

Lesson # & Title	Focused Area(s)	Lesson Objectives	Graded Tasks (%)
LESSON 1 Introduction to Python Basics	Topic 1: Introduction to Python Topic 2: Python Syntax Topic 3: Variables and types Topic 4: Control Flow Topic 5: Functions and Packages	<ul style="list-style-type: none"> Understand the importance of programming language and what is Python. Learn how to install Jupyter editor and write the python syntax. Learn how to construct variables and the different operations and methods used with each variable data type Understand how to use control flows to manage code structure. Implement user-defined functions using Python. 	

Why This Lesson

Python is a high-level programming language used in different disciplines, especially for mathematics and data analysis. It has large standard libraries that help to solve different kinds of problems like machine learning libraries, visualisation libraries, data analysis libraries, and mathematical libraries. In this lesson, you will have the initial skills to run Python code starting from using Python for different arithmetic operations, to constructing your own functions to solve different problems.

In this lesson, you are required to finish the UNITAR learn online course: DAT208x Introduction to Python Programming.

- Passing grade is 70%
- Finish the virtual lab.

TOPIC 1: Introduction to Python

Program is a set of statements (instructions) that is used to perform a specific task. The task should be performed in the machine (computer) which is based on some calculations the machine can understand and then put the results to the end user. The program code is written using a programming language tool. There are a lot of programming languages such as Java, C++, and Python. These languages are considered high-level programming languages where the way of writing the code is close to human understanding (language). Each language has its own structure and rules.

1.1 Programming.

As defined earlier, programming as a concept is important to develop different solutions for different tasks that make these tasks easier for end users to use. For example, create a program (we can call it at the end as a system, or application depending on the output) for a supermarket, to count the sold items and how many items are available. The program will give a report on the status of all products, the product expiry dates, and the need for more items of specific products. It is obvious how programs now are important in most sectors. Even these programs can be designed to make difficult tasks to be possible to be solved or to be with a value. For example, using artificial intelligence and specifically machine learning algorithms to build a predictive model that can predict the status of the weather, the stock market, the rental rate, or a patient diagnosis based on patient records. The use of different systems that ease the work for a specialist to perform their tasks is also an important key to how programming is important to improve the accuracy and effectiveness of the whole work or organisation.

Based on that, programmers need to understand how to understand the different tasks, how the programming language can learn and be used with the computer to learn about the task and so it specific job. To transfer the problem to the programming language and the programming language translates the problem to the machine, and the machine then makes the received orders and performs the tasks and gives the output.

1.2 "Think"

The code is a set of instructions, that can be defined as algorithms, that computers executed to solve a certain problem. Computer is suitable to perform such tasks that are based on different calculations and then transformation of data to produce the information, or the output data. For example, an algorithm to compute the square of a number, or an algorithm to draw a specific chart for the series of data in the form of a column chart or line chart. This algorithm then needs to get input, execute the instruction, then produce the output, which is where the final chart, or the squared value. So, the data input is transformed into another data. The computer's job is to perform these instructions as received and will not do any input to these instructions. Computers execute as how the algorithm (programmer) **thinks**. As a result, the programming skills can be described as "how to think!" before "how to do!". Any error in the output will not be a computer error, it is the programmer error.

TOPIC 2: Python Syntax

The Python as a high-level language deals with different data types. Therefore, there is roles and structure that should be followed to be familiar with the code syntax. In this topic, students will learn how to do different operators in Python's such as arithmetic operators and logical operators. How to use the Python libraries and different functions such as `sum()`, `sorted()`, and `type()`, where the programs need specific operations. In addition,

the students will learn the different data types with the operations applied to them, the roles, and the methods associated with each data type.

2.1 Python

Pythons are designed at a high-level to make programmers or any human read the code and understand the concept easily. Even easy to write the code and be familiar with the structures and rules that a programmer needs to know to create a complete project. Python is designed to be interpreted programming language where no need for the computer to compile the program into machine instructions. (Search on other interpreted programming languages)

As can be shown in Figure 1, Python becomes at the top of other programming languages based on a study by TIOBE software quality company. Compared to 2021, the popularity of Pythons increased by 3.56% which makes python to be the first programming language, and C comes in the second rank, then Java. With the increased use of data analytics and machine learning, Python is one of the programming languages that have many libraries to support solving complex problems. Many disciplines now can use Python to solve these problems either in data science, computer science, engineering, or mathematics.







Aug 2022	Aug 2021	Change	Programming Language	Ratings	Change
1	2	▲	 Python	15.42%	+3.56%
2	1	▼	 C	14.59%	+2.03%
3	3		 Java	12.40%	+1.96%
4	4		 C++	10.17%	+2.81%
5	5		 C#	5.59%	+0.45%
6	6		 Visual Basic	4.99%	+0.33%
7	7		 JavaScript	2.33%	-0.61%
8	9	▲	 Assembly language	2.17%	+0.14%
9	10	▲	 SQL	1.70%	+0.23%

Figure 1. Top programming Languages, 2022

(source: <https://www.tiobe.com/tiobe-index/>)

With Python, you are able to use a huge number of libraries or install any libraries or packages that help solve problems. There are many standard libraries that contain functions that can be used in your code such as the `sum()` function. Example of libraries that you can install, and use is the Numpy library which is used for numerical analysis. It deals with arrays and has many functions that can be used with arrays or with linear

algebra. In Python 3, the Numpy package becomes available by default, and you just need to import the library to your program, no need to install it.

2.1 Syntax

- Comments:

Comments in any programming language give the ability for the programmer to explain the statement (line of code), a portion of code, function, class, or package. These comments are not considered as part of the instructions and computers will not execute these lines. The main purpose of comments is to allow any programmer to understand each part of the code.

There are two ways in python to create a comment:

1. Single-Line comment. You start the line using the hash character "#".
2. Multiline comment. You start and end the comment with triple quotes " " " (see Figure 2)



```
# I'm a single-line comment (starting with the '#' character)

"""m a multi-line comment,
also called a docstring,
enclosed in triple quotes
"""
```

Figure 2. Comments in python

- Numbers and Letters

Python deals with numbers and texts. The numbers can be integers, or decimal numbers. When dealing with numbers you can directly put the number as it is, and Python can interpret the number. To work with text, you need to encapsulate the text with single/double quotes where Python understands them clearly. If you put the number inside a quotation, then this number will be considered as a text. Texts are represented as Strings as depicted in Figure 3. Later we will learn different methods that can be used with numbers and strings as well.

```

# the number 5
5

# negative numbers work as you'd expect
-25

# a floating point number
3.1416

# a floating point number in scientific notation
1e-5

# strings be enclosed with double quotes
"banana"

# or single quotes
'bananarama'

# strings can contain multiple words and punctuation
"Hi Class, I'm a string with multiple words!"

```

Figure 3. Numbers and string in python

- Operations

There are many operations that can be done on numbers and strings with limited rules with strings. The arithmetic operations like "+, -, *, /" can be used directly with also additional operations with numbers. Table 1 shows some of the arithmetic operations used in python.

Table 1. Arithmetic operations in python

Operators	Meaning	Example	Result
+	Addition	4 + 2	6
-	Subtraction	4 - 2	2
*	Multiplication	4 * 2	8
/	Division	4 / 2	2
%	Modulus operator to get remainder in integer division	5 % 2	1
**	Exponent	5**2 = 5 ²	25
//	Integer Division/ Floor Division	5//2 -5//2	2 -3

Operation on strings is limited with some operators that return specific output. For example, the plus operator "+" when it comes between two strings, then this operation is a concatenating operation. Multiplication operator is just allowed when one operand is a string, and the other operand is an integer. The return of this operation is a duplication of the string times the integer number. Figure 4 explains the two operations.

```
In [1]: ► "Big " + "Data " + "Analytics."
Out[1]: 'Big Data Analytics.'
```

```
In [3]: ► "Data " * 4
Out[3]: 'Data Data Data Data '
```

Figure 4. Operations on Strings

Try to do other operations on a string and discuss the output.

- Functions and methods

There are different functions that can be used in python to give more details about the data. For example, the functions

- `type(arg)`: returns the datatype of the argument.
- `Abs(arg)`: returns the absolute value of a number.
- `Min(arg1, agr2, ag3,)`: returns the minimum value among the list of values.
- `Print()`: special function used to print its arguments to the screen.
- `Str(Number)`: return the numeric value into a string.
- `Int(arg)`: returns the input argument, float or character, into an integer.

Methods are functions that do a specific task on a specific data type. There are methods that are applied to numbers and methods that are applied to strings. Example:

- `"test one".upper()`: returns the string test with upper case (output: "TEST ONE")
- `"Test one".replace("one", "two")`: replace method replaces the matched first argument with the second argument. The arguments could be one character, a word, or more. Any match will be replaced. It is important to know that this method is case-sensitive. (Output: "Test two")

TOPIC 3: Variables and types

A variable is a name the programmer defines to represent a value, usually a number or a text string. It is a text symbol that represents a value; it represents the location where the computer keeps that value. Since numbers are the basis of everything a computer performs, the location of each object is a number that designates the location in computer memory where that object is stored. When working with data types like strings and numbers, it's crucial to save the relevant information in memory so that it may be used later when executing various

statements. The data is temporarily stored for this purpose using the word variables, which will thereafter undergo various transformations throughout the code.

Different data types are available for variables. A value must be assigned when defining a variable, and the data type of the value determines the variable's data type as well.

- How to define a Variable

In python, to define a variable, you should follow first the rules of defining the variable name. the name should be one word that can be a mix of characters or words and numbers, can be separated by underscore "_", can't start with numbers, and can't use special characters.

After choosing the proper name, you define the variable using the "=" operator as follows:

```
my_number = 5
```

Here we are defining the variable datatype with the type of the operand on the right side of the = operator.

Depending on the variable datatype, all the operations discussed earlier are applied to the variables. One important key with variables in python, you can redefine the data type of the variable by giving it a new datatype during the code run. For example, let's assume that you have variable1 which is defined first as float, and during the running of the code, you have changed the value of variable1 to be in integer type. Then, the datatype of variable1 will be an integer. After a few lines in the code, you used the same name with a different datatype, or string type. Then, the variable will take the new datatype.

```
variable1 = 50.10 # this variable of type float.
```

```
variable1 = 50 # now the variable of type integer.
```

```
Variable1 = "It is time to be in string format" # the variable becomes in string type.
```

That means, when dealing with variables, it is necessary to make sure that you are using the correct variable datatypes.

- Lists and dictionary

There is also the ability to define a variable that can hold different data with different types. These variables can be either list type or dictionary type.

- List is a collection datatype defined as variables where the data on the right side is enclosed using square brackets []. The data in the list is separated by a comma, and data are called items, and they are ordered as defined. Each item can be accessed using an index. The first item is at index 0.



```
my_list = [5, 3, 1, 2, 4]
```

List has methods that can help work with data like sorting the list, append method to add items to the list, reverse methods to reverse the content of the list, and pop method that will remove and return the last item in the list.

```
my_list = ["Apple", "Banana", "Cantaloupe"]

# .reverse( ) reverses the elements of the list in place
my_list.reverse( )
print( my_list )

# .sort( ) sorts the list in-place
my_list.sort( )
print( my_list )

my_list.pop( )
```

Output:

```
['Cantaloupe', 'Banana', 'Apple']
```

```
['Apple', 'Banana', 'Cantaloupe']
```

```
['Apple', 'Banana']
```

To retrieve “Apple” which is the first item at index 0, you can use the variable name with the indicating the index of the item as follows:

```
Print(my_list[0]) # the output will be “Apple”
```

This also applied to the rest of the items in the boundary of the number of items. Here in the example above, there are three items with indexes 0, 1, and 2. To retrieve items from the end, it is also applicable to use [-1] to indicate the last item.

Another indexing option is to retrieve a group of items. This can be done using the slice “:” to indicate the start index and the last index. The last index will not be retrieved.

```
Print(my_list[0:2]) # the output will be “Apple, Banana”, Cantaloup will not be retrieved.
```

- Dictionary

Is a collection of data that take two details, {key: value}

Example:

```
In [4]: ►
Weeks = {
    1 : "Mon",
    2 : "Tue",
    3 : "Wed"}
print (Weeks)

{1: 'Mon', 2: 'Tue', 3: 'Wed'}
```


The access to dictionaries and the values can be as in lists. The values can take numbers, strings, lists, or another dictionary. The Key can be a number or a string that indicates the dictionary index.

TOPIC 4: Control Flows

Usually, statements in a program execute in the order in which they're written. This is called sequential execution. Various Python statements enable you to specify that the next statement to execute may be other than the next one in sequence. This is called transfer of control and is achieved with Python control flow, or control statements. Python provides three types of selection statements that execute code based on a condition—an expression that evaluates to either True or False:

- The if statement performs an action if a condition is True or skips the action if the condition is False.
- The if...else statement performs an action if a condition is True or performs a different action if the condition is False.
- The if...elif...else statement performs one of many different actions, depending on the truth or falsity of several conditions.
- To write an if statement you need to specify the logical condition and what action to be executed if the condition is true.

```
x = 4
if x % 2 == 0:
    print( x, "is even" )
```

4 is even

In this example, the condition is to check if the number is even. If true, it will print the statement, {number} is even.

```
TEST = "eric" in "America" # <- Boolean value stored in all-caps variable
if TEST:
    print( "I found a substring!" )
```

I found a substring!

This example is to compare strings. The operator "in" is used to match if the left operand exists in the right operand.

- if/else examples:

```
x = 4
if x % 2 == 0:
    print( x, "is even" )
else:
    print( x, "is odd" )
```

4 is even

In this example, if the number is not even, then it will execute the else statement. You can add also another condition with an else statement by using “elif” which here means else if, followed by the condition like:

```
traffic_signal = "Yellow"
if traffic_signal == "Green":
    print( "Let's go!" )
elif traffic_signal == "Yellow":
    print( "Prepare to stop." )
elif traffic_signal == "Red":
    print( "Stop!" )
else:
    print( "Unknown signal" )
```

Prepare to stop.

You can also add more conditions in one statement using *and*, *or* as logical combinations.

```
grade = 65
if grade > 60 and grade <= 65:
    print( "You got B-" )
```

Python provides two repetition statements—while and for:

- The while statement repeats an action (or a group of actions) as long as a condition remains True.
- For statement repeats an action (or a group of actions) for every item in a sequence of items.

```
x = 0
while x < 10:
    print( x, end=" " )
    x += 1
```

0 1 2 3 4 5 6 7 8 9

In this simple example, you can learn the structure of the loop. The condition is important to have a stopping point or otherwise, you will enter an infinite loop. The loop will not stop.

TOPIC 5: Functions and Packages

Functions are pieces of reusable code that solve a particular task. There is a large and useful set of functions built into Python. These are sometimes simply there for the using, like print and input, and sometimes are part of a module that must be imported, and like random. However large this collection of functions is, it is impossible that it will include everything that every programmer needs. At some point, there will be a need to create a function that does something new, and Python should permit this. Why would a programmer want to create a function of their own? It is partly a principle of “reduce, reuse, or recycle.” Functions are all about code reuse. If some section of code can be invoked as a function instead of being repeated many times, then there will be less typing involved. It is also to support more correct programs: a small code unit like a function can be very thoroughly tested and nearly guaranteed to be correct. It is also to promote the use of working code: once a function is tested, it can be placed in a collection of code (module) and used again instead of being rewritten many times. Packages are like a directory of Python Scripts where each script = module specifies functions, methods, and types. There are thousands of packages available in python such as:

- Numpy
- Matplotlib
- Scikit-learn

Forum (not Graded)

In your organisation, how do you think big data can impact your organisation?

- end of lesson content –