CSX3001/ITX3001
CS1201 COMPUTER PROGRAMMING 1

CLASS 02 SEQUENTIAL STRUCTURE

BASIC CODING, EXPRESSION EVALUATION, INPUTTING AND OUTPUTTING DATA, DATA TYPE, AND DATA CONVERSION

PYTHON

Variable in any programming language is like a container, which stores a value of different types. For example,

```
str = "Hello Python"
```

The line `str = "Hello Python"` in the above code creates a variable named `str` and assigns a value of `"Hello Python"` (`This is a string type`). Basically, the assignment operator (=) assigns a value to a variable. Once again, a variable is like a container; you can use it to hold almost anything – a simple number, a string of letters, or Boolean.

The variable name is a named location used to store data in the memory (RAM). You can think of a variable as a box to store toys in it and those toys can be replaced at any time. Let's see an example below.

```
number = 10
print(number)

# assign a new value to number
number = 15.5
print(number)
```

Initially, the value contained in a variable *`number`* was 10. Later it's changed to 15.5. So, when you run the program, the output will be:

```
10
15.5
```

I think you might have a question about why I give a red color on the third line. The **#** symbol uses for writing a comment. The comments are notes that are ignored by the computer and are there to help you reading the code to understand what's going on. You just put hashtag (#) in front of the text you want to comment.

You can name a variable almost anything you want, with a few exceptions. You can't name them something that **is already a pre-defined or reserved word in python**. Below are an example of pre-defined keywords in Python which are not allowed to be used as variable name.

```
False, class, finally, is, return, None, continue, for,
lambda, try, True, def, from, nonlocal, while, and, del,
global, not, with, as, elif, if, or, yield, assert, else,
import, pass, break, except, in, raise
```

You also can't have **two variables with the same name in the same block of code**. Actually, you can have the same name for two variables in Python, but the value will be based on the latest assignment.

PYTHON DATA TYPES

There are various data types in Python. These are the basic types that you must know... just for now.

Numeric – Integer and floating-point numbers. They are defined as int and float, respectively. For instance,

```
number = 12 # this is an example of int
decimalNumber = 12.543243 # this is an example of float
```

Now let's do some exercises to get to know more about Python. Create a Python file (.py) as instructed in each exercise. Note that you can also use a Jupyter notebook to complete all exercises. However, may sure that you are able to create individual .py file for each exercise.

NUMBERIC EXERCISE

1. Create a new Python file and name it "class02-number.py". Try the following code and answer the following question. (You may use a Jupyter notebook to do

```
x = 24
print(x)
y = 1.253321
print(y)
```

1.1 What is a data type of a variable x? ....., and a variable y .....

EXPRESSION EVALUATION EXERCISE

2. Create a new Python file and name it “class02-expression1.py”. Try the following code and answer the following question.

```
x = 5 + 2 * 3
y = 100 * 20 - 5 / 100 + 0.05
```

2.1 What is a data type of variable x? ..... and its value .....
2.2 What is a data type of variable y? ..... and its value .....
2.3 Rewrite the first-line and second-line expression with proper order of operations using parentheses.

x = ...
y = ...

3. From question 2, you have learned how the mathematic operators work and their orders (**operator precedence**). High precedence means that operator will be calculated first. Specify the precedence order for the following operators with given numbers. *Note that, one or more operators can have the same precedence order.* 0 is the lowest.

Operator	÷	-	x	+	( )
Precedence Order				0	

When two operators share the same precedence order. For instance, plus and minus. We will calculate the expression with left-to-right calculation. Consider the following expression.

```
3 - 5 + 7
```

The expression will be calculated from 3 - 5, then + 7. The rule which is applied to control this kind of calculation when two operators have the same precedence order is called “**operator associativity**” Most of the operator associativity of mathematic operator is left-to-right associativity.

4. There is one more basic mathematic operator that you must know. Create a new Python file and name it "class02-expression2.py". Try the following code and answer the following questions.

```
w = 12 % 4
print(w)
x = 10 % 7
print(x)
y = x % (w + 1)
print(y)
z = z % 2
print(z)
```

4.1 What is a data type of variable w? ..... and its value .....
4.2 What is a data type of variable x? ..... and its value .....
4.3 What is a data type of variable y? ..... and its value .....
4.4 What is a data type of variable z? ..... and its value .....

The % operator is called *modulo operator*, which might not be as familiar to you. The modulo operator (also called *modulus*) gives the remainder after division. You will get more used to it from the assignment.

Another category of operators that might be useful is the **compound assignment operators**. These are "shortcut" operators that combine a mathematic operator with the assignment operator (=). For example, this expression below

```
a = a + b
```

becomes

```
a += b
```

These operators will update the value of a variable by performing an operation on it. In plain English, an expression like a += b says "add b to a and store the new value in a." Let's see example s for other mathematic operations.

Short form	Long form
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b

5. Let's see the += operator in action. Imagine that we're trying to write a program to calculate the number of animals on an ark.

```
animalsOnArk = 0

numberOfGiraffes = 4
animalsOnArk += numberOfGiraffes

numberOfElephants = 2
animalsOnArk += numberOfElephants

numberOfAntelope = 8
animalsOnArk += numberOfAntelope
```

5.1 After four giraffes, two elephants, and eight antelope board the ark, the final value of animalsOnArk is.....

String is a sequence of Unicode characters. We can use a single quote, double quotes, or triple quotes to represent strings. For instance,

```
message = 'single quotes'
message = "string with double quotes"
message = """This is a
             multiple line string content
             with triple quotes"""
```

STRING EXERCISE

6. Create a new Python file and name it "class02-string1.py". Write a Python code which prints the following output.

```
My name is Ada Lovelace.
I was born in December 10, 1815.
I'm the world's first computer programmer.
```

6.1 Try to complete the with three String variables.

6.2 Try to complete with one String variable.

You can also connect one String and another String together using + symbol. This process called **String concatenation**. See the examples,

```
message = "Hello"
name = "Ada"
result = message + " " + name
```

At the third line shows how the String concatenation works. The value of *result* variable will be

```
"Hello Ada"
```

Actually, there is another similar way to do String concatenation and We think you would love it. Let's take a look at the following code.

```
message = "Hello"
name = "Ada"
result = f"{message} {name}"
```

Python provides you an easier way to format your String called *f-String*. In general, this way is called (instead of String concatenation) String interpolation. The *f-String* works either start with lowercase "f" or uppercase "F". In order to interpolate any variable values, you have to refer the variable inside { }.

7. Create a new Python file and name it "class02-string2.py". Try the following code and answer the following questions.

```
name = "Ada"
surname = "Lovelace"
position = "Programmer"
country = "UK"

firstSentence = f"My name is {name} {surname}."
secondSentence = f"My country is {country}."
thirdSentence = F"My position is {position}."

forthSentence = f"{firstSentence} {secondSentence} "
forthSentence += f"{thirdSentence}"
# += is a compound assignment operator
```

7.1 What is the value of firstSentence?

.....

7.2 What is the value of secondSentence?

.....

7.3 What is the value of thirdSentence?

.....

7.4 What is the value of finalSentence?

.....

7.5 Rewrite the given code in “class02-string3.py” without f-String.

8. Create a new Python file and name it “class02-string4.py”. Try the following code and answer the following questions.

```
print("Name:\t\tAda")
print("Surname:\t\tLovelace")
print("Programmer\nin UK")
```

8.1 What dose \t do?

.....

8.2 What dose \n do?

.....

\t and \n are two of the Escape Sequences (there are more) which are predefined for String decoration.

INPUT/OUTPUT EXERCISE

What if we want a user to enter some number and we do calculate something on it? For accepting input from the user, we have to write this code

```
input() # accept input from the user but store nowhere
```

or

```
x = input() # accept input from the user, store to x
```

The default type of the input will always be String. If you want it to be other type, you can cast the type like this

```
x = input() # x's type is String
x = int(x) # x's type is Int
x = float(x) # x's type is Float
```

Be reminded that the input x must be only number. If x is English words, the program will crash. Let’s try the above code in Python and enter Hello as an input.

We can also specify the guideline message for the user. For instance,

```
x = input("Enter your name: ")
```

will run as

```
Enter your name:
```

the user will then know what they need to enter to the program.

9. Write a code to take two inputs (your ID and your name) and store in two variables namely, *myID* and *myName*. Variables *myID* and my*Name* must be integer and string, respectively. Print the entered ID and name on the screen using **f-string format**. Expected output as follow when the entered ID is 6210001 and entered name is Ada.

```
Enter your ID: 6210001
Enter your name: Ada
My name is Ada and my ID is 6210001
```

10. Write a code to take two inputs and store them in two string variables, namely *firstName* and *lastName*, respectively. Connect the two strings with an empty String (" ") in between and store them in a variable *fullName*. Then print *fullName*'s string value on the screen (using **f-string format**). The expected input/output as follow.

```
Enter your firstName: Ada
Enter your lastName: Lovelace
Welcome Ada Lovelace!!!!
```

11. Extend a code in exercise 10 to take an additional input and store it in an integer variable namely, *age*. Based on entered age, calculate the year of birth and produce the following output.

```
Enter your firstName: Ada
Enter your lastName: Lovelace
Enter your age: 19

Welcome Ada Lovelace!!!! You were born in year 2001.
```

Boolean in Python is more like a two-constant value; True and False. They are used to represent truth values. For instance,

```
isOpen = True
isPossible = False
```

The common type of Boolean expression is one that compares two values using a *comparison operator*. There are six comparison operators.

Symbol	Definition
==	Is equal to
!=	Is not equal to
>	Is greater than
<	Is less than
>=	Is greater than or equal to
<=	Is less than or equal to

Considering the following code and each line's value on the right column.

```
3 + 2 == 5                  True
4 + 5 == 8                  False
3 != 5                      True
4 + 5 != 8                  True

name = "Ada"                "Ada"
name == "Steve"             False
name == "Ada"               True
name == "ada"               False

myHeight = 1.69             1.69
myHeight == 1.69            True

9 > 7                       True
9 < 11                      True

3 + 4 >= 7                  True
3 + 4 > 7                   False


5 + 6 <= 11                 True
5 + 6 < 11                  False
```

In addition, Boolean expressions can be joined together using compound Boolean expressions. It's a lot like making compound sentences in English with the words ***and*** and ***or***. In programming, there is a third case: ***not*** and we call these words ***logical operators***.

Symbol	Definition
and	Logical AND
or	Logical OR
not	Logical NOT

Considering the following code and each line's value on the right column.

```
age = 12                                          12
age > 10 and age < 15                             True

age = 18                                          18
age > 10 and age < 15                             False

name = "Ada"                                      "Ada"
name == "Adah"                                    False
name == "Adah" or name == "Ada"                   True

isAGirl = False                                   False
not isAGirl and name == "Ada"                     True
isAGirl and name == "Adah"                        False
```

BOOLEAN EXERCISE

12. What is the Boolean value of the following expression if A = **true**, B = **false**, C = **true**, and D = **false**?

    12.1 A and B or not (A or D) and C

    .....

    12.2 ((A and not A and not B) or not (not C or D))

    .....

    12.3 (A or B or C or D) and B

    .....

    12.4 Not C and True or B

    .....

    12.5 not (not (not D)))

    .....

12.6 B or A and not C

.....

12.7 A and ((B and not D) or (B or C))

.....

[extra] PYTHON STATEMENT

Multi-line statement can be written as multiple lines with the line continuation character, backward slash (\) as follow:

```
x = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

You can also use the parentheses ( ) for line continuation as follow:

```
x = (1 + 2 + 3 +
     4 + 5 + 6 +
     7 + 8 + 9)
```

If you want to write multiple statements in one-line, you can use semicolons for separated each statement as follow:

```
x = 1; y = 2; z = 3
```

For the assignments, you must submit via a teaching and learning channel (MS Teams, Google Classroom, etc.) as specified by your lecturer. Weekly assignment is part of your course assessment.

For each assignment, a Python file will be named as

"yourid_name_section_class02_sem_year_ASMno.py".

For example, a file name for assignment no. 1 is

"u6310001_Harry_541_class02_1_2020_ASM1.py"

A filename for assignment no. 2 is "u6310001_Harry_541_class02_1_2020_ASM2.py"

ASSIGNMENTS

1. Write a Python code to take **two integer inputs** (months and days) and store them in two variables namely, *month* and *day*. The program does a calculation and prints a total number of remaining days in a year. Assume that there are 30 days per month and 360 days per year.
   Expected inputs/outputs are as follow.
   Sample 1
   = 360 – (month * 30) - day

```
Enter months: 2
Enter days: 20
The remaining days is 280 days
```

   Sample 2

```
Enter months: 5
Enter days: 10
The total number of days is 200 days
```

2. Write a code to take one integer input and store in variable namely, *number*. The entered number has to be 3-digit number. The program prints out the new number which the first and last number are swapped as well as a sum of all digits. Expected inputs/outputs are as follow. **(Hint, a modulo operator is a key)**
   Sample 1

```
Enter 3-digit number: 103
The new number is 301.
The sum value is 4.
```

   Sample 2

```
Enter 3-digit number: 453
The new number is 354
The sum value is 12.
```

Below code is a simple example of if-else in Python

```
if age >= 20:
   print("You are eligible to drink alcohol.")
else:
   print("You can only drink milk, my son.")

#If a value stored a variable age is equal to or higher
#than 20, the code prints "You are eligible to drink
#alcohol."

#Otherwise, it prints another statement.
```

3. Write a Python code that takes a remaining gas in a tank (in litre), a car's consumption rate (km/litre), and a distance you want to travel (in kilometer). The variable names are *gas* (for a remaining gas), *rate* (for a consumption rate), and *distance* (for a travelling distance). The code prints "You need xxx more litres to reach your destination", in a case that a gas is not enough. Otherwise the code prints "You can reach a destination and you still have xxx litres in the tank".

   Expected inputs/outputs as follows:
   Sample 1

```
#inputs
Enter a remaining gas (litre): 30
Enter a consumption rate (km/litre): 20
Enter a traveling distance: 500

#output
You can reach a destination and still have 5 litres in the
tank.
```

   Sample 2

```
#inputs
Enter a remaining gas(litre): 20
Enter a consumption rate (km/litre): 20
Enter a traveling distance: 600

#output
You need 10 more litres to reach your destination.
```