Telekommunikációs Hálózatok

1. gyakorlat

Gyakvez

- Kecskeméti Károly
- Weblap:kkaroly.web.elte.hu
- E-mail:cgsmef@inf.elte.hu
- Vagy Teamsen
- Szoba:4.730

Követelmények

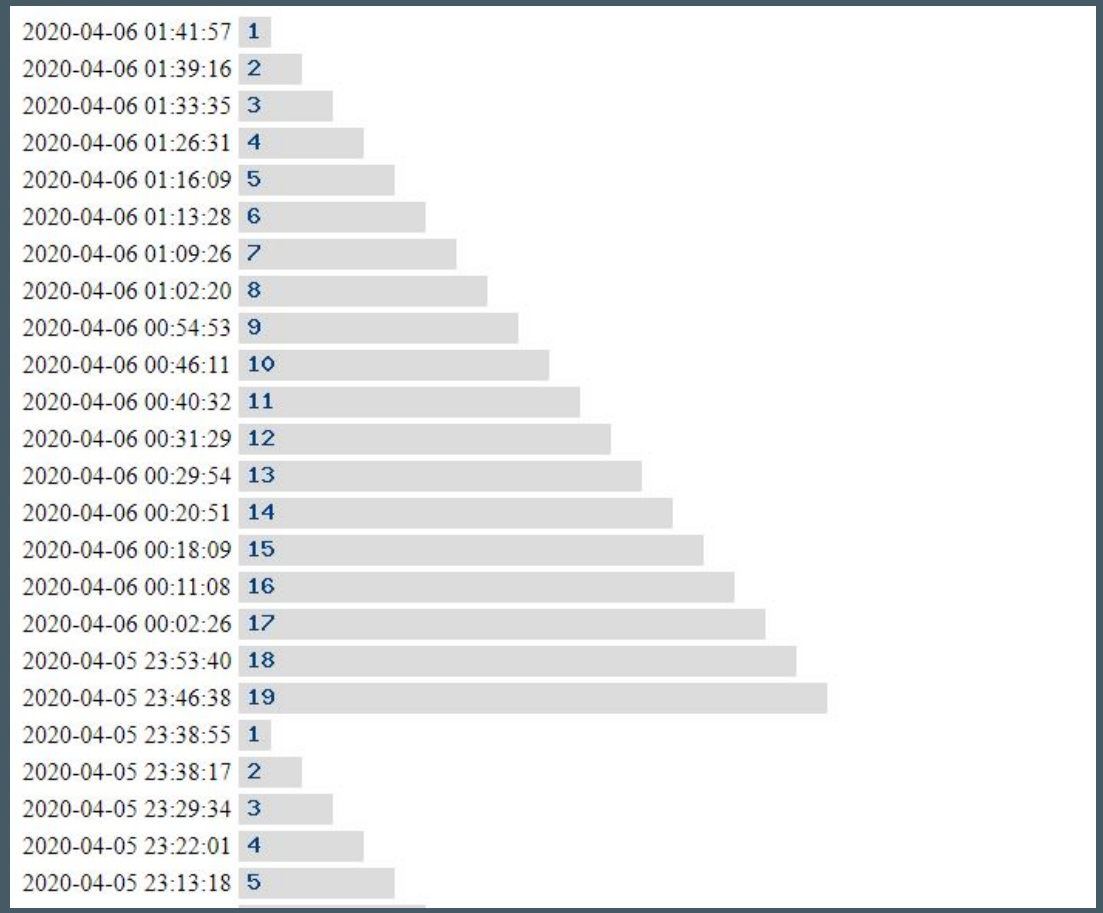
- Maximum 4 hiányzás
- Számonkérések:
  - Socket beadandók (4 db, min 50%)
  - Socket ZH (mint 50%)
  - Mininet beadandó (min 50%)

Socket beadandók

- Programozási, szimulációs feladat
- 3 hét a határidő
- TMS-en keresztül kell leadni, ami értékelni fogja és figyeli a kódhasonlóságot
- Az eredményt megjegyzésbe rakja, időnként fut le a tesztelő
- Másolt kód leadása csalásnak minősül és az egyetemi szabályoknak megfelelően járunk el

Deadline Panic

Ne halogass!



Az értékelő feldolgozási sora, határidő környékén

Ponthatárok

$$\% = \frac{beadando\%}{3} + \frac{mininet\%}{3} + \frac{ZH\%}{3}$$

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

Témakörök

- Python alapok
- Socket programozás
- CRC, kódolások, MD5
- Tűzfalak: Iptables
- Wireshark/tcpdump forgalom elemzés.
- Mininet, hálózati karakterisztikák, alapvető eszközök: traceroute, ping
- MAC learning, STP, ARP, routing beállítások
- Port forwarding, VLAN beállítások
- Tunneling megoldások IPv4/IPv6

Python történelem

Guido Van Rossum, ’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

- Könnyű tanulásra lett tervezve
  - Tiszta, egyszerű szintaxis, kevés, rövid kulcsszó
- Hordozható:
  - Majdnem mindenen elfut (linux, windows, mac, Raspberry Pi,)
- Whitespace-t használ a blokkok elkülönítéséhez
  - Az olvashatóság miatt amúgy is így kéne
- A változókat nem szükséges deklarálni
  - Ettől még nem típus-független nyelv
- Verziók
  - python 2.x, 3.x (laborgépeken: python, py)
  - python2 DEPRECATED

Python REPL(Read-Eval-Print Loop)

Indítás

```
$ python
$ py # laborgépeken
```

Mindegy, hogy ' -t vagy " -t használsz, csak azt konzisztensen

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

Egyszerű számítások

```
>>> 10+2
12
>>> 2*2
4
>>> 3**2
9
>>> 10%2
0
```

Matematikai kerekítések

```
>>> import math
>>> math.floor(3.8)
3
>>> round(3.8)
4
>>> round(3.8,1)
3.8
```

Változók

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

String műveletek

```
>>>print("alma".upper())
ALMA
>>>print("LO" in "Hello".upper())
True
>>>print("Decimal Number: %d, Float: %f, String: %s" % (12,33.4,"almafa"))
Decimal Number: 12, Float: 33.400000, String: almafa
>>>x = 42
>>>print(f"x == {x}")
x == 42
```

List

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

List

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

Tuple – immutable List

Nem módosítható

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

Set

```
>>> mylist = [8,3,2,3,2,4,6,8,2]
>>> myset = set(mylist)
>>> print(mylist)
[8, 3, 2, 3, 2, 4, 6, 8, 2]
>>> print(myset)
{2, 3, 4, 6, 8}
>>> mysortedlist = sorted(mylist)
>>> print(mysortedlist)
[2, 2, 2, 3, 3, 4, 6, 8, 8]
```

Dictionary

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

Dictionary

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

If

```
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")
```

For ciklus

```
 1 mylist = [3,65,2,77,9,33]
 2 for i in mylist:
 3     print("Element:", i)
 4
 5 # Írassuk ki a számokat növekvő sorrendben kettesével!
 6 for i in range(2,10,2): # 2-től 9-ig 2-esével
 7     print(i)
 8
 9 for (k,v) in team.items():
10     print("Player name: %s; #: %d" % (v,k))
11 # Player name: Brown, Josh; #: 3
12 # Player name: Nassib, Ryan; #: 12
13 …
```

While ciklus

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

Python script futtatása

```
1  # vim test.py
2  #!/usr/bin/env python3
3  x = 1
4  for i in range(1, 5):
5      x+=i # nincs ++ operátor
6      print(x, i, 'alma', 'x*x = %d' % (x*x))
7      print(str(i) + " alma")
```

Futtatás:

```
1  $ ./test.py # chmod +x test.py
2  $ python test.py
3  $ py test.py # laborgépeken
```

Függvények

```
1 def is_even(num):
2     if (num % 2) == 0:
3         return True
4     else:
5         return False
6 for i in range(1,10):
7     if (is_even(i)):
8         print("Num:"+str(i))
9 print("Done")
```

Függvények

```
 1  def complex(x):
 2    return x**2, x**3, x**4
 3  print(complex(2))
 4  # (4,8,16)
 5  a, b, c = complex(2)
 6  print(a,b,c)
 7  # 4 8 16
 8  _, rv, _ = complex(2)
 9  print(rv)
10  # 8
```

Lambda Függvények

```
1 is_even = lambda num: (num % 2) == 0
2 is_even_2 = lambda num: True if (num % 2) == 0 else False
3 for i in range(1,10):
4   if (is_even(i)):
5     print("Num:"+str(i))
6 print("Done")
```

List, Dict, Tuple generálás

```
1 mylist = [ x*x for x in range(10) ]
2 # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
3 mydict = { x:x*x for x in range(5) }
4 # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
5 mydict2 = { x:x*x for x in range(5) if x!=2 }
6 # {0: 0, 1: 1, 3: 9, 4: 16}
7 mytuple = tuple( x*x for x in range(3) )
8 # (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_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, 37.00000000000001, 37.5, 38.00000000000001, 39.0]
```

Filter

```
1  fibonacci = [0,1,1,2,3,5,8,13,21,34,55]
2  odd_numbers = list(filter(lambda x: x % 2, fibonacci))
3  print(odd_numbers)
4  # [1, 1, 3, 5, 13, 21, 55]
```

File műveletek

```
1 f = open("demofile.txt", "r")
2 print(f.read())
3 print(f.readline())
4 for x in f:
5   print(x)
6 f.close()
```

```
1 with open("alma.txt", "r") as f:
2    for line in f:
3       print(line.strip().split(","))
```

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

Standard inputtról olvasás

```
1 x = input("Please enter a number:")
2 # x tipusa mindig str!
3 print("Entered number:", x)
```

Parancssori argumentumok

```
1 import sys
2 print(sys.argv[0]) # a script neve
3 print(sys.argv[1]) # első paraméter
4 print(sys.argv[2]) # masodik paraméter
5 …
```

Osztályok

```
 1  class Student:
 2    name = ''
 3    zhpoint = 0
 4    def __init__(self,_name,_point):
 5      self.name =  _name
 6      self.zhpoint = _point
 7
 8    def __str__(self): # human readable
 9      return  f"{self.name}: {self.zhpoint} point"
10
11    def __repr__(self): # machine readable
12      return self.name+"("+str(self.zhpoint)+")"
13
14  p = Student("Ford",20)
15  print(p)    # Ford: 20 point
16  print([p]) # [Ford(20)]
```

Import vs main()

importtest.py

```
1  def main():
2      print("Inside main")
3  if __name__ == "__main__":
4      print("Executed as script")
5      main()
```

testimport.py:

```
1  import importtest
2  importtest.main()
```

vagy:

```
1  from importtest import main
2  main()
```

Output:

```
1  $ python3 importtest.py
2  Executed as script
3  Inside main
4  $ python3 testimport.py
5  Inside main
```

JSON - JavaScript Object Notation

Lásd: https://www.rfc-editor.org/rfc/rfc8259

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

JSON & Python

Dictionary mentése JSON file-ba

```
1 import json
2 data = {
3     "president": {
4         "name": "Zaphod Beeblebrox",
5         "species": "Betelgeusian"
6     }
7 }
8 with open("data_file.json", "w") as write_file:
9     json.dump(data, write_file)
```

JSON formátumú string előállítása Dictionaryből

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

Szerializáció →

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

Deszerializáció ←

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

JSON & Python

JSON objektum beolvasása JSON file-ból

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

JSON & Python

JSON objektum beolvasása JSON formátumú stringből

```
 1 import json
 2 json_string = """
 3 {
 4     "researcher": {
 5         "name": "Ford Prefect",
 6         "species": "Betelgeusian",
 7         "relatives": [
 8             {
 9                 "name": "Zaphod Beeblebrox",
10                 "species": "Betelgeusian"
11             }
12         ]
13     }
14 }
15 """
16 data = json.loads(json_string)
17 for rel in data["researcher"]["relatives"]:
18         print("Name: %s (%s)" % ( rel["name"], rel["species"] ) )
```

Gyakorlás I.

Írjunk függvényt ami megadja a paraméterben kapott évszámról, hogy szökőév-e. Az évszámokat egy file-ból olvassuk be! Egy év szökőév ha osztható néggyel, de akkor nem ha osztható százzal, hacsak nem osztható négyszázzal. Példák:

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

Gyakorlás II.

Írjunk scriptet, ami kiszámolja, hogy hány pont szükséges a zh-n az egyes jegyek eléréséhez. A bemenet egy json-t tartalmazó file legyen, amely tartalmazza a mininet, a beadandók és a ZH-n elért és elérhető maximális pontot. A kimenet pedig az egyes érdemjegyekhez szükséges minimális pont. (Részpont nincs!)

Input:

```
{
  "homework": {
    "max": 4,
    "point": 2,
    "min_percent": 0.5
  },
  "zh": {
    "max": 10,
    "min_percent": 0.5
  },
  "mininet": {
    "max": 100,
    "point": 76,
    "min_percent": 0.5
  }
}
```

Output:

```
$ python zh_grade.py
2: 5
3: 6
4: 10
5: Nope
```