COMP S362F Lab 1 Concurrency concepts and Python revision

Question paper

Exercises

CPU-bound and I/O-bound tasks

Determine whether each of the following tasks is CPU-bound or I/O-bound, and justify your answer.

(1) Find the first million digits of π. (E.g. the first 10 digits of π are 3.141592654.)
(2) Find the largest file in a file system that contains many files.
(3) Find the sha256sum (a checksum algorithm) value of a long string in memory.
(4) Find the sha256sum (a checksum algorithm) value of a large file.

Python basics

Given the variable `s`, `s = "abcde"`, determine the value of each of the following expressions – (1) to (4). If an expression results in an error, explain why.

```
s = "abcde"

# (1)
s[5]

# (2)
s[-1: -2]

# (3)
s[-1: 0: -2]

# (4)
[c for c in s if c > "c"]
```

Practicals

Factorial

The *factorial* of a non-negative integer n, $n!$, is the product of all positive integers less than or equal to n:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

E.g. $4! = 4 \times 3 \times 2 \times 1 = 24$. For the value $n = 0$, $0! = 1$. (Ref: https://en.wikipedia.org/wiki/Factorial)

(1) Write a function, called `factorial_loop()`, that computes the factorial of an integer n using iteration, i.e. a loop.

(2) Write a function, called `factorial_recur()`, that computes the factorial of an integer n using recursion. The recursive definition of factorial is given below.

$$n! = \begin{cases} 1 & \text{if } n \leq 1 \\ n \times (n-1)! & \text{if } n > 1 \end{cases}$$

(3) Write a function, called `factorial_approx()`, that computes the factorial of an integer n using Stirling's approximation for large n as follows. (Hint: use `math.sqrt()`, `math.pi`, and `math.e`.)

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Palindrome

A *palindrome* is a word whose sequence of characters read the same forward and backward. E.g. some palindromes are "ada", "madam", "a", and the empty string "". (Ref: https://en.wikipedia.org/wiki/Palindrome)

! Submission

Design a recursive algorithm to determine whether a string is a palindrome, and write a recursive function called `palindrome_recur()` to implement the algorithm. You may assume that all characters of the string are the same case, i.e. all lowercase or all uppercase. Test the function with strings "", "a", "ada", "madam", "an", "the", and others.

Write the `palindrome_recur()` function in the `lab01.py` file for submission.

Myth value

Random numbers are useful in many wonderful ways. The following algorithm uses random numbers to compute a well-known mathematical constant.

```
* Algorithm myth_value(n) *
initialize count to 0
do n times:
    set x to a random value between 0.0 and 1.0
    set y to a random value between 0.0 and 1.0
    if x * x + y * y is less than 1:
        add 1 to count
return count / n * 4
```

> **!** Submission
>
> Write a function called `myth_value()` to implement the above algorithm. (Hint: use the `random()` function of the `random` module.) Test the function with values of n being 10,000, 100,000, 1,000,000, 10,000,000, and others. What is the resulting mathematical constant?
>
> Write the `myth_value()` function in the `lab01.py` file for submission.

The Python `random` module and its `random()` function may be confusing due to the same name. These are two equivalent examples of printing a number between 0 and 1 with different syntaxes.

```python
# File: random1.py
from random import random  # I.e. from module_name import function_name
print(random())            # I.e. function_name()
```

```
0.9503534963568846
```

```python
# File: random2.py
import random           # I.e. import module_name
print(random.random())  # I.e. module_name.function_name()
```

```
0.0816200253410575
```

Submission notes

(1) Write your code in the provided skeleton program file.
(2) Write your full name and 8-digit student number near the beginning of the file.
(3) Submit the completed file to OLE before the cut-off time. Late or incorrect submission will result in zero mark.

(4) Do not modify the function signatures and constant names in the file. The program is graded by both automatic testing and manual examination.
(5) If necessary, add import statements, variables, functions, classes to the file.

```
# File: lab01.py

# Replace the following by your full name and 8-digit student number
student_name = "Chan Tai Man"
student_id = "01234567"

from random import random

def palindrome_recur(s):
    pass  # replace this line by your code

def myth_value(n):
    pass  # replace this line by your code

if __name__ == "__main__":
    pass  # replace this line by your code (if necessary)
```

Extras

Factorial

(1) Rewrite the function `factorial_recur()` as a lambda expression. (Hint: use a conditional expression.)
(2) Rewrite the function `factorial_approx()` as a lambda expression.

Palindrome

(1) Write a function called `palindrome_loop()` to determine whether a string is a palindrome using iteration.
(2) Write a function called `palindrome_slice()` to determine whether a string is a palindrome using a slice to reverse the string.

Myth value

(1) Rewrite the `myth_value()` function as a (long) lambda expression. (Hint: use list comprehension, a conditional expression, and the `sum()` function.)