Computational Thinking Module - Python

Computational Thinking Team 2024/2025

Notes

1. This module is designed to guide Computational Thinking programming, then some materials may not exist because they are out of Computational Thinking scope.
2. You may open this module while you are doing the module exercise in practical class.
3. It is recommended to run all programs listed in this module in your computer, so you know the output of the programs.
4. It is better to experimenting on the programs listed in this module to grasp better understanding about what your programs really do.
5. You are strongly recommended to learn programming from other sources or materials and explore your programming language.

Contents

1 Module 1 - Input, Output, and Conditional Statements

1.1 Introduction

In this module, we use Python 3.9. You can download Python in https://www.python.org/downloads/release/python-390/. Choose option "Windows x86-64 executable installer".

Some characteristics of this language:

- Python is *case-sensitive*, which means caps and non-caps characters have different meaning.
- Python is very sensitive to indentation and line separation. Wrong usage of indentation may cause compilation failure.
- Variables in Python are implicit and dynamic. Then, data type of a variable doesn't need to be explicitly stated. But in this module, we still learn all data types.

1.2 Input and Output

In Python, program that print `"Hello, World!"` to screen :

```
print("Hello, World!")            # (1)

print("This is", end="")          # (2)
print(" my first program")        # (3)

print("Using","Comma")
#This will outputs "Using Comma"

print("Using"+"Plus")
# This will outputs "UsingPlus" which we call concatenation
# Be mindful that concatenation will only works on strings,
# such you will need to convert other data types.

variable = "some value"
print(f"Using format: {variabel}")
# This will outputs "Using format: some value"
# {variable} will be replaced with the value of corresponding variable

# This is a comment

# Everything after hash symbol
# will be ignored by interpreter

'''
Moreover, we may wrap out multiline comments
with three single-quote (''')
'''
```

The section marked with number 1 writes `"Hello, World!"` to the screen. The syntax `print` will automatically move the cursor to the next line after it has finished writing to the screen,

unless we add the `end=""` parameter. The program above will output `"Hello, World!"` followed by `"This is my first program"` in the next line.

To read inputs, we need a container to store the data that is inputted. For example, below is a program that receives input and rewrites the input entered.

```
S = input("Input some words: ")                                    # 1
print("You inputted " + S)                                         # 2

N = int(input("Input an integer: "))                               # 3
print("If your number added with 5, th result is " + str(N + 5)) # 4
```

In part number (1) and (3), the program reads the input from the user and the input is entered into the variable `S` and `N`. Then, if you want the program to receive a number (so it can be added / multiplied / other), we wrap `input()` with `int()`. Besides `int`, `string` (collection of characters), Python can accept `float` (real number) data type, and many more.

1.3 Data Types

There are several types of data, but in Computational Thinking we will only use the following data types:

`bool`	Boolean	`True` or `False`
`int`	Integer	all integer numbers
`float`	Real	all real numbers
`string`	Text	collection of characters

Example of use:

```
B = True          # Boolean
I = 123456789012  # Integer
R = 2.331973      # Real
S = "abc"         # Text
S = 'def'
```

Pay attention to the `S` variable, you are free to use single quotes (`'`) or double quote (`"`).

1.4 Operator

1.4.1 Arithmetic Operator

Operator	Description	Example
`+`	Addition	`2 + 3` evals to `5`
`-`	Substraction	`1 - 8` evals to `-7`
`*`	Multiplication	`5 * 6` evals to `30`
`/`	Division	`13 / 5` evals to `2.6`
`//`	Division (rounded down)	`13 // 5` evals to `2`
`%`	Modulo	`13 % 5` evals to `3`

1.4.2 Assignment Operator

Operator	Description	Example
=	Assignment	`N = 5`
+=	Addition	`N += 5`, N will be added by 5.
-=	Substraction	`N -= 5`, N will be substracted by 5.
*=	Multiplication	`N *= 5`, N will be multiplied by 5.
/=	Division	`N /= 5`, N will be divided by 5.
//=	Division (Rounded Down)	`N //= 5`, N will be divided by 5 (then rounded down).
%=	Modulo	`N %= 5`, N will be modulo by 5.

1.4.3 Relational Operator

Operator	Description	True Example	False Example
==	Equals	`2 == 2`	`2 == 3`
!=	Not equals	`3 != 2`	`3 != 3`
<	Less than	`2 < 3`	`2 < 2`
>	Greater than	`3 > 2`	`3 > 2`
<=	Less or equal	`2 <= 2`	`1 <= 3`
>=	Greater or equal	`6 >= 5`	`2 >= 4`

1.4.4 Logical Operator

Operator	Description	True Example	False Example
and	And	`(1 < 2)and (3 == 3)`	`(1 == 2)and (3 == 3)`
or	Or	`(1 < 2)or (4 == 3)`	`(3 < 2)or (2 == 3)`
not	Negation	`not (3 < 2)`	`not (1 > 2)`

1.5 Control

In programming, we can control the flow of our program. Thus, our program can behave depending on user input. Suppose we make a program that checks whether a positive number:

```
print("Input value of N: ", end="")
N = int(input())

if (N > 0):
    print(str(N) + " is positive number")

# str(N) is used to convert N to string such that it can be concatenated.
```

Then, if we want to handle the non-positive number:

```
...
    if (N > 0):
        print(str(N) + " is positive number")
    else: # N <= 0
```

```
        print(str(N) + " is not positive number")
...
```

However, we know that sometimes numbers can be zero or negative, so we add:

```
...
    if (N > 0):
        print(str(N) + " is positive number")
    elif (N < 0):
        print(str(N) + " is negative number")
    else: # N == 0
        print(str(N) + " is zero")
...
```

Also notice that we can repeat the else. In addition, we can also put `if` in `if`.

```
...
    if (N >= 0):
        if (N > 0):
            print(str(N) + " is positive number")
        else: # N == 0
            print(str(N) + " is zero")
    else: # N < 0
        print(str(N) + " is negative number")
...
```

1.6 Module 1 Exercise

1.6.1 Even-Odd Program

With the program which determines positive and negative numbers above, modify it such that if the number is positive will also tell whether it is odd or even.

Example 1

```
Input value of N: 20
20 is an even positive integer
```

1.6.2 Practice Problem 1 Year 2022

Mr. Riz is taking an exam consisting of 14 short-answer questions and 2 essay questions. The exam lasts for 2 hours. During the first hour, he focused on solving only a few questions. He is worried that he might not be able to complete all questions on time. Mr. Riz plans to work on 1 remaining short-answer question for 10 minutes and 1 remaining essay question for 20 minutes.

Determine if Mr. Riz manages to complete all questions on time.

Example 1

```
Enter the number of short-answer questions already completed: 12
Enter the number of essay questions already completed: 1
Mr. Riz will successfully complete all questions
```

Mr. Leo refers to a number as a "Bravo" number if the sum of each of its digits is evenly divisible by each of its digits. If a number does not meet this criterion, it is referred to as an ordinary number, and a 4-digit number is provided. Determine the category of the number.

Hint: Use division and modulo to get the value of each digit.

Example 1

```
Enter a number: 4228
The number is a Bravo number.
```

Explanation for Example 1: The sum of each digit in the number 4228 is 4 + 2 + 2 + 8 = 16, and 16 is evenly divisible by each of its digits, which are 2, 4, and 8, so it is called a "Bravo" number.

Example 2

```
Enter a number: 3425
The number is an ordinary number.
```

1.6.3 Practice Problem 3 Year 2022

A frog observed by Mr. Riz is on an x-axis. The frog jumps based on a simple pattern. The first jump is a units to the right, the second jump is b units to the left, the third jump is a units to the right, the fourth jump is b units to the left, and so on for k times. The number k is obtained based on the frog's initial position with the following rules:

- If the frog is at an even coordinate, its k value is the frog's ability to jump to the right (a) multiplied by the frog's ability to jump to the left (b).
- If the frog is at an odd coordinate, its k value is $a \times 2$.
- If the frog is at a negative coordinate, it returns to the first rule ($a \times b$).
- However, if the frog is at a negative coordinate that is a multiple of 3, its k value is $b \times 2$.

Determine the current coordinate of the frog.

Example 1

```
Enter the coordinate: 0
Enter the length of jump to the right: 5
Enter the length of jump to the left: 1
Current coordinate is 13
```

Explanation for Example 1: Since 0 falls under the first rule, the frog will jump 5 times. The calculation of the frog's coordinate will be like this: 0 + 5 - 1 + 5 - 1 + 5 = 13

Example 2

```
Enter the coordinate: -3
Enter the length of jump to the right: 2
Enter the length of jump to the left: 5
Current coordinate is -18
```

2 Module 2 - Looping and Array

2.1 Looping

In programming, often we need looping to achieve a certain result. If this repetition is done manually, the source size will become too large. For example, if we want to write "Hello World" on the screen 1000 times, then it will need at least 1000 statements. Using looping, the problem can be completed using only a few lines of the program.

```
for i in range(1000):
    print("Hello world")
```

2.1.1 While Loop

One of the syntax of looping that is often used is While-Do syntax. The program will check a condition that is given beforehand run the statement inside it.

Here are the programs that accept a and b then write $a, a+1, a+2, \ldots, b-1, b$.

```
a = int(input())
b = int(input())
i = a
while (i <= b):
    print(i)
    i += 1
```

Notice that:

1. While Loop Syntax is **while (condition):**
2. While loop body will be executed until the condition is false.
3. Body while loop is the code that is indented inside the loop.

2.1.2 For Loop

The second form of looping is the form For. This form is generally used if the number of looping is known. However, for loops can also be made with a while-do loop.

Here are the programs that accept a and b then write $a, a+1, a+2, \ldots, b-1, b$.

```
a = int(input())
b = int(input())
for i in range(a, b+1):
    print(i)
```

Notice that

1. For Loop syntax is **for variable in range(value):**
2. For loop body will be executed determined by the range given.
3. For loop body is the code that is indented inside the loop.
4. Variable value will change each time the loop is executed for the values in range.

`range` can be used with one, two, or three parameters. For more explanation, see this code:

```
range(10)          # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
range(2, 5)        # 2, 3, 4
range(7, 2, -1)    # 7, 6, 5, 4, 3
range(2, 8, 2)     # 2, 4, 6
```

2.2 Nested Loop

Looping can also be nested. For example, the following program will output a square pattern of asterisk.

```
n = int(input())
for i in range(n):
    for j in range(n):
        print('*', end='') # Executed n * n times
    print() # Executed n times
# Always mindful of indentations in loops.
```

This also shows that code inside the body of a loop can be other code such as I/O, operations, other loops. You can use while loop inside a for loop and vice versa.

2.3 Array

Arrays are variables with one name, but they contain many values. Access the values done with an index.

Consider the following example:

Indeks	0	1	2	3	4	5	6	7	8	9
A	3	10	5	7	11	19	23	35	37	12

- `A[0] = 3`
- `A[1] = 10`
- `A[7] = 35`

In the example above, we have a variable named `A`. The variable `A` has 10 values, where these values can be accessed with an index. To access the `x`-index, use `A[x]`. And we can consider the value of `A[x]` as a stand-alone variable. This concept is what we call arrays. Also note that the index starts from 0.

2.3.1 Array Declaration

Because arrays are also a variable, the array also requires declarations like other variables.

Example of array declaration:

- `x = [0 for x in range(n)]` create an array sized `n` with all contains zero.
- `x = ['*'for x in range(100)]` create an array sized `100` with all contains `'*'`.

In above example, the array `A` consists of `A[0]`, `A[1]`, `A[2]`, ... `A[9]`. Accessing index out of bound will cause a runtime error. Therefore, declare the range of indices that will be used appropriately.

2.3.2 Array and Variable

An array can be considered a variable, so that all types of operations in the variable also apply to the array. For example, we have an array

```
arr = [0 for i in range(10)]
```

Then the array `arr` will be defined for index 0 to index 9. Then we can do the instruction

```
arr[2] = int(input())
```

Given 5 numbers we can do

```
arr[0] = int(input('Input value index 0: '))
arr[1] = int(input('Input value index 1: '))
arr[2] = int(input('Input value index 2: '))
arr[3] = int(input('Input value index 3: '))
arr[4] = int(input('Input value index 4: '))
```

However, the input method is not efficient. It will be more efficient if we input using looping.

```
for i in range(5):
    arr[i] = int(input('Input value index ' + str(i) + ': '))
```

2.3.3 String

In the first module, we have learned about string. String can be *considered* as an array of characters. For example, the string s1

```
s1 = 'Hello'
```

can be *considered* as an array of characters `H`, `e`, `l`, `l`, `o`. Therefore, we can access the character in the string using the index. For example, `Hello[0]` is `H`.

But, there is a difference between string and array of characters. String is immutable, while array of characters is mutable. This means that we cannot change the value of the character in the string. For example, we cannot do `s1[1] = 'a'` as this will results into an error.

In contrast, if we have an array of characters, we can change the value of the character. For example, if we define s2 as

```
s2 = ['H', 'e', 'l', 'l', 'o']
```

then we can change the value of the character in the array. For example, we can do `s2[1] = 'a'`. This will change the value of the character in the array of characters to a that is `s2 = ['H', 'a', 'l', 'l', 'o']`.

2.4 Module 2 Exercise

2.4.1 Smallest Power of 10

Create a program that receives an integer N and write the smallest integer 10^x that is greater or equal than N.

Example 1

```
Input N: 5
10
```

Example 1

```
Input N: 1234
10000
```

Which loop is more appropriate for this question?

2.4.2 Number Triangle

Create a program that receives an integer N and create a number triangle with a base of N as in Example below.

Example 1

```
Input N: 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Which loop is more appropriate for this question?

From both of the above, what can we conclude about the characteristics of each loop?

2.4.3 Reverse

Create a program that accepts N integers and rewrite the integers in reverse.

Example

```
Input N: 5
5
2
1
6
3
Reversed:
```

```
3 6 1 2 5
```

2.4.4 Anagram

Given two arrays A and B. Create a program that will check whether B is an anagram of A or not. Assume that each element of A and B will always a whole number [1,10].

Array B is an anagram of array A if the elements of A can be rearranged into array B.

Hint: make a frequency array, an array whose first element is how many 1s exist in the array, second element is how many number 2s exist in the array, and so on.

Example 1

```
Input number of elements in A: 3
Input element A number 1: 8
Input element A number 2: 4
Input element A number 3: 3
Input number of elements in B: 3
Input element B number 1: 4
Input element B number 2: 8
Input element B number 3: 3
B is anagram of A
```

Example 2

```
Input number of elements in A: 2
Input element A number 1: 1
Input element A number 2: 4
Input number of elements in B: 2
Input element B number 1: 1
Input element B number 2: 5
B is not anagram of A
```

2.4.5 Palindrome

Create a program that accepts a string and write whether a string is palindrome or not. Assume the string only consists of lowercase characters (a-z).

Example 1

```
Input string length: 3
Input string: eat
eat is not a palindrome
```

Example 2

```
Input string length: 3
Input string: mom
mom is a palindrome
```

Example 3

```
Input string length: 7
Input string: racecar
racecar is a palindrome
```

2.4.6 Practice Problem 1 Year 2021

Mr. Kil has several Lego pieces with dimensions $1 \times 1 \times 1$. Mr. Kil is tasked with making as few cubes as possible with the Lego pieces he has. The method is to make the largest cube possible with the available Lego, then with the remaining Lego, repeat the same process until there are no more Lego pieces left.

Determine how many cubes Mr. Kil can make.

Note: It is prohibited to use built-in functions to find the square root of a number. The use of built-in functions to find the square root of a number will result in a score of 0 for this problem.

Hint: Use a loop to find the nearest cubic value to the number of Lego pieces at any given time.

Example 1

```
Enter the number of Lego pieces: 36
Mr. Kil can make 3 cubes.
```

Explanation for Example 1:
36 = 27 + 8 + 1

Example 2

```
Enter the number of Lego pieces: 11
Mr. Kil can make 4 cubes.
```

Explanation for Example 2:
11 = 8 + 1 + 1 + 1

2.4.7 Practice Problem 2 Year 2021

Given a two-variable inequality as follows.

$$(\frac{x}{2})^2 + (\frac{5y}{4} - 2\sqrt{|x|})^2 \leq 120$$

In this problem, Mr. Dip asks you to implement a heart shape using * with the help of the given inequality on the range $-30 \leq x \leq 30$ and $-15 \leq y \leq 15$. Your answer is considered correct if the heart shape is clearly visible and not inverted or slanted.

Notes: Use the abs(x) function to turn the value of x into a positive value.

Example 1

```
     *************          *************
   *****************      *****************
  *******************  *******************
 *******************  *******************
 ********************************************
 ********************************************
 ********************************************
 ********************************************
  ******************************************
   ****************************************
    **************************************
      ********************************
       ******************************
         **************************
           ********************
             ****************
               ************
                  *********
                    *****
                     ***
```

2.4.8 Practice Problem 3 Year 2021

Create a program that accepts a natural number $N > 1$ and outputs its prime factors. The result must be sorted from the smallest to the largest prime factors.

Example 1

```
Enter N: 110
The prime factors are 2, 5, 11.
```

Example 2

```
Enter N: 160
The prime factors are 2, 5.
```

2.4.9 Practice Problem 1 Year 2022

Mr. Leo is given a list consisting of N elements. Help Mr. Leo to determine the number of sublist (a sublist is the result of selecting several consecutive indices in the list) that has an even sum.

Test Case 1

```
Enter the value of N: 5
Enter the number for index 1: 1
Enter the number for index 2: 2
Enter the number for index 3: 3
Enter the number for index 4: 4
```

```
Enter the number for index 5: 5
There are 6 sublists with an even sum.
```

Explanation for Example 1:
The sublists with an even number of elements are $[1,2,3]$, $[1,2,3,4]$, $[2]$, $[2,3,4,5]$, $[3,4,5]$, and $[4]$.

2.4.10 Practice Problem 2 Year 2022

Mr. Kil has an unsorted collection of numbers, determine how many sorted number groups can be formed from the array. Data is always distinct (different values).

The input data is in the range of 1-1000.

Example 1

```
Enter the number of data: 5
Enter data for index 1: 2
Enter data for index 2: 1
Enter data for index 3: 3
Enter data for index 4: 6
Enter data for index 5: 5
The number of sorted number groups is 2.
```

Explanation for Example 1:
Ordered groups: [1,3] and [5,6]

Example 2

```
Enter the number of data: 6
Enter data for index 1: 7
Enter data for index 2: 12
Enter data for index 3: 35
Enter data for index 4: 6
Enter data for index 5: 10
Enter data for index 6: 9
The number of sorted number groups is 4.
```

Explanation for Example 2:
Ordered groups: [6,7], [9,10], [12,12], [35,35]

3 Module 3 - Function and Procedure

3.1 Function

What is a function? Function is a part of a program that is capable of working certain tasks or operations outside the main program. Function will return value according to the algorithm given.

For example, the function to calculate squares is as follows:

```
def square(x):
    x2 = x * x
    return x
```

The above function is named `square` and returns the square value of `x`. That function accepts a parameter named `x`.

For example, the following is a complete program that receives input and outputs square of the input number.

```
def square(x):
    x2 = x * x
    return x2

n = int(input())
n2 = square(n)
print(n2)
```

Notice in the example above, the variable in the main program is named `n`. But in the function, the variable is named `x`. Even if the value of `x` changes, the value of `n` in the main program will not change.

A function can also accept more than one parameter. In addition, functions can also do things like ordinary programs, but cannot change variables in the main program. For example, here is a function that calculates the value of a^b.

```
def power(a, b):
    # assume b >= 0
    c = 1
    for i in range(b):
        c *= a
    return c
```

Note: You are not recommended to put arrays as function parameters, because there are some things that have not been taught at Computational Thinking.

3.2 Procedure

The actual procedure is the same as the function, but there is no return value. As for example, here is a program to write menus:

```
print("Menu:")
print("1. Burger")
print("2. Fried Chicken")
print("3. Instant Noodle")
print("Enter your choice: ")
food_option = int(input())

print("Menu:")
print("1. Avocado Juice")
print("2. Thai Tea")
print("3. Milk Tea")
print("Enter your choice: ")
beverage_option = int(input())

print("Menu:")
print("1. Fried fries")
print("2. Kerupuk")
print("3. Doughnut")
print("Enter your choice: ")
additional_option = int(input())
```

As written above, we need to write down the menu many times. We can summarize it to:

```
def write_menu(opt1, opt2, opt3):
    print("Menu:")
    print("1. " + str(opt1))
    print("2. " + str(opt2))
    print("3. " + str(opt3))
    print("Enter your choice: ")

tulis_menu("Burger", "Fried Chicken", "Instant Noodle")
food_option = int(input())

tulis_menu("Avocado uice", "Thai Tea", "Milk Tea")
beverage_option = int(input())

tulis_menu("Fried fries", "Kerupuk", "Doughnut")
additional_option = int(input())
```

3.3 Module 3 Exercise

3.3.1 Quadratic Function

You have a function defined as:

$$f(x) = x^2 - 2x + 5$$

Create a program that will accept A and B, then writes $f(A), f(A+1), \ldots, f(B)$!

Note: You have to write a function in your program that accepts a parameter x and returns the value of $f(x)$.

Example

```
Input A: 3
Input B: 6
f(3) = 8
f(4) = 13
f(5) = 20
f(6) = 29
```

3.3.2 Pascal Triangle

Pascal triangle can be made using a matrix. Matrix will be initialized by filling all the elements in the first row and the first column with 1. Then, every empty elements will be defined as the sum of element from previous row (but the same column) and the sum of element from previous column (but the same row). Make a program that accepts an integer N and writes a pascal triangle of size $N \times N$

Example

```
Input N: 4
1 1 1 1
1 2 3 4
1 3 6 10
1 4 10 20
```

3.3.3 Practice Problem 1 Year 2021

Mr. Ric and Mr. Dip often go to ITB together by motor vehicle. However, Mr. Ric feels that alternating the schedule for paying for gasoline only disadvantages him. So Mr. Dip proposed a way to divide the payment schedule. For every 2^n, they will change the pattern by swapping the order (complement) of the payment in the pattern $2^{(n-1)}$. Here is the pattern proposed by Dip:

n	Pattern
0	**A**
1	A**R**
2	AR**RA**
3	ARRA**RAAR**

As can be seen, the text in bold is the opposite of the non-bold part of a pattern. Mr. Ric really likes this idea. Help Mr. Ric create a program to generate this pattern.

Use Functions or Procedures in solving this problem. Clue: The sequence is the addition of complements from the previous sequence as shown in the example. Prepare an array of characters with a size of 2 times the previous sequence.

Example 1

```
Enter character 1: A
Enter character 2: R
Enter the value of n: 2
The obtained pattern: ARRA
```

Example 2

```
Enter character 1: U
Enter character 2: W
Enter the value of n: 4
The obtained pattern: UWWUWUUWWUUWUWWU
```