

# Számítógépes Hálózatok

## 1. gyakorlat

# Elérhetőségek

- Gyak.vez: Pusztai Gábor
- fájlok: <http://wornox.web.elte.hu/>
- email: [afbdpk@inf.elte.hu](mailto:afbdpk@inf.elte.hu)

# Követelmények

- Maximum 4 hiányzás
- Számokérések (minden rész 1/3 súllyal):
  - Python, socket beadandók (kb. 4)
    - TMS rendszeren tesztelve ([tms.inf.elte.hu](http://tms.inf.elte.hu))
  - Python, socket ZH (!!! 50% elérés szükséges !!!)
    - Élesben feladat megoldás
  - Mininet nagybeadandó
    - Routing, tűzfal, IP cím beállítások

# Beadandók

## Leírás:

- Programozási, szimulációs feladat
- Általában 2-3 hét a beadás
- TMS rendszeren kell leadni, ami értékelni fogja és figyel a kódhasonlóságot
  - **!!! Az eredményt megjegyzésbe rakja, időnként fut le a tesztelő, néha kell pár percet várni az eredményre !!!**
- Másolt kód leadása csalásnak minősül és az egyetemi szabályoknak megfelelően járunk el.
- Nem kötelezőek, de nagyban beleszámítanak a jegybe

# Ponthatárok

Gyakorlati jegy = beadandók \* 33.3% +  
ZH \* 33.3% +  
mininet \* 33.3%

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

# Tervezett tanmenet

Tanmenet 2022-2023-1			
Hét	Óra	Dátum	Témakör
1	1	2022-09-13	Python alapok
2	2	2022-09-20	Subprocessek
3	3	2022-09-27	Hálózati forgalom, Kommunikációs csatorna
4	4	2022-10-04	Többklienses kapcsolat
5	5	2022-10-11	Redundancia, Kódolás
6	6	2022-10-18	Proxy, UDP
7	7	2022-10-25	NAT, Portforwarding, Multicast
8	-	2022-11-01	Mindenszentek (szünet)
9	ZH	2022-11-08	ZH
10	8	2022-11-15	Mininet
11	9	2022-11-22	Mininet II.
12	ZH	2022-11-29	PótZH
13	10	2022-12-06	Mininet III. (Online)
14	-	2022-12-13	Őszi (szünet)

# Python történelem

- Guido Van Rossum, 1989 karácsonya
- Van Rossum írta '96-ban:

„Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of [ABC](#) that would appeal to [Unix/C hackers](#). I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of [Monty Python's Flying Circus](#)).”



# Python tulajdonságok

- Könnyű tanulásra lett tervezve
  - Tiszta, egyszerű szintaxis, kevés, rövid kulcsszó
- Hordozható
  - Majdnem minden elfut (linux, windows, RasbPi, Big Data)
- 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!
- Verziók
  - python 2.x, 3.x (elérés: python, py)
  - DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020.
  - Python 2-őt már csak a dependenciák miatt használunk max!



# 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, illetve van ''' is

# Egyszerű számítások

```
Python>10+2
```

```
12
```

```
Python>2*2
```

```
4
```

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

```
9
```

```
Python>10%2
```

```
0
```

# Matematikai kerekítések

- Alapjáraton minden osztás tört osztás
- `int(2.9)` -> 2 (truncation), nem kerekítünk, csak eldobjuk a tört részt
- `math.ceil(1.1)` -> 2 : A számnál nagyobb vagy egyenlő egész számot adja vissza
- `math.floor(1.9)` -> 1: A számnál kisebb vagy egyenlő egész számot adja vissza
- `round(3.3)` -> 3, `round(3.5)` -> 4: A szokásos matematikai kerekítési szabályok szerint dolgozik

# Változók

```
Python> a = 42
Python> b = 32
Python> c = a + b
Python> print(c)
74
Python> c = 'valami'
Python> print(a+c)
ERROR
```

# 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
```

# 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:]
[27, 48, 54, 89]
```

# Tuple – nem módosítható lista

```
Python> players = (12,31,27,'48',54)
Python> players[2] = 'alma'
ERROR
Python> del players[2]
ERROR
Python> players[2:]
[27, 48, 54, 89]
```



# 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> print(myset)
set([8, 2, 3, 4, 6])
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"  
Python> print( 91 in team )  
True  
Python> print ( 'alma' in team )  
False
```

# Szótár

```
Python> team = {  
    91: "Ayers, Robert",  
    13: "Beckham Jr,",  
    3:  "Brown, Josh",  
    54: "Casillas, Jonathan",  
    21: "Collins, Landon"}
```

```
Python> print (team.keys())  
dict_keys([91,13,3,54,21])
```

```
Python> 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 számokat növekvő sorrendben kettesével!

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

# Ciklus

```
for (k,v) in team.items():  
    print "Player name: %s; #: %d" % (v,k)
```

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

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

# Python script futtatása

```
#vi test.py

#!/usr/bin/env/python
x = 1
for i in range(1,5):
    x+=i          //megj: nincs ++ oprátor
    print (x,i,'alma', 'x*x = %d' % (x*x))
    print(str(i) + " alma")

#python test.py vagy py test.py
```

# Függvények

```
#!/usr/bin/env 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

```
#!/usr/ /python

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)
```

# Példa: Hőmérséklet

```
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) #map(function, iterable data)  
C = map(celsius, F)  
  
temperatures_in_F = list(map(fahrenheit, temperatures))  
temperatures_in_C = list(map(celsius, temperatures_in_F))  
  
print(temperatures_in_F)  
# [97.7, 98.60000000000001, 99.5, 100.4, 102.2]  
  
print(temperatures_in_C)  
#[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))
                #filter(function, iterable data)

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

# Beolvasás I. :

## File műveletek

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

print(f.readline())

for x in f:
    print(x)

f.close()
```

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

```
f = open("demofile.txt", "w")          # w-write, a-append

f.write(„Bla Bla")
```

# Beolvasás II. :

## Standard inputról olvasás

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

# Beolvasás III. :

## Parancssori paraméterek

```
import sys

print (sys.argv[0]) #← a script neve

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



# Kódolási hibák

Néha hibaüzenet, ha ékezetes betűk vannak, akár a kommentben is!

```
SyntaxError: Non-ASCII character '\xc3' in file gyak2.py on line 44,  
but no encoding declared; see http://python.org/dev/peps/pep-0263/  
for details
```

Megoldás I. : A script első sorába, ezzel nincs hiba üzenet:

```
# coding: utf-8
```

Megoldás II. : Szöveg kiírása unicode-ként:

```
print u'áéúúóü'
```

Megoldás III. : Állítsuk át a fájl kódolását:

```
például Notepad++ Encoding menüopció
```

## Példa 2. : 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)+")"
    def __repr__(self):
        return "in list "+self.nev+"("+str(self.ZHpont)+")"

p = Hallgato("Ford",20)

print(p)      # __str__
# Ford(20)
print([p])    # __repr__
# [in list Ford(20)]
```

# Import vs main()

gyak2Proba.py

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

gyak2import.py

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

vagy

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

```
$ python3 gyak2proba.py  
Ez fog lefutni ha scriptkent hívod meg!  
Ez itt a main
```

```
$ python3 gyak2import.py  
Ez itt 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 serializáció 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 a paraméterben kapott évszámról, hogy szökőév-e.

Az évszámokat egy fájlból olvassuk be!

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

Példák:

- szökőév: 1992, 1996, 2000, 2400
- nem szökőév: 1993, 1900

## Feladat 2.

- írjunk scriptet, ami kiszámolja, hogy hány pont szükséges a zh-n az egyes jegyek eléréséhez. A bement egy json fájl legyen, amely tartalmazza a mininethez és a házikhoz az elért és maximális pontot, illetve a zh-hoz a maximális pontszámot és, hogy hány százalék teljesítése szükséges az érvényességhez. A kimenet az egyes érdemjegyekhez szükséges minimális pont. (Rész pont nincs!)

```
{  
  "haziPont": {"max": 20, "elert": 10 },  
  "zhPont": {"max": 20, "minimum": 0.5 },  
  "mininetPont": {"max": 20, "elert": 20 },  
}
```

```
python zhSzamolo.py  
2 : 10.0  
3 : 10.0  
4 : 20.0  
5 : Remenytelen
```

# Szorgalmi Feladat

Írjunk függvényt ami megadja az n. fibonacci számot

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

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

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

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

...

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

**VÉGE**