} és {gMean}")
```

[1] Az érkező kép felbontása és frekvenciája is fontos bemenet! Későbbiek során fogjuk látni, hogy egy Full HD felbontású, 10 másodperc hosszú, 30 FPS-es videót eltárolhatunk egy `[1920x1080x300]`-as tömbben!

1.3 Azonosságok

A feladatban szerepel 3 jól ismert azonosság. Ezeket könnyen ellenőrizhetjük is a `numpy` könyvtár segítségével!

```
import numpy as np      # Bővített matematikai eszköztár

# Definiáljuk a változóinkat
x = np.pi/2
y = np.pi/3

# Számítsuk ki az egyes kifejezések értékeit!
idLeft1 = np.sin(x)**2+np.cos(x)**2
idRight1 = 1.0
idLeft2 = np.cos(x+y)
idRight2 = np.cos(x)*np.cos(y)-np.sin(x)*np.sin(y)
idLeft3 = np.log(np.power(x,y))
idRight3 = y*np.log(x)

# Kiíratás
print(f"1. azonosság: {idLeft1} = {idRight1}")
print(f"2. azonosság: {idLeft2} = {idRight2}")
print(f"3. azonosság: {idLeft3} = {idRight3}")
```

1.4 Maximumkeresés

```
# Definiáljuk a változónkat!
homerseklet = [20.5, 22.1, 23.7, 23.0, 22.7]

# Maximumkeresés
homMax = max(homerseklet)
homMaxIndex = homerseklet.index(homMax)

# Kiíratás
print(f"A legmagasabb érték: {homMax}, ami a {homMaxIndex} indexű elem.")
```

1.5 Gyümölcsök

```
# Definiáljuk a változóinkat!
gyumolcs_1 = "alma"
gyumolcs_2 = "korte"

# Fűzzük össze őket a feladatnak megfelelően
gyumolcsok = gyumolcs_1.upper() + gyumolcs_2[::-1]

print(f"A kapott string {gyumolcsok}")
```