

# AI With Python Workshop

This notebook complements the powerpoint slides during the workshop and will be used to do the coding exercises

## 1.1 Hello World

Our first coding exercise. We print "Hello World" on our computer screen. The `print` function prints a message on the screen.

In the coding cell below type `print("Hello World")` and press the **Run** button

```
# Write code below  
print("Hello World")
```

## 1.2 Variables

Variables act like containers which can store values.

Python has no command for defining a variable. A variable is defined the moment you assign a value to it. E.g.,

- `name = "Ben"`
- `age = 12`
- `math_score = 95.3`

**Task 1:** Define a variable called `name` to store your name. Use the `print` function to print it on screen. Complete the task in the code cell below.

Can use the `type()` function to check the type of a variable.

```
# Write code below
name = "Symphony"
print(name)
print(type(name))
```

## 1.3 Relational Operators

Relational Operators are used for comparing values. It returns a Boolean value.

Commonly used operators are:

- `==` (equal to)
- `<` (Less than)
- `>` (Greater than)
- `>=` (Greater than or equal to)
- `<=` (Less than or equal to)
- `!=` (Not equal to)

**Task 1:** Let's try some of these relational operators.

```
# Run this code
number_1 = 4
number_2 = 4
number_3 = 7

string_1 = "Hello"
string_2 = "hello"

# == Equal to operator
print("Is number_1 is equal to number_2? ", number_1 == number_2)

# < Less than operator
print("Is number_1 is less than number_2? ", number_1 < number_2)

# <= Less than equal to operator
print("Is number_1 is less than or equal to number_2? ", number_1 <= number_2)

# > Greater than operator
print("Is number_1 is greater than number_3? ", number_1 > number_2)

# == Equal to operator for string
print("Is string_1 is equal to string_2? ", string_1 == string_2)
```

## 1.4 Conditional Statements

Conditional Statements are statements that can control the flow of a program. One of the conditional statements is called if-else statement.

The syntax of `if-else` statements is:

```
if condition_1:
    code block_1

elif condition_2:
    code block_2

elif condition_3:
    code block_3
...
else:
    code block_N
```

**Note:** Python uses indentation to indicate a new code block.

**Task 1:** Use Python to code the real-life example given in the workshop slide.

```
# Write the code below. We have started the code for you.
weather = "Sunny"
if weather == "Sunny":
    print("Wear sunglasses")

elif weather == "Rainy":
    print("Take umbrella")

elif weather == "Typhoon":
    print("Stay home")

else:
    print("Enjoy!")
```

## 1.5 Lists

List is used to store multiple items in a single variable. Items of a list can be any data type.

Some examples of lists in Python are:

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

```
list_of_strings = ["Laptop", "Hello how are you?", "Apple"]
```

```
mixed_list = [1, 2, "Laptop", "Hotpot", 5]
```

To access an item of a list, we use **index**. In Python, index starts from **0**, which is the **first** item of the list.

**Task 1:** Create a python list called `shopping_list` to store the strings `"grape"` , `"apple"` , and `"butter"` . Please (1) print the list and (2) print the last element of the list.

```
# Write the code below
shopping_list = ["grape", "apple", "butter"]
print(shopping_list)
print(shopping_list[2])
```

## 1.6 Dictionaries

Dictionary also can store multiple items in a single variable, but items are stored as **key: value** pairs.

Example of a Dictionary in Python:

```
my_dictionary = {"name": "Eason", "age": 27, "gender": "M"}
```

Values in dictionaries can be accessed with the key. For example, to print the value of element with `key="name"` in `my_dictionary` , we do:

```
print(my_dictionary["name"])
```

**Output:** `"Eason"`

**Task 1:** Create a dictionary with your following information (keys) – name, age, gender, major. Please (1) print the dictionary and (2) print the value with the key "name".

```
# Write the code below
my_dictionary = {"name": "Eason", "age": 27, "gender": "M", "major": "educatio
print(my_dictionary)
print(my_dictionary["name"])
```

## 1.7 Loops

Loop is for repeating the same code block multiple times. Loops can make the code shorter.

In this workshop we will only focus on 1 type of loop, i.e., `for-loop`. `for-loop` is particularly useful to iterate over a list.

Syntax for `for-loop` in Python is as follows:

```
for <variable> in <list or sequence>:
    part of code that needs to be repeated
```

An example of usage of for loop:

```
my_list = ["Hello", "How are you?", 1, 2, 3]

for element in my_list:
    print(element)
```

**Output:**

```
"Hello"
"How are you?"
1
2
3
```

**Task 1:** Create a loop that prints number 0-10

*Hint:* To create a sequence (NOT LIST) from **0** to **n** you can use the range function as

```
range(0, n+1)
```

```
# Write code below
for number in range(0,11):
    print(number)
```

## 1.8 Functions

Function is a block of organized, reusable code that is used to perform a single, related action.

You can pass parameters into a function (optional) and the function can also return values (optional).

Before you call a function, you need to define it.

The syntax for defining a function is:

```
def Function_Name(parameter1, parameter2):
    Code block to run
```

Syntax for calling a function is:

```
Function_Name(parameter1_value, parameter2_value)
```

**Task 1:** Define a function to print "Welcome to the workshop". Hint: The function tasks no parameter and returns nothing.

```
# Write the code below
def greeting():
    print("Welcome to the Workshop")

greeting()
```

**Task 2:** Define a function to print the addition of any two numbers. Hint: The function tasks two parameters and returns nothing.

```
# Write the code below
def addition(number1,number2):
    print(number1+number2)

addition(2,3)
```

**Task 3:** Define a function that calculates the multiplication of any two numbers and returns the value. Hint: The function takes two parameters and returns a value.

```
# Write the code below
def multiplication(number1,number2):
    return number1*number2

print(multiplication(5,4))
```

## 1.9 Libraries

Libraries are a set of predefined code that can be re-used. It saves our time so that we don't have to code everything from scratch. Libraries make coding a lot easier!

To import the whole library we use:

```
import library_name
```

**Task 1:** Import `art` library and try its `tprint` function like this:

```
art.tprint("AI with Python")
```

```
# Run the cell
# Some preparations
!pip install --upgrade pip
!pip install art
```

```
# Write the code below  
import art  
art.tprint(text="AI with Python")
```

## End of Part 1 -- Basics

*Don't forget to stop the computation for your notebook. Go to **Run** and then click on **Stop Machine***