Python-part3

May 24, 2024

1 File Reading

The file `test.csv` should be in the same directory and have the form:

```
name, surname, regno, grade
student1, surname1, 3108144, 8
student2, surname2, 3106140, 6
student3, surname3, 3108145, 9
student4, surname4, 3106142, 7
student5, surname5, 3108146, 10
```

[140]:
```
f = open('test.csv','r')
type(f)
```

[140]:
```
_io.TextIOWrapper
```

[141]:
```
l = next(f)
print(l)
```

```
name, surname, regno, grade
```

[142]:
```
# create a generator that that yield only the lines for which p(regno)
def readFile(file, p):
    f = open(file,'r')
    next(f)
    for line in f:
        _, _, regno, grade = line.split(", ")
        if p(regno): yield (int(regno), int(grade))

def average(gen):
    sum = 0
    N = 0
    for regno, grade in gen:
        sum += int(grade)
        N += 1
    if N: return(sum/N)
    else: print("No data!")
```

[143]:
```
def year06(s): return (s[2] == '0' and s[3] == '6')

# Compute the avarage of the students that started in 2006
g = readFile('test.csv', year06)
## or:
## g = readFile('test.csv', lambda s: s[2] == '0' and s[3] == '6')

print(average(g))

# Compute the average of all students in the database
g = readFile('test.csv', lambda _: True)

print(average(g))
```

```
6.5
8.0
```

[144]:
```
def fileData(filename1):
    file = open(filename1, "r")
    next(file); # consume the first line
    for line in file:
        name, surname, reg, grade = line.split(',')
        yield (int(reg),int(grade))

type(fileData("test.csv"))
```

[144]: generator

1.1 Exercise 4: Goat

Hint!

Collections library

[146]:
```
from collections import deque

init = (frozenset({'m', 'w', 'g', 'c'}), frozenset())

bad1 = frozenset({'w','g'})
bad2 = frozenset({'g','c'})


def isFinal(s): return (not s[0])

def nextStates(s):
    side = 0 if 'm' in s[0] else 1
    for x in s[side]:
        src = s[side] - frozenset({'m', x})
        dst = s[1 - side] | frozenset({'m', x})
```

```
        if (bad1 <= src or bad2 <= src): continue
        yield (src, dst) if side == 0 else (dst, src)

queue = deque([init])
prev = {init: None}

def bfs(init):
    while queue:
        s = queue.popleft()
        for n in nextStates(s):
            if isFinal(n):
                prev[n] = s
                return n
            if n not in prev:
                queue.append(n)
                prev[n] = s

    return None

def printSol(s):
    if not (s == None):
        p = prev[s]
        printSol(p)
        print(s)


final = bfs(init)

if (final == None): print("No solution found")
else: printSol(final)
```

```
(frozenset({'m', 'w', 'g', 'c'}), frozenset())
(frozenset({'w', 'c'}), frozenset({'m', 'g'}))
(frozenset({'m', 'w', 'c'}), frozenset({'g'}))
(frozenset({'c'}), frozenset({'m', 'g', 'w'}))
(frozenset({'m', 'g', 'c'}), frozenset({'w'}))
(frozenset({'g'}), frozenset({'m', 'w', 'c'}))
(frozenset({'m', 'g'}), frozenset({'w', 'c'}))
(frozenset(), frozenset({'m', 'w', 'g', 'c'}))
```

[ ]: