Python for Beginners (Part 1)

Introduction

Python is a high-level, interpreted programming language that is easy to learn and can be used for a wide range of tasks, such as web development, scientific computing, and data analysis. This tutorial will introduce you to the basics of the Python language, including variables, data types, control structures, functions, tuples, lists, dictionaries and classes.

Hello World program

```
print("Hello world")
```

The output `Hello world` will appear in the console.

You can also explain the meaning of your code using *comments* starting with `#`

```
# begin a very simple program
print("Hello world")
# end this program
```

Variables

A variable is a named storage location that can hold a value. To create a variable in Python, simply assign a value to it using the equals sign (=). The value can be of any data type, such as a string, integer, or float.

```
name = "John Doe"
age = 30
```

```
gpa = 3.5
```

You can check type of a variable by `type()` function

```
print(type(name))
print(type(age))
print(type(gpa))
```

Data Types

Python has several built-in data types, including strings, integers, floats, and booleans.

Strings and f-String

A string is a sequence of characters enclosed in quotation marks. You can use either single quotes (') or double quotes (") to define a string.

```
name = "John Doe"
```

f-String allows you to insert a computation into a string.

```
print(f"The sum of 2 and 3 is {2 3}")
```

Integers

An integer is a whole number, either positive or negative, without a decimal point.

```
age = 30
```

Floats

A float is a number with a decimal point.

```
gpa = 3.5
```

Booleans

A boolean value can only be either `True` or `False`.

pythonCopy code

```
is_student = True
```

None

`None` means nothing.

```
variable_x = None
```

Control Structures

Control structures are used to control the flow of execution of a program. Python has several control structures, including `if` statements and `for` loops.

If Statements

An `if` statement allows you to execute a block of code only if a certain condition is met.

```
if age >= 18:
    print("You are an adult.")
```

For Loops

A `for` loop allows you to repeat a block of code for a specified number of times.

```
for i in range(10):
    print(i)
```

While Loops

A `while` loop allows you to repeat a block of code if your condition remains true.

```
i = 0
while i<10:
    print(i)
    i=i+2
```

Functions

A function is a reusable block of code that performs a specific task. Functions can take arguments (inputs) and return a value. In Python, you can define a function using the `def` keyword.

```
def greet(name):
    print("Hello, " + name)

greet("John Doe")
```

Tuples

A tuple is an ordered, immutable collection of elements of any data type. You can define a tuple by enclosing its elements in parentheses and separating them by commas.

makefileCopy code

```
point = (1, 2)
```

Lists

A list is an ordered, mutable collection of elements of any data type. You can define a list by enclosing its elements in square brackets and separating them by commas.

cssCopy code

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

Dictionaries

A dictionary is an unordered, mutable collection of key-value pairs. You can define a dictionary by enclosing its elements in curly braces and separating each key-value pair with a colon.

makefileCopy code

```
person = {"name": "John Doe", "age": 30}
```

Classes

A class is a blueprint for creating objects which can have attributes (data) and methods (functions). In Python, you can define a class using the `class` keyword.

```
class Student:
    def __init__(self, name, age, gpa):
        self.name = name
        self.age = age
        self.gpa = gpa

student = Student("John Doe", 30, 3.5)
print(student.name, student.age)
```

In the example above, the `__init__` method is a special method that is called when an object is created from the class. The `self` parameter refers to the object being created, and the `name`, `age`, and `gpa` parameters are attributes that can be accessed using dot notation (e.g., `student.name`).

You can also add additional methods to a class, which can perform actions or return values related to the object.

```
class Student:
    def __init__(self, name, age, gpa):
        self.name = name
        self.age = age
        self.gpa = gpa

    def is_honors(self):
        if self.gpa >= 3.5:
            return True
        else:
            return False

student = Student("John Doe", 30, 3.5)
if student.is_honors():
    print(student.name + " is in honors.")
```