FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES AND MULTIMEDIA ARTS

IT0011
(INTEGRATIVE PROGRAMMING TECHNOLOGIES)

EXERCISE

2

PYTHON FUNDAMENTALS

Student Name / Group Name:	Carl Allen Torno
Members (if Group):	**Name** / **Role**
Section:	TW21
Professor:	Dr. Jay-Ar Lalata

I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE

- Analyze a problem and identify and define the computing requirements appropriate to its solution. [PO: B]

II. COURSE LEARNING OUTCOME/S (CLO)ADDRESSED BY THE LABORATORY EXERCISE

- Able to write program applying the fundamental syntax in Python. Solve complex condition problems using control statements, identify errors, and use techniques to optimize the program [CLO: 1]

III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE

At the end of this exercise, students must be able to:

- Program applying the fundamental syntax in Python.
- Solve condition complex problems.
- Use techniques to optimize the program.

IV. BACKGROUND INFORMATION

Variable Assignment

Think of a variable as a name attached to a particular object. In Python, variables need not be declared or defined in advance, as is the case in many other programming languages. To create a variable, you just assign it a value and then start using it. Assignment is done with a single equals sign (=):

Python >>>

```
>>> n = 300
```

This is read or interpreted as "`n` is assigned the value `300`." Once this is done, `n` can be used in a statement or expression, and its value will be substituted:

Python >>>

```
>>> print(n)
300
```

Python also allows chained assignment, which makes it possible to assign the same value to several variables simultaneously:

```
Python >>>
>>> a = b = c = 300
>>> print(a, b, c)
300 300 300
```

The chained assignment above assigns `300` to the variables `a`, `b`, and `c` simultaneously.

Variable Types in Python

In many programming languages, variables are statically typed. That means a variable is initially declared to have a specific data type, and any value assigned to it during its lifetime must always have that type.

Variables in Python are not subject to this restriction. In Python, a variable may be assigned a value of one type and then later re-assigned a value of a different type:

```
Python >>>
>>> var = 23.5
>>> print(var)
23.5

>>> var = "Now I'm a string"
>>> print(var)
Now I'm a string
```

Do the following activity and answer the problem questions:

Assignments

One of the building blocks of programming is associating a name to a value. This is called assignment. The associated name is usually called a *variable*.

```
>>> x = 4
>>> x * x
16
```

In this example `x` is a variable and it's value is `4`.

If you try to use a name that is not associated with any value, python gives an error message.

```
>>> foo
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'foo' is not defined
>>> foo = 4
>>> foo
4
```

If you re-assign a different value to an existing variable, the new value overwrites the old value.

```
>>> x = 4
>>> x
4
>>> x = 'hello'
>>> x
'hello'
```

It is possible to do multiple assignments at once.

```
>>> a, b = 1, 2
>>> a
1
>>> b
2
>>> a + b
3
```

Swapping values of 2 variables in python is very simple.

```
>>> a, b = 1, 2
>>> a, b = b, a
>>> a
2
>>> b
1
```

When executing assignments, python evaluates the right hand side first and then assigns those values to the variables specified in the left hand side.

Problem 4: What will be output of the following program.

```
x = 4
y = x + 1
x = 2
print x, y
```

Problem 5: What will be the output of the following program.

```
x, y = 2, 6
x, y = y, x + 2
print x, y
```

Problem 6: What will be the output of the following program.

```
a, b = 2, 3
c, b = a, c + 1
print a, b, c
```

Numbers

We already know how to work with numbers.

```
>>> 42
42
>>> 4 + 2
6
```

Python also supports decimal numbers.

```
>>> 4.2
4.2
>>> 4.2 + 2.3
6.5
```

Python supports the following operators on numbers.

- + addition
- − subtraction
- * multiplication
- / division
- ** exponent
- % remainder

Let's try them on integers.

```
>>> 7 + 2
9
>>> 7 - 2
5
>>> 7 * 2
14
>>> 7 / 2
3
>>> 7 ** 2
49
>>> 7 % 2
1
```

If you notice, the result `7 / 2` is `3` not `3.5`. It is because the `/` operator when working on integers, produces only an integer. Lets see what happens when we try it with decimal numbers:

```
>>> 7.0 / 2.0
3.5
>>> 7.0 / 2
3.5
>>> 7 / 2.0
3.5
```

The operators can be combined.

```
>>> 7 + 2 + 5 - 3
11
>>> 2 * 3 + 4
10
```

It is important to understand how these compound expressions are evaluated. The operators have precedence, a kind of priority that determines which operator is applied first. Among the numerical operators, the precedence of operators is as follows, from low precedence to high.

- `+`, `-`
- `*`, `/`, `%`
- `**`

When we compute `2 + 3 * 4`, `3 * 4` is computed first as the precedence of `*` is higher than `+` and then the result is added to 2.

```
>>> 2 + 3 * 4
14
```

We can use parenthesis to specify the explicit groups.

```
>>> (2 + 3) * 4
20
```

All the operators except `**` are left-associcate, that means that the application of the operators starts from left to right.

```
1 + 2 + 3 * 4 + 5
  ↓
  3   + 3 * 4 + 5
        ↓
  3   +   12  + 5
      ↓
      15      + 5
              ↓
             20
```

V. LABORATORY ACTIVITY

INSTRUCTIONS:
Read and analyze the problems then create a Python program that satisfies the requirement of each problem.

TASK 1: Zodiac Sign

ACTIVITY 2.1

Write a Python program that will read in month and day (as numerical value). The program will return the equivalent zodiac sign. Use if statement. Note: You are not allowed to use user-defined functions.

Expected Output:
If the inputs are 3 and 1 which means March 1, it should output, "Pisces."

Copy the source code directly from the IDE so that colors of the codes are preserved for readability. Snip the output as seen in the sample and paste it below.

Source Code:

```
month = int(input("Enter Month: "))
day = int(input("Enter Day: "))

months = {
    1:'January',
    2:'February',
    3:'March',
    4:'April',
    5:'May',
    6:'June',
    7:'July',
    8:'August',
    9:'September',
    10:'October',
    11:'November',
    12:'December',
}

if month <= 0 and day <= 0:
    print("[INVALID MONTH AND DAY] No Negative or Zero Inputs")
elif day <= 0:
    print("[INVALID DAY] No Negative or Zero Inputs")
elif month <= 0 or month >= 13:
```

```
    print("[INVALID MONTH]")

elif month == 1:
    if(day <= 19):
        print(f"{months[month]} {day} is Capricorn")
    elif(day <= 31):
        print(f"{months[month]} {day} is Aquarius")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 2:
    if(day <= 18):
        print(f"{months[month]} {day} is Aquarius")
    elif(day <= 29):
        print(f"{months[month]} {day} is Pisces")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 3:
    if(day <= 20):
        print(f"{months[month]} {day} is Pisces")
    elif(day <= 31):
        print(f"{months[month]} {day} is Aries")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 4:
    if(day <= 19):
        print(f"{months[month]} {day} is Aries")
    elif(day <= 30):
        print(f"{months[month]} {day} is Taurus")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 5:
    if(day <= 20):
        print(f"{months[month]} {day} is Taurus")
    elif(day <= 31):
        print(f"{months[month]} {day} is Gemini")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 6:
```

```
    if(day <= 20):
        print(f"{months[month]} {day} is Gemini")
    elif(day <= 30):
        print(f"{months[month]} {day} is Cancer")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 7:
    if(day <= 22):
        print(f"{months[month]} {day} is Cancer")
    elif(day <= 31):
        print(f"{months[month]} {day} is Leo")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 8:
    if(day <= 22):
        print(f"{months[month]} {day} is Leo")
    elif(day <= 31):
        print(f"{months[month]} {day} is Virgo")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 9:
    if(day <= 22):
        print(f"{months[month]} {day} is Virgo")
    elif(day <= 30):
        print(f"{months[month]} {day} is Libra")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 10:
    if(day <= 22):
        print(f"{months[month]} {day} is Libra")
    elif(day <= 31):
        print(f"{months[month]} {day} is Scorpio")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 11:
    if(day <= 21):
        print(f"{months[month]} {day} is Scorpio")
    elif(day <= 30):
```

```
        print(f"{months[month]} {day} is Sagittarius")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")

elif month == 12:
    if(day <= 22):
        print(f"{months[month]} {day} is Sagittarius")
    elif(day <= 31):
        print(f"{months[month]} {day} is Capricorn")
    else:
        print(f"[INVALID DAY] There is no {day} in {months[month]}")
```

Output: (Use three sets of inputs and screenshot all the outputs and paste them here)

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/Zodiac.py
Enter Month: 10
Enter Day: 30
October 30 is Scorpio
PS C:\Users\202210168\Documents\Python>
```

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/Zodiac.py
Enter Month: 6
Enter Day: 23
June 23 is Cancer
PS C:\Users\202210168\Documents\Python>
```

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/Zodiac.py
Enter Month: 8
Enter Day: 17
August 17 is Leo
PS C:\Users\202210168\Documents\Python>
```

TASK 2: Number Increase

ACTIVITY 2.2

Using for-nested loop, write a Python program that asks for a positive integer number. The program should display an increasing series of numbers from the first row until the last row. If the user entered an invalid input such as 0 or a negative number, display an error message. Note: You are not allowed to use user-defined functions.

Sample Output:
Enter a number: 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

Source Code:

```
row = int(input('Enter a number: '))

if row <= 0:
    print("[INVALID] Enter a Positive Integer!!")
else:
    for i in range(1, row+1):
        for j in range(i):
            print(j + 1, end = " ")
        print("")
```

Output: (Use three sets of inputs and screenshot all the outputs and paste them here)

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/Patterns.py
Enter a number: 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/Patterns.py
Enter a number: 10
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
PS C:\Users\202210168\Documents\Python>
```

TASK 3: Count the Vowels

ACTIVITY 2.3

Write a python program that accepts a line of text and prints the number of characters, number of vowels, number of blank spaces, and number of commas and period in it. Note: You are not allowed to use user-defined functions.

If the input string is "I love you, Jay-ar."

Output:
Number of characters: 13
Number of vowels: 7
Number of spaces: 3
Number of commas: 1
Number of periods: 1

Source Code:

```
text = input("Enter a word: ")

# Setting a list of vowels for basis
vowels = ['a','e','i','o','u']

vowelCount = 0
spaces = 0
totalChar = 0
for i in text:
    # Counter for spaces
    if(' ' in i):
        spaces += 1

    # For counting ALPHABETS AND NUMBERS
    if(i.isalnum()):
        totalChar += 1

    # Counting Vowels
    if(i.lower() in vowels):
        vowelCount += 1

print(f"\nNumber of characters: {totalChar}")
print(f"Number of vowels: {vowelCount}")
print(f"Number of spaces: {spaces}")
print(f"Number of commas: {text.count(',')}")
print(f"Number of periods: {text.count('.')}")
```

Output: (Use three sets of inputs and screenshot all the outputs and paste them here)

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/StringCounter.py
Enter a word: It is gonna be a LEGEND.. wait for it -DARY!

Number of characters: 31
Number of vowels: 13
Number of spaces: 9
Number of commas: 0
Number of periods: 2
PS C:\Users\202210168\Documents\Python>
```

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/StringCounter.py
Enter a word: I love Leyla, Since-2024

Number of characters: 15
Number of vowels: 7
Number of spaces: 3
Number of commas: 1
Number of periods: 0
PS C:\Users\202210168\Documents\Python>
```

```
PS C:\Users\202210168\Documents\Python> & "C:/Program Files/Python311/python.exe" c:/Users/202210168/Documents/Python/StringCounter.py
Enter a word: My birthday is 10-30-2003..

Number of characters: 20
Number of vowels: 3
Number of spaces: 3
Number of commas: 0
Number of periods: 2
PS C:\Users\202210168\Documents\Python>
```

VI. QUESTION AND ANSWER

Briefly answer the questions below.

• Describe how data type works in Python. ➢ Data types in python doesn't need to be declare it automatically detects the assignation to the variable name making python easy for beginners.
• What is the importance of indentation in control statements? ➢ It shows the relationship between the control of the statements where the statement is in the control structure or it is outside of it meaning if it is indented it is in the control structure.
• Why looping is significant in solving computational problems? ➢ So it reduces repetition of computation like you need to add repeating solutions in short they allow the repetition of a set of instructions. Making the program short and fast.

VII. GRADING SYSTEM/ RUBRIC

Trait	(Excellent)	(Good)	(Fair)	(Poor)
Correct Identification of Inputs and Outputs(30pts)	Able to identify correctly all input and output and provide alternative. **(30-26pts)**	Able to identify correctly all input and output **(25-17pts)**	Able to identify only one input or output **(16-9pts)**	Unable to identify any input and output **(8-1pts)**
Requirement Specification(40pts)	The program works and meets all specifications. Does exception al checking for errors and out-of- range data **(40-31pts)**	The program works and meets all specifications. Does some checking for errors and out of range data **(30-21pts)**	The program produces correct results but does not display correctly Does not check for errors and out of range data **(20-11pts)**	The program produce s incorrect results **(10-1pts)**
Free from syntax, logic, and runtime errors (30pts)	Unable to run program **(30-26pts)**	Able to run program but have logic error **(25-17pts)**	Able to run program correctly without any logic error and display inappropriate output **(16-9pts)**	Able to run program correctly without any logic error and display appropriate output **(8-1pts)**

VIII. REFERENCES

- Muller, A. C. & Guido, S. (2017). Introduction to Machine Learning with Python. O'Reilly Media, Inc
- Eric Matthes (2016). Python Crash Course. No Starch Press, Inc.
- http://onefeu.instructure.com