

GET214: **COMPUTING AND SOFTWARE ENGINEERING**

Lesson 1

INTRODUCTION TO PYTHON SYNTAX

Learning outcomes

At the end of the lesson, students should be able:

- i. Explain the strengths of python programming language
- ii. Display output using the `print()` function and obtain user input using the `input()` function
- iii. Explain the rules for naming variables
- iv. Use the built-in `len()` function to get a string's length
- v. Concatenate string literals and variables using the `+` operator
- vi. Use arithmetic operators to perform calculations
- vii. Explain the precedence of arithmetic operators
- viii. Identify the error type and line number in error messages and correct them
- ix. Write comments and docstrings in python

What can we do with Python?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

The Strength of Python Programming Language

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python syntax Compared with Other programming Languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

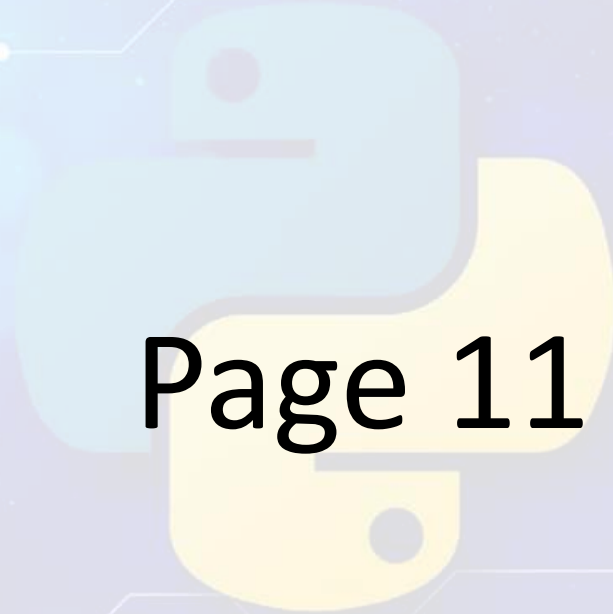
Python **output** and **input**

- Output statement is used to display message on the screen while input statement is used to obtain user entries via the keyboard during program execution.
- Python uses the **print()** function for outputs
- It uses the **input()** function to obtain user inputs

Use of the print() function

Code	Output
<pre>print("Today is Monday.") print("I like string beans.")</pre>	Today is Monday. I like string beans.
<pre>print("Today", "is", "Monday") print("Today", "is", "Monday", sep="...")</pre>	Today is Monday Today...is...Monday
<pre>print("Today is Monday, ", end="") print("I like string beans.")</pre>	Today is Monday, I like string beans.
<pre>print("Today", "is", "Monday", sep="? ", end="!!") print("I like string beans.")</pre>	Today? is? Monday!!I like string beans.

Questions to attempt from reference text



Page 11

Use of the `input()` function

```
variable = input("prompt")
```

- A **variable** refers to a value stored in memory. In the statement above, variable can be replaced with any name the programmer chooses.
- The **`input()`** function reads one line of input from the user. A function is a named, reusable block of code that performs a task when called. The input is stored in the computer's memory and can be accessed later using the variable.
- A **prompt** is a short message that indicates the program is waiting for input. In the statement above, "prompt" can be omitted or replaced with any message.

input(), print() example

```
name = input("Enter your full name: ")  
print("Welcome,", name)
```


```
Enter your full name: Tommy  
Welcome, Tommy
```

Activity 1.1

Difference between source code and output

- State your observed difference between the code and the output displayed in the previous slide.

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Pages 12-13

Variables

- Variables are named memory locations where stored values may change during program execution.
- Variables allow programs to refer to values using names rather than memory locations
- Examples:
 - `name = "Goodluck"`
 - `age = 20`
 - `fees = 40000`
 - `reg_num = "23/EG/EE/3211"`

Rules for naming variables

- A variable name can consist of letters, digits, and underscores and be of any length.
- The name cannot start with a digit.
- Variable names are case sensitive.
- Variable names must not have spaces between characters.
- A name should be short and descriptive, so words are preferred over single characters in programs for readability. *Example:* A variable named count indicates the variable's purpose better than a variable named c.
- Python has reserved words, known as **keywords**, which have special functions and cannot be used as names for variables (or other objects).

Python keywords


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

Activity 1.2

Write a Python computer program that:

- Creates a variable, team1, assigned with the value "Liverpool".
- Creates a variable, team2, assigned with the value "Chelsea".
- Creates a variable score1, assigned with the value 4.
- Creates a variable, score2, assigned with the value 3.
- Prints team1, "versus", and team2 as a single line of output.
- Prints "Final score: ", score1, "to", score2 as a single line of output.

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Pages 14-16

String Basics

- A string is a sequence of characters enclosed by matching single (') or double (") quotes.

Example: "Happy birthday!" and '21' are both strings.

- To include a single quote (') in a string, enclose the string with matching double quotes (").

Example: "Won't this work?"

- To include a double quote ("), enclose the string with matching single quotes (').

Example: 'They said "Try it!", so I did'

Valid and Invalid Strings

Valid string	Invalid string
<code>"17" or '17'</code>	<code>17</code>
<code>"seventeen" or 'seventeen'</code>	<code>seventeen</code>
<code>"Where?" or 'Where?'</code>	<code>"Where?"</code>
<code>"I hope you aren't sad."</code>	<code>'I hope you aren't sad.'</code>
<code>'The teacher said "Correct!" '</code>	<code>"The teacher said "Correct!" "</code>

The Python `len()` function on strings

- A common operation on a string object is to get the string length, or the number of characters in the string, including white spaces.
- The **`len()`** function, when called on a string value, returns the string length.

```
print(len('Python'))  
print(len('CPE 311'))  
print(len('CPE311'))
```

6

7

6

Activity 1.3: The len() function


- Write a program that asks the user to input their first and last name separately. Use the following prompts (example input in bold):

```
What is your first name? Alan  
What is your last name? Turing
```

- The program should then output the length of each name. Based on the example input above, the output would be:

```
Your first name is 4 letters long  
Your last name is 6 letters long
```

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is centered in the background.

Pages 17-18

String Concatenation

- Concatenation is an operation that combines two or more strings sequentially with the concatenation operator (+).


Example: "A" + "part" produces the string "Apart"

Activity 1.4: String concatenation

Write a Python computer program that:

- Assigns the string "Freda" to a variable, name.
- Assigns the string "happy" to a variable, feel.
- Prints the string "Hi Freda!" with a single print() function using the variable name.
- Prints the string "I'm glad you feel happy." with a single print() function using the variable feel.

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is centered in the background.

Pages 18-19

Numbers in Python

- Python supports two basic number formats
 - integer and floating-point
- An integer represents a whole number
- A floating-point format represents a decimal number
- The format a language uses to represent data is called a **data type**
- String is the data type for representing textual data

Python default data types

- Python does not require explicit data type declaration for variables
- The data type of a variable is inferred by the data stored in it.
- Thus, by the initial value stored in a variable, python assumes and data type for it.

Initialized Variable	Assumed Data type
total = 23.5	float
age = 20	integer
first_name = 'Thesy'	string


The `type()` function

- The `type()` function is used to find out of what data type is a variable name.
- Thus, given the variables in the previous slide, the statement `print(type(age))` will return
`<class 'int'>`
- This indicates that the variable, `age`, is of type integer.

Activity 1.5: Values and types

- Write a Python computer program that:
 - I. Defines an integer variable named 'int_a' and assigns 'int_a' with the value 10.
 - II. Defines a floating-point variable named 'float_a' and assigns 'float_a' with the value 10.0.
 - III. Defines a string variable named 'string_a' and assigns 'string_a' with the string value "10".
 - IV. Prints the value of each of the three variables along with their type.

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Pages 20-21

Basic Arithmetic Operators

Operator	Description	Usage Example
+	Addition	$5 + 3 = 8$
-	Subtraction	$5 - 3 = 2$
*	Multiplication	$5 * 3 = 15$
/	Floating point division	$5 / 3 = 1.6666666666666667$
//	Integer division	$5 // 3 = 1$
%	Modulo division	$5 \% 3 = 2$
**	Exponentiation	$5 ** 3 = 125$

Precedence of Operators

- When a calculation has multiple operators, each operator is evaluated in order of **precedence**.

Ex: $1 + 2 * 3$ is 7

because multiplication takes precedence over addition.

- However,

$(1 + 2) * 3$ is 9

because parentheses forces the precedence to be altered such that the elements in the parenthesis are evaluated as a unit of its own.

The precedence of arithmetic operators are in the following order:

- Exponentiation has the highest precedence and is evaluated first.
- Unary negation is evaluated next, before multiplication, division, and remainder.
- Multiplication, both types of division, and remainder are evaluated before addition and subtraction.
- Addition and subtraction are evaluated before assignment (=).
- With two exceptions, operations of equal precedence are left associative, so they are evaluated from left to right.
- Exponentiation and assignment operations are right associative, so consecutive instances of these are evaluated from right to left.
- You can use parentheses to change the order of evaluation.


Expression	Evaluation	Value
$5 + 3 * 2$	$5 + 6$	11
$(5 + 3) * 2$	$8 * 2$	16
$6 \% 2$	0	0
$2 * 3 ** 2$	$2 * 9$	18
$-3 ** 2$	$-(3 ** 2)$	-9
$(3) ** 2$	9	9
$2 ** 3 ** 2$	$2 ** 9$	512
$(2 ** 3) ** 2$	$8 ** 2$	64
$45 / 0$	Error: cannot divide by 0	
$45 \% 0$	Error: cannot divide by 0	

Activity 1.6: Arithmetic Operators

Write a Python computer program that:

1. Assigns the integer value 10 to a variable, meters.
2. Assigns the floating-point value 3.28 to a variable, meter2feet.
3. Calculates 10 meters in feet by multiplying meter by meter2feet. Store the result in a variable, feet.
4. Prints the content of variable feet in the output.

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Pages 21-23

Python Error Messages

- A natural part of programming is making mistakes.
- Even experienced programmers make mistakes when writing code.
- Errors may result when mistakes are made when writing code.
- The computer requires very specific instructions telling the computer what to do.
- If the instructions are not clear, then the computer does not know what to do and gives back an error.

- When an error occurs, Python displays a message with the following information:
 1. The line number of the error.
 2. The type of error (Ex: `SyntaxError`).
 3. Additional details about the error.
- For example, typing `print "Hello!"` without parentheses is a syntax error.
- In Python, parentheses are required to use `print`.

- When attempting to run `print "Hello!"`, Python displays the following error:

```
Traceback (most recent call last):
```

```
File "/home/student/Desktop/example.py", line 1
```

```
    print "Hello"
```

```
    ^
```

```
SyntaxError: Missing parentheses in call to  
'print'. Did you mean print("Hello")?
```

The caret character (^) shows where Python found the error.

Common Types of Errors in Python Programming

Mistake	Error message	Explanation
<pre>print("Have a nice day!"</pre>	<pre>SyntaxError: unexpected EOF while parsing</pre>	The closing parenthesis is missing. Python is surprised to reach the end of file (EOF) before this line is complete.
<pre>word = input("Type a word:)</pre>	<pre>SyntaxError: EOL while scanning string literal</pre>	The closing quote marks are missing. As a result, the string does not terminate before the end of line(EOL).
<pre>print("You typed:", wird)</pre>	<pre>NameError: name 'wird' is not defined</pre>	The spelling of word is incorrect. The programmer accidentally typed the wrong key.

<pre>prints("You typed:", word)</pre>	<pre>NameError: name 'prints' is not defined</pre>	The spelling of <code>print</code> is incorrect. The programmer accidentally typed an extra letter.
<pre>print("Hello")</pre>	<pre>IndentationError: unexpected indent</pre>	The programmer accidentally typed a space at the start of the line.
<pre>print("Goodbye")</pre>	<pre>IndentationError: unexpected indent</pre>	The programmer accidentally pressed the Tab key at the start of the line.

Activity 1.7: Error Messages 1

The following program has three errors.

- Type the program into your IDE exactly as it appears
- Run the program to find the first error, and correct the corresponding line of code.
- Then run the program again to find and correct the second error.
- Keep running and correcting the program until no errors are found.

```
word = input("What is your favorite word? ")  
print()  
print("You said:", Word)  
print("That's all, Folks!")
```


Activity 1.7: Error Messages 2

This code is based on an earlier example, but the code contains several mistakes.

- One line is missing required punctuation, and another line uses incorrect symbols.
- Run the program to find the first error, and correct the corresponding line of code.
- Keep running and correcting the program until no errors are found.

```
team1 = "Liverpool"  
score1 = "4"  
team2 = "Chelsea"  
score2 = "3"  
  
print(team1 "versus" team2)  
print["Final score:" score1 "to" score2]
```

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one blue and one yellow, is centered in the background.

Pages 25-26,28

Python Comments

- In the program segment, lines 1, 8, and 10 contain comments.
- Each comment begins with a hash character (#).
- All text from the hash character to the end of the line is ignored when running the program.
- In contrast, hash characters inside of strings are treated as regular text.
- For instance, the string "Item #1: " does not contain a comment.

```
1  # Display the menu options
2  print("Lunch Menu")
3  print("-----")
4  print("Burrito")
5  print("Enchilada")
6  print("Taco")
7  print("Salad")
8  print() # End of menu
9
10 # Get the user's preferences
11 item1 = input("Item #1: ")
12 item2 = input("Item #2: ")
```

How to write comments

When writing comments:

- The # character should be followed by a single space.
- For example, # End of menu is easier to read than #End of menu.
- Comments should explain the purpose of the code, not just repeat the code itself.
- For example, # Get the user's preferences is more descriptive than # Input item1 and item2.

Python Docstring

- Python programs may optionally begin with a string known as a docstring.
- A **docstring** is documentation written for others who will use the program but not necessarily read the source code.
- Documentation can be long, so docstrings are generally written as multi-line strings (`"""..."""`).
- Common elements of a docstring include a one-line summary, a blank line, and a longer description.

Questions to attempt from reference text

The Python logo, consisting of two interlocking snakes, one light blue and one light yellow, is centered in the background.

Pages 30