Welcome to our first Python Tutorial!

Python is a general-purpose programming language.

This means that you can use Python for a wide variety of programming tasks, from web development to data analysis and beyond.

Let's start with the basics.

1. VARIABLES AND DATA TYPES

A **variable** is a *name* that refers to a *value*.

Here's how you assign a *value* to a **variable** in Python:

+ Code + Text

```
my_variable = 10
```

To view the contents of a variable, you can access its value by "**calling**" it, which involves invoking its name:

```
print(my_variable)  # This will output: 10
```

Python has several basic **data types**, including:

```
# Integer (whole number)
my_integer = 10
print(my_integer)  # This will output: 10


# Float (decimal number)
my_float = 10.0
print(my_float)  # This will output: 10.0


# String (sequence of characters)
my_string = "Hello, world!"
print(my_string)  # This will output: Hello, world!


# Boolean (True or False)
my_boolean = True
print(my_boolean)  # This will output: True
```

In addition to single values, which are referred to as **scalars**, Python also supports **more complex data structures** like **lists** and **dictionaries**.

These data structures enable you to **organize and manipulate data** in more intricate ways.

Lists are **ordered** collections of elements that can contain multiple values of any type:

```
my_list = [1, 2, 3, 4, 5]
print(my_list) # Here we have created a list of integers
```

To **access a specific element in a list**, you can use **square brackets [ ]** and provide the **index** of the desired element.

For example:

```
fruits = ["apple", "banana", "orange", "mango"]


fruits[0]  # Output: apple
```

Q: What will the following command print:

```
fruits[2]
```

You can also use **negative indexing** to access elements from the end of the list.

The last element has an index of -1, the second-to-last element has an index of -2, and so on.

For example:

```
print(fruits[-1])  # Output: mango
print(fruits[-3])  # Output: banana
```

In addition to accessing individual elements, you can also **access a range of elements** using **slicing**.

Slicing allows you to **extract a sublist** by specifying a **start index**, an **end index** (**exclusive**), and an **optional step value**.

For example:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]


print(numbers[2:5])      # Output: [3, 4, 5]
print(numbers[1:7:2])    # Output: [2, 4, 6]
```

Dictionaries, on the other hand, are **unordered** collections of **key-value pairs**, allowing you to store and retrieve values based on their associated keys.

The first step is to **define** a dictionary:

```
student = {
    "name": "Suleiman",
    "age": 20,
    "courses": ["math", "science"]
}
```

Q: Write a command that outputs the contents of the dictionary we just defined:

```
# Write your code here:
```

Accessing elements in a dictionary:

```
print(student["name"])  # This will print: John
```

What does the following dictionary contain:

```
students = {
    "student1": {
        "name": "Suleiman",
        "age": 20
    },
    "student2": {
        "name": "Shu",
        "age": 22
    },
    "student3": {
        "name": "Lucy",
        "age": 19
    }
}
```

2. BASIC OPERATIONS

Python supports **all of the basic arithmetic operations**:

```
# Addition
add_result = 10 + 5
print(add_result)  # This will output: 15


# Subtraction
subtract_result = 10 - 5
print(subtract_result)  # This will output: 5


# Multiplication
multiply_result = 10 * 5
print(multiply_result)  # This will output: 50


# Division
divide_result = 10 / 5
print(divide_result)  # This will output: 2.0
```

Q: What will be the output of the following command:

```
num1 = 10
num2 = 5
print(num1 + num2)
```

3. CONTROL STRUCTURES

Python includes several control structures that **let you manage the flow of your program**.

One example is the use of **if statements**, which enable you to **evaluate logical conditions**:

```
age = 18
if age >= 18:
    print("You are an adult")  # This will print: You are an adult
```

And here is an example of an **if-else statement**:

```
my_number = 10

if my_number > 5:
    print("The number is greater than 5.")  # This will output: The number is greater than 5.
else:
    print("The number is 5 or less.")
```

For loops in Python **iterate over a sequence** (like a **list**, a **range**, or a **string**), **in the order that they appear** in the sequence.

Here is an example of a **for loop**, which **outputs each element in our list**:

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

Here is another example:

```
for i in range(5):
    print(i)  # This will output: 0, 1, 2, 3, 4 (on separate lines)
```

Q: What is

`range(5)`

?

Q: Write a for loop which iterates over the fruits list and prints each fruit in the list:

```
# Enter your code here:
```

4. FUNCTIONS

A **function** is a block of code that **performs a specific task**. You can define your own functions using the '**def**' **keyword**.

Q: What does the following function do?

```
def add_numbers(a, b):
    return a + b


add_numbers(num1, num2)
```

Q: How about this one?

```
def greet(name):
    """This function greets the person passed in as parameter"""
    print("Hello, " + name + ". Good morning!")
```

Q: What command should we type to actually use the function?

```
# Write code to print: Hello, Aysha. Good morning!
```

The following is an example of a function with **multiple parameters**:

```
def calculate_area(length, width):
    """This function calculates the area of a rectangle."""
    return length * width

area = calculate_area(10, 5)
print(area)  # This will print: 50
```

Congratulations! You have completed a basic introduction to Python!