Exercise 1

András Novoszáth

```
from math import e

def factor(num):
    fact = 1
    if num != 0:
        for i in range(1, num + 1):
            fact *= i
    return fact

def pois_dist(average_success, actual_success):
    prob_k_success = (average_success ** actual_success * e ** (-average_succe
    return prob_k_success

avg_rate = 5
error_rate = 10

# First I calculate the probability of functioning without error.
prob_no_error = 0
for file in range(1, error_rate):
    prob_no_error += pois_dist(avg_rate, file)

#Now we can get the daily probability of error.
prob_error = 1 - prob_no_error
print(prob_error)

days = 7 # I assume that the program works 7 days a week
min_fails = 2

# The probability of having at least two days with errors a week is equals wit
# having maximum one error-less days a week:

prob_min_2_errs = 1 - prob_no_error ** 7 - prob_no_error ** 6 * prob_error
print(prob_min_2_errs)
```

Exercise 1

András Novoszáth

Answer

It is not entirely clear from the description whether the program would stop after 10 files or would produce an error but than continue to work.

The probability of having at least two errors a week is 0.2102 or around 21%.

Exercise 2

András Novoszáth
One of the Reply companies uses a keypad (with the numbers 1 to 9) to unlock their office door.
They used to have a 4-digit PIN, without repetition of digits. Now, management has decided that this
should be changed to a 6-digit PIN, without repetitions, in order to increase security. To help the
employees remember the PIN, they decide that the new 6-digit PIN should always contain one of the
left-to-right rows or top-to-bottom columns on the keypad, i.e. one of the sequences 123, 456, 789,
147, 258, or 369.
1 2 3
4 5 6
7 8 9
Calculate the number of variations for both the old and the new PIN system. Is the new rule actually
more secure?

Answer

The number of variation in the old system:
vars_old = 9! - 5! = 362760
print(vars_old)

The number of variations in the new system:
vars_new = 9! - 3! - (permustations of the six sequences)

Exercise 3

András Novoszáth

Question 1

Answer:

If we know the starting hour: `s` , we can get the probability of `H` with poisson distribution where the average value is T*p and the expected value is

- if H > S+T: `h-(s+t)`
- if H < S+T: `h+(s+t)`

Question 2

Answer:

If the clock really 'advances' an hour (and only by one single hour) than it is impossible because (in case I sleep 8 hours in order to show h=6 it should advance 4+6=10 hours.)

If it instead "goes back" with `f` probability, then the change of sleeping 8 hours is as follows:

cases where there clock shows h=6 = 0.5 * poisson(avg= 0.6 * 7, actual = 1) + 0.5 * poisson(avg= 0.6 * 8, actual = 2)

- There is 0.5 probability of sleeping either 8 or 7 hours
- When sleeping 7 and 8 hours the clock needs to step back 1 and 2 hours respectively
- The probability of stepping stepping back can be calcaulted with poissoin distribution

Nonetheless, I have not checked for only arriving to whole hours, this might change further the calculation.

Exercise 4

András Novoszáth

Exercise 5

András Novoszáth

```python
inp1 = ["aba", "ba", "cam", "pin"]
inp2 = ["vat", "dui", "bin"]
out = ["vat", "pin", "dui", "cam", "bin", "ba", "aba"]

# First we reorder the lists into decreasing order
def reorder(list):
    newlist = []
    if list[0] < list[-1]: # we decide whether the list is in increasing or de
        decrord = []
        for i in list:
            decrord.append(list.pop()) # if it is increasing order we change t
        newlist = decrord
    else:
        newlist = list
    return newlist

def mergelist(list1, list2):
    ordlist1 = reorder(list1)
    ordlist2 = reorder(list2)

    mergedlist = []

    for item in ordlist1: # we compare the first elements of each decreasing o
        if ordlist1[0] > ordlist2[0]:
            mergedlist.append(ordlist1.pop(0))
        else:
            mergedlist.append(ordlist2.pop(0))

    for item in ordlist2:
        mergedlist.append(ordlist2.pop(0))

    return mergedlist

inp12 = mergelist(inp1, inp2)
print(inp12)
```

Exercise 6

András Novoszáth

Answer

```
multiples = 0
for num in range(1, y):
    if num % x1 == 0:
        multiples += num
    elif num % x2 == 0: # we use elif so it will not count twice those which a
        multiples += num

print(multiples)
```

Exercise 7

András Novoszáth

```
from math import sqrt

filenames = ['file0.txt', 'file1.txt', 'file2.txt', 'file3.txt', 'file4.txt',

U1 = 1 + sqrt(1 + 4 * 2 * len(filenames)) / 2
U2 = 1 - sqrt(1 + 4 * 2 * len(filenames)) / 2

print(U1)
print(U2)

for U in [U1, U2]:
    print("distribute elements among U amont of lists")
    if U2 < 2:
        pass

# for files in filenames:
```