CUHK Jockey Club AI for the Future Project
中大賽馬會智為未來計劃

AI With Python Workshop

Welcome to the AI with Python Workshop by CUHK-Jockey Club AI for the Future Project

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.

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

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

1.2 Variables

Variables are containers storing values.

E.g., To initialize variables,

- `my_string_variable = "Welcome to the workshop"`
- `my_number_variable = 1702`
- `my_decimal_variable = 3.4`

Let's play with variables

Task 1: Create a variable called `name` to store your name. Use print statement to print it on screen.

Hint: Use `print()` function to print the variable.

Complete the task in the code cell below.

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

Task 2: Create 3 variables x, y, and z. Store their values as follows: x = 3; y = 7; z = x + y. Print the value of z and verify if it is correct.

Complete the task in the code cell below.

```
# Write code below
x = 3
y = 7
z = x+y
print(z)
```

1.3 Data Types

Variables can store data of different types, and different types can do different things.

The most common ones are:

- **Integers:** Numbers (E.g., 1, 400, -999, etc.)
- **Float:** Decimals (E.g., 3.67, -4.7, 9.42, etc.)
- **String:** Collection of characters (E.g., "Hello", "Good morning", "Amazing",etc.)
- **Boolean:** True and False

Bonus: To check the data type of any variable you can use `type()` function. E.g., for a variable named `my_variable`, use `type(my_variable)`

```
# Try Bonus below
my_variable1 = 3
my_variable2 = "Hello"
print(type(my_variable1))
print(type(my_variable2))
```

1.4 Relational Operators

Relational are used for comparing values.

Commonly used operators are:

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

Let's try some of these relational operators

```
# Run this code
number_1 = 23
number_2 = 23
number_3 = 30

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)
```

Task 1: Print if number_3 is greater than number_1

```
# Complete the code below
print("Is number_3 greater than number 1?")
print(number_3 > number_1)
```

1.5 Conditional Statements

Conditional Statements control the flow of program based on conditions.

We will use `if-else` statements.

The syntax of `if-else` statements is:

```
if (Condition 1):
    Statements_1
    ...
elif (Condition 2):
    Statements_2
    ...
elif (Condition 3):
    Statements_3
    ...
...
else:
    Statements_n...
```

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

Task 1: Run the code below to understand the `if-else` statement syntax

```
# Run this code
# You can try changing the values of numbers and modify the code to play with it
number_1 = 5
number_2 = 4

if (number_1 > number_2):
    print("Number 1 is greater than Number 2")
    print("Yayyy!")

elif (number_1 == number_2):
    print("Number 1 is equal to Number 2")

else:
    print("Number 1 is less than Number 2")
    print("Interesting!")
```

Task 2: 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!")
```

1.6 Lists

List is used to store multiple items in a single variable.

Some examples of list in Python is:

```
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 any element of a list, we use **index**. Index starts with 0.
For example, if we want to print the first element (i.e., index 0) of the list `mixed_list`.

We will do: `print(mixed_list[0])`

Output `1`

Similarly, to access the last element of the list `list_of_strings`, we will use:
`list_of_strings[2]`

Value of list_of_strings[2]: `"Apple"`

Task 1: Create a python list called `shopping_list` and print the last element of the shopping list.

```
# Write the code below
shopping_list = ["Apple", "Butter", "Cheese"]
print(shopping_list[2])
```

Task 2: Change the last element of the list to "Laptop" and print the whole list.

```
# Write the code below
shopping_list[2] = "Laptop"
print(shopping_list)
```

1.7 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": "Symphony", "age": 30, "gender": "F", "major": "computer science, cognitive science"}`

In a dictionary, there are key-value pairs in the form `{key:value}`

To access an element/value of a list, we use the key. For example, to print the value of element with `key="name"` in `my_dictionary`, we do:

`print(my_dictionary["name"])`
Output: `"Symphony"`

Task 1: Create a dictionary with your following information (keys) – name, age, gender, major.

```
# Write the code below
my_dictionary = {"name": "Symphony", "age": 30, "gender": "F", "major": "computer science, cognitive science"}
```

Task 2: Print this dictionary.

```
# Write the code below
print(my_dictionary)
```

Task 3: Change the value of the key age to your age minus 10.

```
# Write the code below
my_dictionary["age"] = my_dictionary["age"] - 10
print(my_dictionary)
```

1.8 Loops

Loop is for repeating the same code block multiple times. Loop can make 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)
```

Task 2: Create a loop that prints "Hello" 5 times

```
# Write code below
for number in range(0,5):
    print("Hello")
```

1.9 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)
```

Some examples of functions:

- Function that takes no parameter and returns nothing

```
def foo():
    print("Hello World")
    print("We are going to learn Python today")
```

- Function that takes parameters and returns nothing

```
def bar(sentence_1, sentence_2):
    print(sentence_1)
    print(sentence_2)
```

- Function that takes parameters and returns something

```
def foobar(number_1, number_2):
    return number_1 + number_2
```

```
# Run this cell and try calling the defined functions in the next cell
def foo():
    print("Hello World")
    print("We are going to learn Python today")

def bar(sentence_1, sentence_2):
    print(sentence_1)
    print(sentence_2)

def foobar(number_1, number_2):
    return number_1 + number_2
```

```
# Try calling the foo, bar, and foobar function below and notice the differences.
foo()

bar("Hello", "How are you")

addition = foobar(7,3)

print(addition)
```

Task 1: Create a function that prints your "Welcome to the AI with Python workshop"

```
# Write the code below
def my_function():
    print("Welcome to the AI Python Workshop")

my_function()
```

Task 2: Create a function that takes two numbers as its parameters and returns the multiplication of those two numbers

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

multiplied = multiplier(5,4)

print(multiplied)
```

1.10 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 whole library we use:

```
import library_name
```

To import a function from a library we use:

```
from library_name import function_name
```

Task 1: Install `emoji` and `art` libraries

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

Task 2: From `emoji` library import the `emojize` function and use it like this:

```
emojize("Python is fun :red_heart: We will use Python to create mind-blowing :exploding_head: AI :robot:")
```

Can you guess what it does?

```
# Write the code below
from emoji import emojize
print(emojize("Python is fun :red_heart: We will use Python to create mind-blowing :exploding_head: AI :robot:"))
```

Task 3: Import `art` library and try its tprint function like this:

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

```
# 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 Computation`