WHAT IS PYTHON?

➢ Python is an interpreted, high-level programming language used for development of various systems and programs.

➢ Python’s simplicity and readability make it the most popular programming language!
VARIABLE ASSIGNMENT

➢ Programs operate dynamically. Therefore, like a mathematical problem, Python and other programming languages use variables to store values dynamically.

➢ Variable can store values of different data types from ‘integer’ (number) to ‘string’ (word) to ‘list’ of values.

```python
int_var = 10  # int
float_var = 1.23  # float
string_var = "Hello"  # string
string_var2 = "Bye"  # string
boolean_var = True  # boolean (True or false)
list_var = [1, 2, 3, 4]  # list
list_var2 = ["python", 10]  # list
```

➢ Variable names must begin with a letter and may be followed by letters, digits, and underscore (_).

<table>
<thead>
<tr>
<th>Valid variable name</th>
<th>Invalid variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid_name</td>
<td>&amp;invalid_name</td>
</tr>
<tr>
<td>validName</td>
<td>1invalid-name</td>
</tr>
</tbody>
</table>

➢ ‘Variable assignment’ is a statement that assigns a value to a variable.

```python
year = 2017  # valid variable assignment
age, major, student = 21, "compSci", True  # valid variable assignment
next_election = 2016 + 4  # valid variable assignment
```

➢ Variables can be modified.

```python
age = 18  # initializing a variable
age = age + 3  # adding 3 to its initial value; age is now 21
age = age - 1  # subtracting 1 from modified value; age is now 20
```
If-else is a *conditional statement* that runs different statements depending on whether an expression (or condition) is true or false.

Algorithm of If-else statement:

![Algorithm diagram]

- Relational operators and Boolean results:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 4</td>
<td>True</td>
</tr>
<tr>
<td>3 &lt;= 4</td>
<td>True</td>
</tr>
<tr>
<td>3 == 4</td>
<td>False</td>
</tr>
<tr>
<td>3 != 4</td>
<td>True</td>
</tr>
<tr>
<td>3 &gt; 4</td>
<td>False</td>
</tr>
<tr>
<td>4 &gt;= 4</td>
<td>True</td>
</tr>
</tbody>
</table>

- General form of ‘If’ statement:

  ```python
  if <Boolean expression>:
      statement(s)
  ```

  where ‘if’ is an if statement keyword and `<Boolean expression>` is an expression used as conditional statement. **NOTICE** how there is an indentation in front of the statement. This indentation distinguishes whether the statement is part of the if statement or not.

  You can indent by pressing ‘Tab’ button *once* or ‘Space bar’ 4 *times.*

- Example:
‘num’ is an int variable and the conditional statement checks if ‘num’ – 10 is greater than 0 (which is the same as checking if ‘num’ is greater than 10). If it is true, the print statement is run. Since ‘num’ is initialized to 11 and it is indeed greater than 10, the output statement is “11 is greater than 10”.

➢ General form of ‘If-else’ statement:

```
if <Boolean expression>:
    statement(s)
else:
    statement(s)
```

where ‘if’ is an if statement keyword and <Boolean expression> is an expression used as conditional statement. ‘else’ is also a keyword and the statement following ‘else’ keyword is run only when the conditional statement is false.

➢ Example:

```
age = 18  # int variable
if age >= 21:  # Expression
    print("Party time!")
else:  # when condition is false
    print("Python time!")
```

‘age’ is an int variable and the conditional statement checks if age is greater than or equal to 21. If it is true, the statement “Party time!” is printed and if it is false, the statement “Python time!” is printed. Since age is 18, the conditional statement is false, and therefore the output statement is “Python time!”.
FOR (EACH) LOOP

➢ For (each) loop is conveniently used in cases where a programmer needs to repeat a block of code fixed number of times.

➢ Algorithm of For (each) loop:

➢ General form of for loop:

```
for item in list:
    statement(s)
```

where ‘for’ is a for loop keyword and ‘item’ and ‘list’ are variables. Statements are set of codes that you want your for loop to iterate through. 

**NOTICE** how there is an indentation in front of the statement. This indentation distinguishes whether the statement is part of the for loop or not.

You can indent by pressing ‘Tab’ button once or ‘Space bar’ 4 times.

➢ Examples:

*Example 1)*

```python
week = ['mon', 'tue', 'wed', 'thu']  # list
for day in week:
    print(day)
```

‘week’ is a list variable that contains items: “mon”, “tue”, “wed”, and “thu”. ‘day’ is a variable that represents each item in the list.

The list has 4 items, so for loop will iterate the statement 4 times.

Since the statement asks to print ‘day’ variable, each item in the list will be
printed (in order) as the for loop iterates.
Therefore, the output is:

<table>
<thead>
<tr>
<th>mon</th>
<th>tue</th>
<th>wed</th>
<th>thu</th>
</tr>
</thead>
</table>

**Example 2)**

```python
week = ['mon', 'tue', 'wed', 'thu']  # list
num = 1
for day in week:
    num = num + 1
    print(num)
```

Now, there is another variable named ‘num’, which is initialized to 1. Inside the for loop, it is asked to increment (add 1) the variable and print it. For loop is repeating the statements 4 times since it is still iterating the same list. ‘num’ is incremented 4 times and after each increment, it is printed.
Therefore, the output is:

| 2 | 3 | 4 | 5 |

**Example 3)**

```python
testScores = [90, 75, 92, 91, 100]  # list
for score in testScores:
    if(score > 90):
        print(score)
```

‘testScores’ is our new list and it contains 5 items.
‘score’ is a variable that represents each item in the list.
Within the for loop, we have ‘if’ condition and ‘print’ statement.
**NOTICE** the additional indentation before ‘print’ statement. This means that the ‘print’ statement is part of the ‘if’ condition and that the statement is run only when the condition is met. In each iteration, only the item that is greater than 90 will be printed. Therefore, the output is:

| 92 | 91 | 100 |