

## Python Programming Summary Lesson

Course :Blockchain  
Lecturing by :Sek Socheat  
Class : C7/10 Afternoon  
Student's name : Orn Sivechorng

1. **Variables:** (អញ្ជាតិ) is a named storage location that holds a value in a program. It allows data to be stored and used or changed during program execution.

Key features of a variable:

- **Name:** identifies the variable.
- **Value:** the data stored in the variable.
- **Type:** identifies the kind of data stored.
- **Scope:** determines where the variable can be accessed.
- **Mutability:** indicates whether the value can be changed.

Common Python Variable Types:

1. **Integer (int):** whole numbers.  
Example: `x = 10`
2. **Float (float):** numbers with decimal points.  
Example: `y = 3.14`
3. **String (str):** text or a sequence of characters inside quotes.  
Example: `name = "Sok"`
4. **Boolean (bool):** represents True or False.  
Example: `is_valid = True`
5. **List (list):** an ordered and mutable collection that can contain multiple values.  
Example: `numbers = [1, 2, 3]`
6. **Tuple (tuple):** an ordered and immutable collection.  
Example: `coordinates = (4, 5)`
7. **Dictionary (dict):** stores data as key-value pairs.  
Example: `person = {"name": "Alice", "age": 25}`
8. **Set (set):** an unordered collection of unique values.  
Example: `unique_numbers = {1, 2, 3}`
9. **NoneType (None):** represents the absence of a value.  
Example: `result = None`

Important:

- Basic variables store one value at a time, such as int, float, string, Boolean or None.
- Data structures can store multiple values. The main data structures are list, tuple, dictionary and set.

**2. BASIC OPERATIONS:** are fundamental actions performed on variables and values.

### 2.1 Arithmetic Operations

- (+) Addition:  $10 + 5 = 15$
- (-) Subtraction:  $20 - 8 = 12$
- (\*) Multiplication:  $6 * 3 = 18$
- (/) Division:  $15 / 3 = 5.0$
- (//) Floor division:  $17 // 5 = 3$
- (%) Modulus:  $17 \% 5 = 2$  (remainder)
- (\*\*) Exponentiation:  $2 ** 3 = 8$

### 2.2 Assignment Operations

- (=) Assignment : assigns a value.  
Example:  $x = 10$
- +=, -=, \*=, /= perform an operation and update the variable.

**2.3 Comparison Operations:** Comparison operators compare values and return True or False.

- (==) Equality: Equal to
- (!=) Inequality: equal to
- (>) Greater than
- (<) Less than
- (>=) Greater Than or Equal To
- (<=) Less than or equal to

### 2.4 Logical Operations

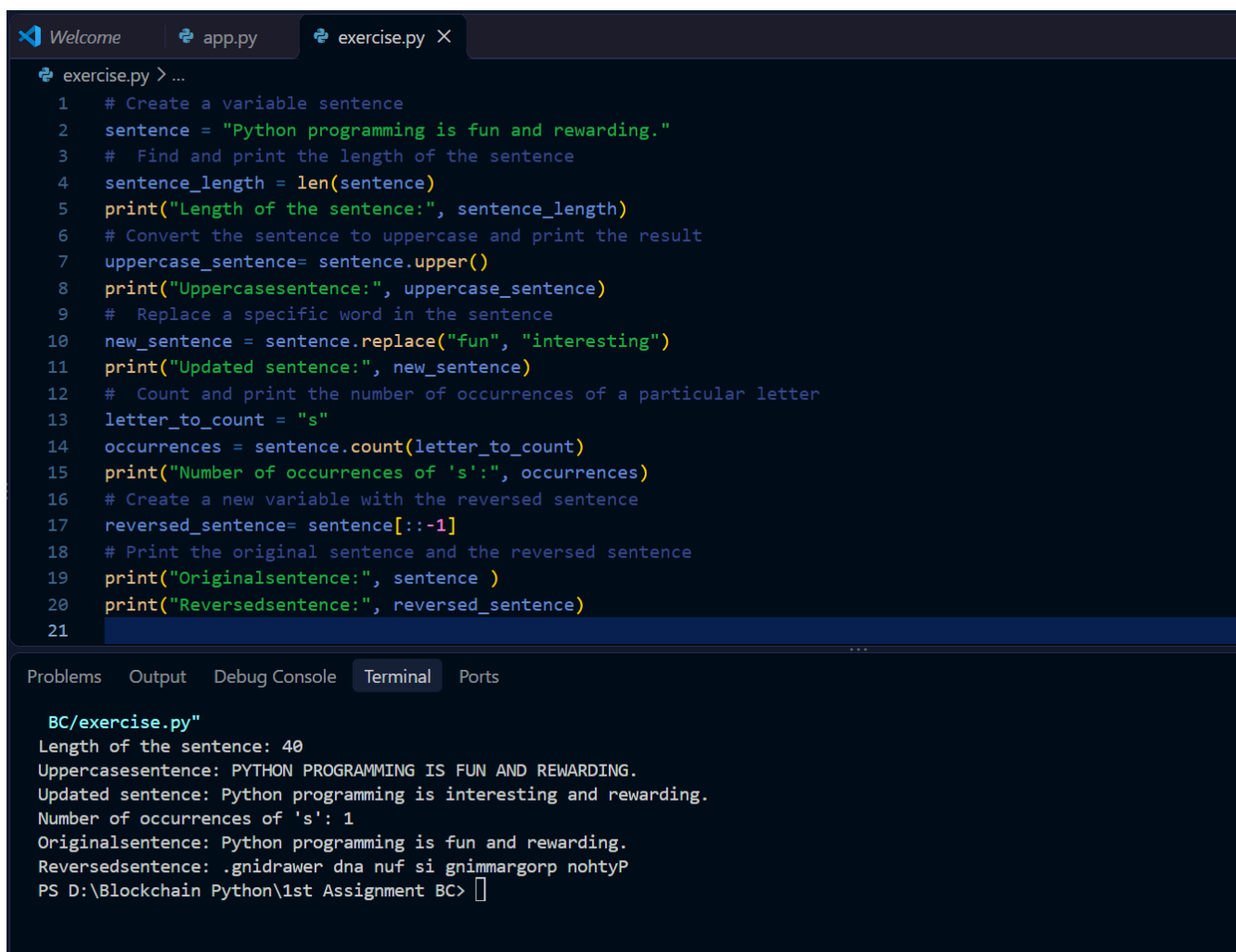
- **and**: True only if both conditions are True.  
Example: True and False  $\rightarrow$  False
- **or**: True if at least one condition is True.  
Example: True or False  $\rightarrow$  True
- **not**: reverses a Boolean value.  
Example: not True  $\rightarrow$  False

## 2.5 Membership Operations: Used to check whether a value exists in a sequence.

- **in:** checks if a value exists.  
Example: 2 in [1, 2, 3] → True
- **not in:** checks if a value does not exist.  
Example: 4 not in [1, 2, 3] → True

## 3. String manipulation: means working with text to create, modify, access, and analyze strings.

- **Concatenation:** Joining two or more strings using +.
- **String Formatting:** Used to insert variables into a string. F-strings are commonly used.
- **Length of a String:** len() counts the number of characters.
- **Accessing Characters:** Characters can be accessed using indexes. Python starts indexing from 0.
- **Slicing:** extracts part of a string.
- **Conversion**
- **Stripping Whitespace:** strip() removes spaces from the beginning and end of a string.
- **Replacing Substrings:** replace() replaces one part of a string with another.
- **Checking for Substrings:** The in operator checks whether a substring exists in a string.
- **Splitting:** split() breaks a string into a list of smaller strings.



```
exercise.py > ...
1 # Create a variable sentence
2 sentence = "Python programming is fun and rewarding."
3 # Find and print the length of the sentence
4 sentence_length = len(sentence)
5 print("Length of the sentence:", sentence_length)
6 # Convert the sentence to uppercase and print the result
7 uppercase_sentence = sentence.upper()
8 print("Uppercasesentence:", uppercase_sentence)
9 # Replace a specific word in the sentence
10 new_sentence = sentence.replace("fun", "interesting")
11 print("Updated sentence:", new_sentence)
12 # Count and print the number of occurrences of a particular letter
13 letter_to_count = "s"
14 occurrences = sentence.count(letter_to_count)
15 print("Number of occurrences of 's':", occurrences)
16 # Create a new variable with the reversed sentence
17 reversed_sentence = sentence[::-1]
18 # Print the original sentence and the reversed sentence
19 print("Originalsentence:", sentence)
20 print("Reversedsentence:", reversed_sentence)
21
```

Problems Output Debug Console Terminal Ports

```
BC/exercise.py"
Length of the sentence: 40
Uppercasesentence: PYTHON PROGRAMMING IS FUN AND REWARDING.
Updated sentence: Python programming is interesting and rewarding.
Number of occurrences of 's': 1
Originalsentence: Python programming is fun and rewarding.
Reversedsentence: .gnidrawer dna nuf si gnimmargorp nohtyP
PS D:\Blockchain Python\1st Assignment BC> |
```