COMP9021 Mid Term Summary

1. Formatted Output

   Lab1 fahrenheit_to_celcius.py

```
for fahrenheit in range(min_temperature, max_temperature + step, step):
    celsius = 5 * (fahrenheit - 32) / 9
    # {:10d} or {:10}:  fahrenheit as a decimal number in a field of width 10
    # {:7.1f}: celsius as a floating point number in a field of width 7
    #          with 1 digit after the decimal point
    print(f'{fahrenheit:10}\t{celsius:7.1f}')
```

2. Try – Except Statement

   Lab1 span.py

```
try:
    arg_for_seed = int(input('Input a seed for the random number generator: '))
except ValueError:
    print('Input is not an integer, giving up.')
    sys.exit()
try:
    nb_of_elements = int(input('How many elements do you want to generate? '))
except ValueError:
    print('Input is not an integer, giving up.')
    sys.exit()
if nb_of_elements <= 0:
    print('Input should be strictly positive, giving up.')
    sys.exit()
```

3. Random and Seed

   Lab1 span.py

```
# Generates a list of nb_of_elements random integers between 0 and 99.
seed(arg_for_seed)
L = [randint(0, 99) for _ in range(nb_of_elements)]
# Prints out the list, computes the maximum element of the list, and prints
```

`random()` 函数中常见的函数如下：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import random

print( random.randint(1,10) )        # 产生 1 到 10 的一个整数型随机数
print( random.random() )             # 产生 0 到 1 之间的随机浮点数
print( random.uniform(1.1,5.4) )     # 产生  1.1 到 5.4 之间的随机浮点数，区间可以不是整数
print( random.choice('tomorrow') )   # 从序列中随机选取一个元素
print( random.randrange(1,100,2) )   # 生成从1到100的间隔为2的随机整数

a=[1,3,5,6,7]                # 将序列a中的元素顺序打乱
random.shuffle(a)
print(a)
```

4. / & // - the differences in python2.7 and python3

   In python3: type(A / B) = float; type(A / B) = int;

   In python2.7: type(A / B) = int

   Lab1 intervals.py

```
# - intervals[0] to record the number of elements between 0 and 4,
#   that is, elements e such that e // 5 == 0
# - intervals[1] to record the number of elements between 5 and 9
#   that is, elements e such that e // 5 == 1
# - intervals[2] to record the number of elements between 10 and 14
#   that is, elements e such that e // 5 == 2
# - intervals[3] to record the number of elements between 15 and 19
#   that is, elements e such that e // 5 == 3
intervals = [0] * 4
for e in L:
    intervals[e // 5] += 1
```

5. Get mean and standard deviation

   Lab1 man_median_standard_deviation.py

```
def find_median(L):
    L.sort()
    median = (L[int(len(L) / 2 - 1)] + L[int(len(L) / 2)]) / 2 \
                    if len(L) % 2 == 0 else L[int(len(L) / 2)]
    return median

def find_standard_deviation(L):
    mean = find_mean(L)
    var = 0
    for i in range(len(L)):
        var += (L[i] - mean) ** 2
    standard_deviation = sqrt(var / len(L))
    return standard_deviation
```

6. Find the number of trailing 0s in a factorial

   Find the number of multiples of 5

   Lab2 trailing_0s.py

```
def third_computation(x):
    cnt = 0
    while x // 5:
        cnt = cnt + x // 5
        x = x // 5
    return cnt
```

7. Find the divisor of a number: start from 2 and end with n // 2

   Lab2 perfect.py

```
# Replace pass above with your code to check whether i is perfect,
# and print out that it is in case it is.
# 1 divides i, so counts for one divisor.
# It is enough to look at 2, ..., i // 2 as other potential divisors.
sum_div = 0
for j in range(1, number // 2):
    if number % j == 0:
        sum_div += j
        # print(f'{j} of {number}')
if sum_div == number:
    perfect_number.append(number)
```

8. Find if all the numbers in a list are the same: use set

   Lab2 multiplication.py

```
col[0] = mid1 // 1000 + mid2 // 100 + result // 1000
col[1] = x // 100 + (mid1 // 100) % 10 + (mid2 // 10) % 10 \
         + (result // 100) % 10
col[2] = (x // 10) % 10 + y // 10 + (mid1 // 10) % 10 + \
         mid2 % 10 + (result // 10) % 10
col[3] = x % 10 + y % 10 + mid1 % 10 + result % 10
set_col = set(col)
if len(set_col) == 1:
    print(f'{x} * {y} = {result}, all columns adding up to {col[0]}.')
```

9. Find prime numbers

   Lab3 consecutive_primes.py

```
def is_prime_number(number):
    is_prime = True
    for i in range(2, number // 2):
        if number % i == 0:
            is_prime = False
            break
    return is_prime
```

10. Dictionary

    Lab3 triples_2.py

```
def all_possible_integers():
    integer_dict = dict()
    for i in range(32):
        for j in range(i, 32):
            if i ** 2 + j ** 2 >= 100 and i ** 2 + j ** 2 < 1000:
                integer_dict[i ** 2 + j ** 2] = [i, j]
    all_integers = sorted(integer_dict.items(), \
                          key = lambda integer_dict: integer_dict[0])
    return all_integers
```

11. Ord(char) & chr(number): ord('A') = 65, ord('Z') = 90, ord('a') = 97, ord('z') = 122

    Lab4 characters_triangle,py

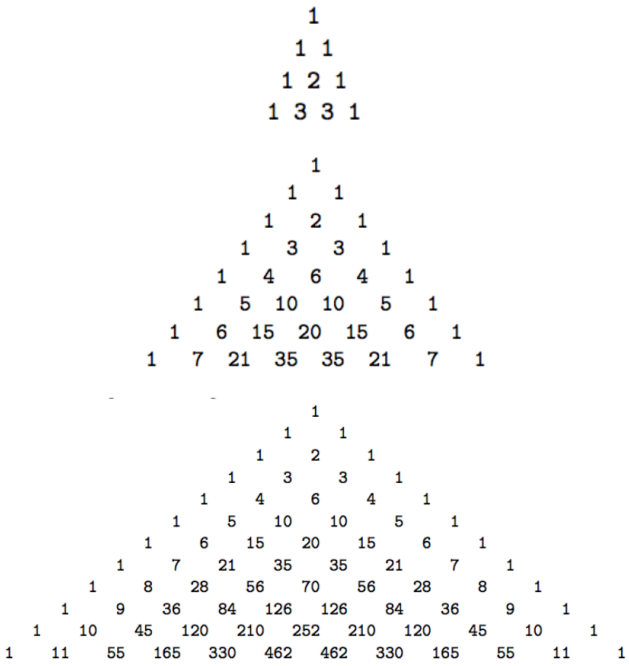
```
def display(height):
    all_char = []
    nb_of_char = 0
    idx = 0

    for i in range(1, height + 1):
        nb_of_char += i

    for j in range(0, nb_of_char):
        all_char.append(chr(65 + j % 26))

    for nb in range(1, height + 1):
        tmp = ''
        reverse_tmp = ''
        for cur1 in range(nb):
            tmp += all_char[idx + cur1]
        for cur2 in range(2, nb + 1):
            reverse_tmp += tmp[- cur2]
        idx += nb
        print(' ' * (height - nb) + tmp + reverse_tmp)
```

12. Create and display pascal_triangle



Create a list of lists to store each line in pascal triangle.

```
def create_char_set(N):
    char_set = []
    if N == 0:
        char_set.append([1])
    elif N == 1:
        char_set.append([1])
        char_set.append([1, 1])
    else:
        char_set.append([1])
        char_set.append([1, 1])
        for cur in range(3, N + 2):
            tmp = [1]
            for i in range(cur - 2):
                tmp.append(char_set[-1][i] + char_set[-1][i + 1])
            tmp.append(1)
            char_set.append(tmp)

    return char_set
```

When displaying the pascal triangle, you must pay attention to the space between numbers and the space before each line.

```
def display(char_set):
    height = len(char_set)
    max_bit = find_max_bit(char_set)
    for i in range(height):
        tmp = ' ' * (height - i - 1) * max_bit
        for nb in range(len(char_set[i])):
            if not nb == len(char_set[i]) - 1:
                tmp += ' ' * (max_bit - len(str(char_set[i][nb]))) \
                       + str(char_set[i][nb]) +  ' ' * max_bit
            else:
                tmp += ' ' * (max_bit - len(str(char_set[i][nb]))) \
                       + str(char_set[i][nb])

        print(tmp)
```

13. Open and read files

    Assignment1 rain.py

    F = open(open_file, ‘r’); FileNotFoundError

```
try:
    open_file = input("Which data file do you want to use? ")
    f = open(open_file, 'r')
except FileNotFoundError:
    print("Error!There is not such file in current directory.")
    sys.exit()
```

Read each line of the file.

```
height_of_land = []
with open(open_file) as land_file:
    for line in land_file:
        each_line = list(filter(None, line.split(' ')))
        height_of_land += each_line
f.close()
```