

Telekommunikációs Hálózatok

1. gyakorlat

Elérhetőségek

- gyakorlatvezető: Szalai-Gindl János
- honlap: <http://szalaigj.web.elte.hu/>
- email: szalaigindl@inf.elte.hu
- szoba: 2.507 (déli tömb)

Követelmények

- Maximum 4 hiányzás
- Jegy összetétele, minden rész $\frac{1}{4}$ súllyal:
 - „számolós” ZH (**!!! 50% elérés szükséges !!!**)
 - Egyszerűbb számolós feladatok
 - Socket ZH (**!!! 50% elérés szükséges !!!**)
 - Python
 - Mininet nagyprojekt
 - Routing, tűzfal, IP cím beállítások
 - Python, socket házi feladatok (kb. 5 db.)
 - BEAD rendszeren tesztelve

Házi feladatok

- Házi feladat:
 - Általában valamilyen programozási feladat
 - Általában 2-3 hét lesz a megoldásra
 - BE-AD rendszert fogjuk használni egy automata tesztkörnyezettel együtt, ami értékeli a beadott programokat és figyeli a kódhasonlóságokat.
 - **!!! Az eredményt megjegyzésbe rakja, időnként fut le a tesztelő !!!**
 - A házi feladatokat Python3-ban kell megírni, mert a BE-AD rendszeren keresztül elérhető tesztkörnyezet is abban van!
 - **Másolt beadandó leadása csalásnak minősül, az egyetemi szabályoknak megfelelően járunk el ilyen esetekben!**

Ponthatárok

$$\begin{aligned} \text{Jegy} = & \\ & (\text{szerzettHF} / \text{maxHF}) * 25 \\ & + \\ & (\text{szerzettMininet} / \text{maxMininet}) * 25 \\ & + \\ & (\text{szerzettZH1} / \text{maxZH1}) * 25 \\ & + \\ & (\text{szerzettZH2} / \text{maxZH2}) * 25 \end{aligned}$$

Százalék	Érdemjegy
0 - 49 %	Elégtelen (1)
50 - 59 %	Elégséges (2)
60 - 74 %	Közepes (3)
75 - 84 %	Jó (4)
85 – 100 %	Jeles (5)

A ZH-kon el kell érni 50%-ot a kurzus teljesítéséhez!!!

Python történelem és tulajdonságok

- Guido Van Rossum holland programozó készítette a 90-es évek elején
 - nevét a Monty Python Repülő Cirkusza után kapta
- Python tulajdonságai:
 - Könnyű tanulásra lett tervezve
 - Tiszta, egyszerű szintaxis, kevés kulcsszó
 - Hordozható
 - Majdnem minden elfut
 - Szóközöket használ program blokkok elkülönítéséhez
 - Egy jó programozó amúgy is használná, akkor a nyelv miért ne?
 - A változókat nem szükséges deklarálni
 - Ettől még nem típus-független nyelv!

Python parancssor

```
#python  
python> import this  
python> print("Hello world!")  
python> user_name="Jozsi"  
python> print("Hello " + user_name)  
python> user_age=25  
python> print("You are " + str(user_age) + " years old.")
```

megj: pythonnal mindegy hogy ' -t vagy " -t
használsz

Egyszerű számítások

```
Python>10+2
```

```
12
```

```
Python>2*2
```

```
4
```

```
Python>3**2
```

```
9
```

```
Python>10%2
```

```
0
```


Változók

```
Python> a = 42
Python> b = 32
Python> c = a + b
Python> print(c)
74
Python> c = 'valami'
Python> print(a+c)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

String műveletek

```
Python>print('alma'.upper())
```

```
ALMA
```

```
Python>print( "LO" in "Hello".upper() )
```

```
True
```

```
Python>print("Decimal Number: %d, Float: %f, String: %s" %  
(12,33.4,"almafa"))
```

```
Decimal Number: 12, Float: 33.400000, String: almafa
```

```
Python>"Decimal Number: {1:d}, Float: {1:f}, String:  
{2:s}".format(12,33.4,"almafa")
```

Listák

```
Python> players = [12,31,27,'48',54]
Python> print(players)
[12, 31, 27, '48', 54]
Python> players[0]
12
Python> players[-1]
54
Python> players + [22, 67]
[12, 31, 27, '48', 54, 22, 67]
Python> print(len(players))
5
```

Listák

```
Python> players = [12,31,27,'48',54]
Python> players.append(89)
Python> print(len(players))
6
Python> players[2] = 'alma'
Python> players[2:]
['alma', '48', 54]
Python> del players[2]
Python> players
[12, 31, '48', 54]
```

Tuple – nem módosítható lista

```
Python> players = (12,31,27,'48',54)
Python> players[2] = 'alma'
TypeError: 'tuple' object does not support item assignment
Python> print(players[2])
27
Python> del players[2]
TypeError: 'tuple' object doesn't support item deletion
Python> print(players[1:-2])
(31,27)
```

Halmazok

```
Python> mylist = [8,3,2,3,2,4,6,8,2]
Python> myset = set(mylist)
Python> print(mylist)
[8, 3, 2, 3, 2, 4, 6, 8, 2]
Python> myset = {8,3,2,3,2,4,6,8,2} # alternative
Python> print(myset)
{2, 3, 4, 6, 8}
Python> type(myset)
<type 'set'>
Python> mysortedlist = sorted(mylist)
Python> print(mysortedlist)
[2, 2, 2, 3, 3, 4, 6, 8, 8]
```

Szótár

```
Python> team = {  
    91: "Ayers, Robert",  
    13: "Beckham Jr,",  
    3: "Brown, Josh",  
    54: "Casillas, Jonathan",  
    21: "Collins, Landon"}  
Python> len(team)  
5  
Python> team[3] = "Chihiro"  
Python2> print(team.has_key(91))  
True  
Python3> print(91 in team)  
True  
Python2> print(team.has_key('alma'))  
False  
Python3> print('alma' in team)  
False
```

Szótár

```
Python> team = {  
    91: "Ayers, Robert",  
    13: "Beckham Jr,",  
    3: "Brown, Josh",  
    54: "Casillas, Jonathan",  
    21: "Collins, Landon"}  
Python3> print(team.keys())  
dict_keys([91, 13, 3, 54, 21])  
Python3> print(team.values())  
dict_values(['Ayers, Robert', 'Beckham Jr,', 'Brown, Josh',  
'Casillas, Jonathan', 'Collins, Landon'])
```


Elágazások

```
if 100 in team:  
    print('Yes, 100 is in the team')  
elif 76 in team:  
    print('100 is not in the team, but 76 is in it...')  
else:  
    print('Both 100 and 76 are not in the team')
```

Ciklus

```
mylist = [3,65,2,77,9,33]
```

```
for i in mylist:  
    print('Element:', i)
```

Írassuk ki a lista elemeit növekvő sorrendben!

```
for i in range(2,10,2): #2-től 9-ig 2-esével  
    print(i)
```

Ciklus

```
for i in range(5):  
    print("Number" + str(i))  
  
for (k,v) in team.items():  
    print("Player name: {:s}; #: {:d}".format(v,k))
```

```
Player name: Brown, Josh; #: 3  
Player name: Nassib, Ryan; #: 12  
...
```

```
i=1  
while i<10:  
    print(i)  
    i+=1
```

Python script futtatása

```
#vi test.py

#!/usr/envbin/python
x = 1
for i in range(1,5):
    x+=i
    print(x)
    print(str(i) + " alma")

#python test.py
#chmod +x test.py && ./test.py
```

Függvények

```
#!/usr/envbin/python

def is_even(num):
    if (num % 2) == 0:
        return True
    else:
        return False

for i in range(1,10):
    if (is_even(i)):
        print("Szam:"+str(i))

print("Vege")
```

Függvények

```
def complex(x):  
    return x**2, x**3, x**4
```

```
print( complex(2) )  
# (4,8,16)
```

```
a, b, c = complex(2)  
print(a,b,c)  
# 4 8 16
```

```
_, rv, _ = complex(2)  
print( rv )  
# 8
```

Lambda Függvények

```
is_even = lambda num: (num % 2) == 0

is_even_2 = lambda num: True if (num % 2) == 0 else False

for i in range(1,10):
    if (is_even(i)):
        print("Szam:"+str(i))

print("Vege")
```

Lista, Dict, Tuple generálás

```
mylist = [ x*x for x in range(10) ]  
# [0,1,4,9,16,25,36,64,81]  
  
mydict = { x:x*x for x in range(5) }  
# {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}  
  
mydict2 = { x:x*x for x in range(5) if x!=2 }  
# {0: 0, 1: 1, 3: 9, 4: 16}  
  
mytuple = tuple( x*x for x in range(3) )  
# (0, 1, 4)
```


map

```
def fahrenheit(T):  
    return ((float(9)/5)*T + 32)  
  
def celsius(T):  
    return (float(5)/9)*(T-32)  
  
temperatures = (36.5, 37, 37.5, 38, 39)  
F = map(fahrenheit, temperatures)  
C = map(celsius, F)  
  
temperatures_in_Fahrenheit = list(map(fahrenheit, temperatures))  
temperatures_in_Celsius = list(map(celsius, temperatures_in_Fahrenheit))  
  
print(temperatures_in_Fahrenheit)  
# [97.7, 98.60000000000001, 99.5, 100.4, 102.2]  
print(temperatures_in_Celsius)  
#[36.5, 37.000000000000001, 37.5, 38.000000000000001, 39.0]
```

filter

```
fibonacci = [0,1,1,2,3,5,8,13,21,34,55]

odd_numbers = list(filter(lambda x: x % 2, fibonacci))

print(odd_numbers)
# [1, 1, 3, 5, 13, 21, 55]
```

Fájl kezelés és string szétválasztás

```
with open('alma.txt', 'r') as f:
    for line in f:
        print( line.rstrip('\n').split(',') )
```

```
f = open("demofile.txt", "r")

print(f.read())

print(f.readline())

for x in f:
    print(x)

f.close()
```

```
with open('alma.txt', 'w') as f: # w-write, a-append
    f.write("Bla Bla")
```

Standard inputról olvasás

```
x = input("Kell egy szám:")  
  
# x típusa mindig str !!!  
  
print("Kapott szám",x)
```

Parancssori paraméterek

```
import sys

print(sys.argv[0])    #← a script neve

print(sys.argv[1])    #← első paraméter
print(sys.argv[2])    #← második paraméter
...
```

Osztályok

```
class Hallgato:
    nev = ''
    ZHpont = 0

    def __init__(self, _name, _point):
        self.nev = _name
        self.ZHpont = _point

    def __str__(self):
        return self.nev+"(" +str(self.ZHpont)+")"

p = Hallgato("Ford",20)
```

Import vs main()

- `gyak2proba.py`

```
def main():  
    print("Ez a main")  
  
if __name__ == "__main__":  
    print ("Ez fog lefutni ha scriptkent hivod meg!")  
    main()
```

- `gyak2import.py`

```
import gyak2proba  
  
gyak2proba.main()
```

vagy

```
from gyak2proba import main  
  
main()
```

```
$ python3 gyak2proba.py  
Ez fog lefutni ha scriptkent hivod meg!  
Ez a main  
$ python3 gyak2import.py  
Ez a main
```

JSON - JavaScript Object Notation

Segédlet: <https://realpython.com/python-json/>

```
{
  "firstName": "Jane",
  "lastName": "Doe",
  "hobbies": ["running", "sky diving", "singing"],
  "age": 35,
  "children": [
    {
      "firstName": "Alice",
      "age": 6
    },
    {
      "firstName": "Bob",
      "age": 8
    }
  ]
}
```


JSON & Python – **import json**

JSON objektum mentése JSON fájlba

```
import json


data = {
    "president": {
        "name": "Zaphod Beeblebrox",
        "species": "Betelgeusian"
    }
}

with open("data_file.json", "w") as write_file:
    json.dump(data, write_file)
```

JSON string előállítás JSON objektumból


```
json_string = json.dumps(data)
```

JSON & Python – Típus megfeleltetés szerIALIZÁCIÓ során



Python	JSON
dict	object
list, tuple	array
str	string
int, long, float	number
True	true
False	false
None	null

JSON & Python – Típus megfeleltetés deszerializáció során



JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

JSON & Python – JSON fájlok

JSON objektum beolvasása JSON fájlból

```
import json

with open("data_file.json", "r") as read_file:
    data = json.load(read_file)
    print( data["president"]["name"] )
```

JSON & Python – JSON fájlok

```
import json
json_string = """
{
    "researcher": {
        "name": "Ford Prefect",
        "species": "Betelgeusian",
        "relatives": [
            {
                "name": "Zaphod Beeblebrox",
                "species": "Betelgeusian"
            }
        ]
    }
}
"""
data = json.loads(json_string)

for rel in data["researcher"]["relatives"]:
    print('Name: %s (%s)' % ( rel["name"], rel["species"] ) )
```

Feladat 1.

Írjunk függvényt ami megadja egy bemenetben kapott évszámról, hogy szökőév-e.

Egy év szökőév, ha osztható néggyel, de akkor nem, ha osztható százszal, hacsak nem osztható négyszázszal.

Példák: 1992,1996,2000,2400 szökőév, de 1993, 1900 nem.

Feladat 2.

Írjunk függvényt ami megadja az n. fibonacci számot

$\text{fibonacci}(0) \rightarrow 0$

$\text{fibonacci}(1) \rightarrow 1$

$\text{fibonacci}(2) \rightarrow 1$

$\text{fibonacci}(3) \rightarrow 2$

...

$\text{fibonacci}(n) \rightarrow \text{fibonacci}(n-2) + \text{fibonacci}(n-1)$

Feladat 3.

Írjunk szkriptet, ami kiszámolja, hogy hány pont szükséges a második ZH-n a különböző érdemjegyekhez!

A bement egy json-t tartalmazó fájl legyen, amely tartalmazza a mininet, a házik és az első ZH-n hallgató által elért és a maximális pontot. A kimenet pedig az egyes érdemjegyekhez szükséges minimális ZH pontszám.

A szkript továbbá figyelje azt is, hogy a második ZH-n megvan-e a minimális pontszám! Ha nem lenne meg, akkor „Nincs meg a minimum pont” üzenetet írjon az érdemjegyhez!

```
{  
  "haziPont": {"max":20,"elert":10},  
  "1zhPont": {"max":20,"elert":10,"min":10},  
  "mininetPont": {"max":20,"elert":20},  
  "2zhPont": {"max":20,"min":10}  
}
```

```
python zhSzamolo.py  
2 : Nincs meg a minimum pont  
3 : 10.0  
4 : 20.0  
5 : Remenytelen
```


VÉGE
KÖSZÖNÖM A FIGYELMET!