

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

Example

Example

```
if 5 > 2:  
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

Syntax Error:

```
if 5 > 2:  
print("Five is greater than two!")
```

The number of spaces is up to you as a programmer, the most common use is four, but it has to be at least one.

Example

```
if 5 > 2:  
    print("Five is greater than two!")  
if 5 > 2:  
    print("Five is greater than two!")
```

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

Example

Syntax Error:

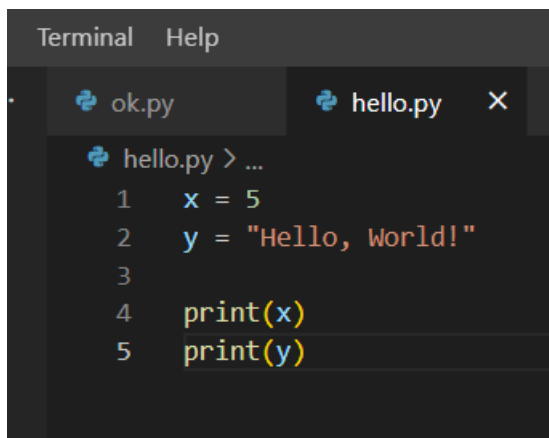
```
if 5 > 2:
print("Five is greater than two!")
    print("Five is greater than two!")
```

Python Variables

In Python, variables are created when you assign a value to it:

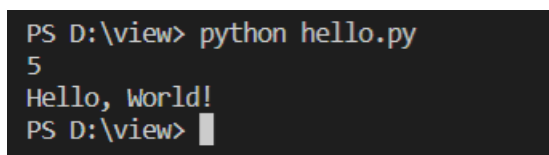
Example:- save file hello.py with following code.

Variables in Python:



```
Terminal  Help
ok.py  hello.py  X
hello.py > ...
1  x = 5
2  y = "Hello, World!"
3
4  print(x)
5  print(y)
```

And in terminal type to see output as shown below:-



```
PS D:\view> python hello.py
5
Hello, World!
PS D:\view> |
```

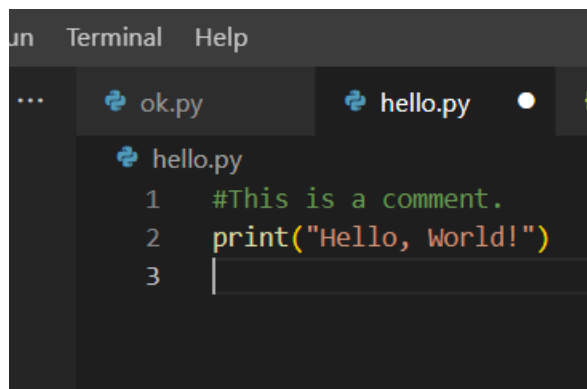
Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a `#`, and Python will render the rest of the line as a comment:

Example

Comments in Python:



```
un Terminal Help
... ok.py hello.py
hello.py
1 #This is a comment.
2 print("Hello, World!")
3
```

Built-in Data Types

In programming, data type is an important concept.

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

Python has the following data types built-in by default, in these categories:

Text Type: `str`

Numeric Types: `int`, `float`, `complex`

Sequence Types: `list`, `tuple`, `range`

Mapping Type: `dict`

Set Types: `set, frozenset`
Boolean Type: `bool`
Binary Types: `bytes, bytearray, memoryview`
None Type: `NoneType`

Getting the Data Type

You can get the data type of any object by using the `type()` function:

Example

Print the data type of the variable x:

```
x = 5  
print(type(x))
```

Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

Save file `hello.py` and run it in terminal

```
#list example  
  
thislist = ["apple", "banana", "cherry"]  
  
print(thislist)  
  
#tuple example  
  
thistuple = ("apple", "banana", "cherry")  
  
print(thistuple)  
  
#set example  
  
thisset = {"apple", "banana", "cherry"}
```

```
print(thisset)

#dictionary example
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print(thisdict)

#array example

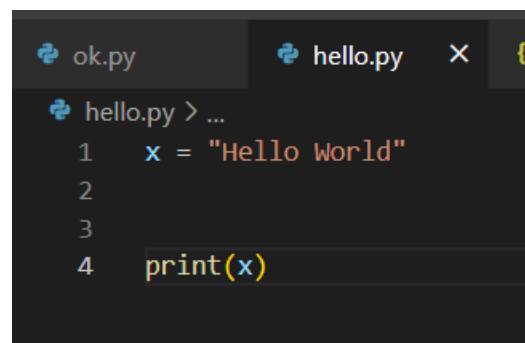
cars = ["Ford", "Volvo", "BMW"]

print(cars)
```

output:-

```
PS D:\view> python hello.py
['apple', 'banana', 'cherry']
('apple', 'banana', 'cherry')
set(['cherry', 'banana', 'apple'])
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
['Ford', 'Volvo', 'BMW']
PS D:\view> █
```

String data type example:-



```
ok.py hello.py X {
hello.py > ...
1 x = "Hello World"
2
3
4 print(x)
```

Integer data type example :-

```
ok.py hello.py {} launch
hello.py > ...
1 x = 20
2
3 #display x:
4 print(x)
5
6 #display the data type of x:
7 print(type(x)) |
```

Float data type example :-

```
ok.py hello.py {} launch.js
hello.py > ...
1 x = 20.5
2
3 #display x:
4 print(x)
5
6 #display the data type of x:
7 print(type(x)) |
```

Python Casting

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

- `int()` - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
- `float()` - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- `str()` - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Integers example:

```
x = int(1)    # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Floats example :

```
x = float(1)    # x will be 1.0
y = float(2.8)  # y will be 2.8
z = float("3")  # z will be 3.0
w = float("4.2") # w will be 4.2
```

Strings example:

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the `print()` function:

Example

```
print("Hello")  
print('Hello')
```

Python Operators

Operators are used to perform operations on variables and values.

In the example below, we use the `+` operator to add together two values:

Example

```
print(10 + 5)
```

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators