Eemil Aspholm

CT30A3370 Käyttöjärjestelmät ja systeemiohjelmointi

Viikkotehtävä 4.

Tehtävä 1.

```
./lottery.py -s 1 -j 3 -c
./lottery.py -s 2 -j 3 -c
./lottery.py -s 3 -j 3 -c
```

```
** Solutions **

Random 369955 -> Winning ticket 1 (of 54) -> Run 0
  Jobs:
 (* job:0 timeleft:2 tix:54 )
Random 603920 -> Winning ticket 38 (of 54) -> Run 0
  Jobs:
 (* job:0 timeleft:1 tix:54 )
--> JOB 0 DONE at time 2
```

```
** Solutions **

Random 625720 -> Winning ticket 88 (of 114) -> Run 1
  Jobs:
 (  job:0 timeleft:2 tix:54 )  (* job:1 timeleft:3 tix:60 )
Random 65528 -> Winning ticket 92 (of 114) -> Run 1
  Jobs:
 (  job:0 timeleft:2 tix:54 )  (* job:1 timeleft:2 tix:60 )
Random 13168 -> Winning ticket 58 (of 114) -> Run 1
  Jobs:
 (  job:0 timeleft:2 tix:54 )  (* job:1 timeleft:1 tix:60 )
--> JOB 1 DONE at time 3
Random 837469 -> Winning ticket 37 (of 54) -> Run 0
  Jobs:
 (* job:0 timeleft:2 tix:54 )  (  job:1 timeleft:0 tix:--- )
Random 259354 -> Winning ticket 46 (of 54) -> Run 0
  Jobs:
 (* job:0 timeleft:1 tix:54 )  (  job:1 timeleft:0 tix:--- )
--> JOB 0 DONE at time 5
```

```
** Solutions **

Random 13168 -> Winning ticket 88 (of 120) -> Run 1
  Jobs:
  Jobs:
 (  job:0 timeleft:0 tix:--- )  (  job:1 timeleft:0 tix:--- )  (* job:2 timeleft:6 tix:6 )
Random 836462 -> Winning ticket 2 (of 6) -> Run 2
  Jobs:
 (  job:0 timeleft:0 tix:--- )  (  job:1 timeleft:0 tix:--- )  (* job:2 timeleft:5 tix:6 )
Random 476353 -> Winning ticket 1 (of 6) -> Run 2
  Jobs:
 (  job:0 timeleft:0 tix:--- )  (  job:1 timeleft:0 tix:--- )  (* job:2 timeleft:4 tix:6 )
Random 639068 -> Winning ticket 2 (of 6) -> Run 2
  Jobs:
 (  job:0 timeleft:0 tix:--- )  (  job:1 timeleft:0 tix:--- )  (* job:2 timeleft:3 tix:6 )
Random 150616 -> Winning ticket 4 (of 6) -> Run 2
  Jobs:
 (  job:0 timeleft:0 tix:--- )  (  job:1 timeleft:0 tix:--- )  (* job:2 timeleft:2 tix:6 )
Random 634861 -> Winning ticket 1 (of 6) -> Run 2
  Jobs:
 (  job:0 timeleft:0 tix:--- )  (  job:1 timeleft:0 tix:--- )  (* job:2 timeleft:1 tix:6 )
--> JOB 2 DONE at time 11
```

2.

```
./lottery.py -l 10:1,10:100 -c
```

Kun ohjelmaa suoritetaan tällaisella epätasapainolla, on valittava työn numero 1 100 lippua yhteensä 101 lipusta kymmenen iteraatiota peräkkäin. Tästä johtuen työllä numero 0 on 1/(1+100) mahdollisuus eli n. 1% mahdollisuus tulla suoritetuksi ennen tehtävän 1 valmistumista. Prosentuaalisen todennäköisyyden vuoksi voitaisiin sanoa, että tehtävä 1 valmistuu melkeinpä aina ennen tehtävän 0 valmistumista. Epätasapaino vaikuttaa olennaisesti töiden valmistumisjärjestykseen.

3.

Kokeilujen perusteella, kun kumpienkin töiden pituus ja tikettien jako on samansuuruinen, on jakautuminen melko reilua. Testien perusteella voidaan myös nähdä, että mitä pidempi työ on, sitä suurempi on myös oikeudenmukaisuuden todennäköisyys.

```
./lottery.py -s 0 -l 100:100,100:100 -c
```
192/200 = 0.96

```
./lottery.py -s 1 -l 100:100,100:100 -c
```
196/200 = 0.98

```
./lottery.py -s 2 -l 100:100,100:100 -c
```
190/200 = 0.95

```
./lottery.py -s 3 -l 100:100,100:100 -c
```
196/200 = 0.98

4.

Kvanttikoon muuttuessa isommaksi voidaan huomata samanlainen ilmiö kuin työn pituuden pienentyessä -> epätasapaino / epäluotettavuus kasvaa isommaksi.

q-size	UF - metric
1	0.96
2	0.94
3	0.99
4	0.88
5	0.80
6	0.74
7	0.70

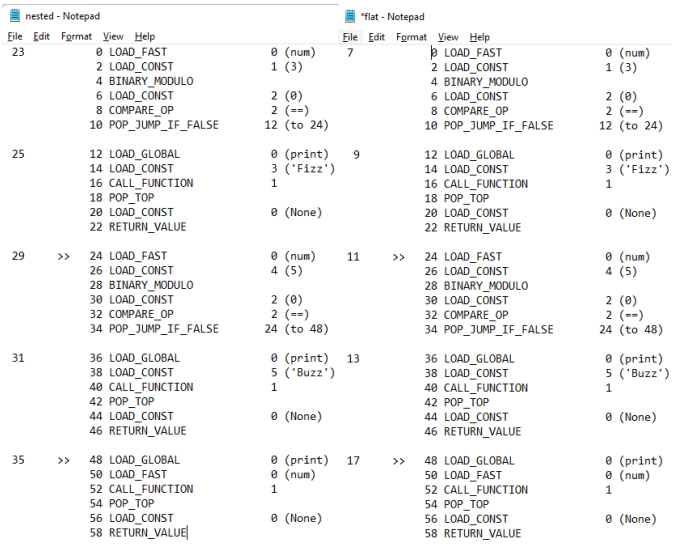
Tehtävä 2.

a. Pythonissa .py päätteiset tiedostot ovat python-koodia jonka ulkoasu on ihmiselle perinteikkäässä ja luettavassa muodossa. Esim (if(x=1) do y = 2x). Pythonin .pyc tiedostot ovat puolestaan tavukoodia, jota ihminen ei pysty lukemaan. Python ohjelmia suoritettaessa python kääntää .py syntaksisen ohjelmatiedoston tavukoodiksi .pyc tiedostoksi, jota se lopulta sitten käyttää tulkintaansa ohjelman suorittamiseksi. Lisäksi jos ohjelmassa on esimerkiksi import käskyllä sisällytettyjä moduuleja ohjelmaan, tällöin pyhon kääntää moduulin sisältämän koodin ja luo .pyc tiedoston, tämä helpottaa ohjelmistokoodin suorittamista, sillä suorituksen yhteydessä moduuleita, joihin ei ole tullut muutoksia ei tarvitse kääntää uudestaan tavukoodiksi.
b. Python ohjelman tavukoodi näyttää omasta mielestäni helpompi tulkinnalliselta verrattuna viime viikkoisen assembly tehtävän koodiin. Pystyn yhdistämään tavukoodista elementtejä .py koodin syntaksiin kuten funktion nimen ”foo”. Tavukoodi itsessään sisältää **.co_code** syntaksin jolla tuodaan esiin että ”foo” on ohjelman funktio/metodi.
[byte for byte in foo .__code__.co_code] syntaksi puolestaan kertoo että funktiossa foo. On luettavia tavuja, ts merkkijono. Mielestäni ohjelmistokoodista

tulisi huomattavasti helpompi tulkintaista, mikäli turha toisto / _co_code jätettäisiin pois perästä.

c. Kuten aiemmin mainittu pythonin tavukoodi omasta mielestäni on huomattavasti loogisempaa ja luettavampaa verrattuna assemblyn syntaksiin. Python ohjelmointikieli ja assembly ovat ensinnäkin erilaisia ohjelmointikieliä, python kuten aiemmin mainittu on tavukoodia, ja asembly puolestaan konekieltä. Pythonin tavukielessä muuttujat ja vakioarvoja tallennetaan pinoon.
d. POP_TOP poistaa pinon ensimmäisen jäsenen, joka käytännössä tarkoittaa, että pinon päällimmäinen operandi on suoritettu.

Tehtävä 3.



nested - Notepad

```
File  Edit  Format  View  Help
 23           0 LOAD_FAST                0 (num)
              2 LOAD_CONST               1 (3)
              4 BINARY_MODULO
              6 LOAD_CONST               2 (0)
              8 COMPARE_OP               2 (==)
             10 POP_JUMP_IF_FALSE       12 (to 24)

 25          12 LOAD_GLOBAL              0 (print)
             14 LOAD_CONST               3 ('Fizz')
             16 CALL_FUNCTION            1
             18 POP_TOP
             20 LOAD_CONST               0 (None)
             22 RETURN_VALUE

 29     >>   24 LOAD_FAST                0 (num)
             26 LOAD_CONST               4 (5)
             28 BINARY_MODULO
             30 LOAD_CONST               2 (0)
             32 COMPARE_OP               2 (==)
             34 POP_JUMP_IF_FALSE       24 (to 48)

 31          36 LOAD_GLOBAL              0 (print)
             38 LOAD_CONST               5 ('Buzz')
             40 CALL_FUNCTION            1
             42 POP_TOP
             44 LOAD_CONST               0 (None)
             46 RETURN_VALUE

 35     >>   48 LOAD_GLOBAL              0 (print)
             50 LOAD_FAST                0 (num)
             52 CALL_FUNCTION            1
             54 POP_TOP
             56 LOAD_CONST               0 (None)
             58 RETURN_VALUE
```

*flat - Notepad

```
File  Edit  Format  View  Help
  7           0 LOAD_FAST                0 (num)
              2 LOAD_CONST               1 (3)
              4 BINARY_MODULO
              6 LOAD_CONST               2 (0)
              8 COMPARE_OP               2 (==)
             10 POP_JUMP_IF_FALSE       12 (to 24)

  9          12 LOAD_GLOBAL              0 (print)
             14 LOAD_CONST               3 ('Fizz')
             16 CALL_FUNCTION            1
             18 POP_TOP
             20 LOAD_CONST               0 (None)
             22 RETURN_VALUE

 11     >>   24 LOAD_FAST                0 (num)
             26 LOAD_CONST               4 (5)
             28 BINARY_MODULO
             30 LOAD_CONST               2 (0)
             32 COMPARE_OP               2 (==)
             34 POP_JUMP_IF_FALSE       24 (to 48)

 13          36 LOAD_GLOBAL              0 (print)
             38 LOAD_CONST               5 ('Buzz')
             40 CALL_FUNCTION            1
             42 POP_TOP
             44 LOAD_CONST               0 (None)
             46 RETURN_VALUE

 17     >>   48 LOAD_GLOBAL              0 (print)
             50 LOAD_FAST                0 (num)
             52 CALL_FUNCTION            1
             54 POP_TOP
             56 LOAD_CONST               0 (None)
             58 RETURN_VALUE
```

Yläpuolelle liitettyjen funktioiden tavukoodit ovat identtiset, eivätkä ne eroa, vaikka ohjelmistokoodin syntaksi eroaa else if /elif rakeenteella. Voidaan siis tehdä johtopäätös, että elif/else if syntaksin tavukoodi on identtinen kuten myös niiden toiminnallisuus. Yritin myös vaihtaa else if osioiden järjestystä, mutta en saanut silläkään muutoksia tavukoodin syntaksiin. En ole aivan varma teinkö oikeanlaisia muutoksia, sillä tehtävän annon kysymys oli hieman epäselvä. Mikäli .py tiedostossa muutetaan else if rakenteen sijaintia, muuttuu myös tavukoodi tiedoston syntaksin sijainti.