

Python Basics

What is Python:

What is Python?

Python is a programming language: this includes the "library" of data we can use and a piece of software called an "interpreter" that reads and executes the code we create. The name Python comes from the surreal British comedy group "Monty Python" due to the creator of the language, Guido van Rossum, being a fan of the show: He wanted his language to be "fun" to use.

Why is python used?

Due to its flexibility Python can be used for simple scripting or creating complex enterprise applications.

Why use Python?

- Simplicity
- Open Source
- High level Language
- Interpreted
- Object-oriented and functional
- Portable
- Vast collection of third party libraries that integrate with the base code library

Real world Application

Python has a rich history of projects that were either developed in the language or support its use in extensions/plugins. Here is a curated list of some of these areas of development

- Video game development
- GUI development
- Plugin development
- Machine Learning
- Big Data processing
- Scripting

Download Python:

<https://www.python.org/>

Python Basics

Download Visual studio code:

<https://code.visualstudio.com/Download>

Python Syntax:

General Naming Conventions

- Python has three standard naming conventions for working with different kinds of data: Screaming Snake Case, Snake Case, and Pascal Case.
- Screaming Snake Case
 - must start with an uppercase letter
 - all applicable characters must be upper case
 - individual words should be separated by an underscore
 - example:
 - SCREAMING_SNAK3_CASE
- Snake Case
 - must start with a lower case letter
 - individual words should be separated by an underscore
 - example:
 - snak3_case
- Pascal Case
 - must start with an upper case letter
 - individual words should be separated by a starting uppercase letter
 - example:
 - PascalCase

Each of these naming conventions is used to represent different kinds of data or meanings in your code.

- Screaming Snake case is used when working with constant value (think pi) in your code
- Pascal Case is used when working with classes
- Snake Case is used in all other cases

Formatting Conventions:

Unlike many other languages, Python uses indentation to represent specific interactions and ownership in code. The main way you will see this play out in your code is how the code is indented. Consider the following code that saves two names for later use

Python Basics

```
name = "Bill Teddington"  
other_name = "Sally Susington"
```

The code above can be executed fine. However, if adjusted slightly, the code will break

```
name = "Bill Teddington"  
    other_name = "Sally Sussington"
```

Notice how the second line is indented: this will cause your program to not work. As you work through the modules make sure you pay attention to the syntax of the examples.

Comments:

Any text after a "#" symbol is ignored by the Python interpreter; this is the simplest way to create comments. You can wrap your comments in triple single or double quote characters (' or ") to create multi line comments.

Single Comment

To create a single line comment you simply add a "#" symbol to the line. The comment is any content after the symbol, so you can write some code before the comment if you want, but not after.

```
# This is a comment  
name = "Ted" # this is also a comment, the name variable will still be set
```

Multi-Line Comment

If you need a comment to span multiple lines you can wrap it in triple single or double quotes.

```
"""this is a  
multi-line  
comment"""
```

OR

```
"""  
you could also
```

Python Basics

```
do your comment  
this way  
"""
```

Variables and Datatypes:

Variables in Python are used to store and manipulate data. A variable is essentially a named storage location in memory where you can store values.

Variables:

Named storage locations used to hold data during program execution.

Data Types:

Different types of data that can be stored in variables, such as integers, floats, strings, booleans, lists, tuples, dictionaries, and more.

Fundamental Data Types:

Basic data types in Python include integers, floats, strings, booleans, and None.

Composite Data Types:

Composite or collection data types include lists, tuples, sets, and dictionaries, used for storing collections of data.

Type Conversion:

Python allows implicit and explicit type conversion between different data types using built-in functions and constructors.

Variable Naming:

Variables in Python should follow naming conventions, such as using descriptive names, avoiding reserved words, and using lowercase with underscores for readability (snake_case).

Variable Declaration and Assignment:

Integer Variable:

```
age = 30
```

Float Variable:

Python Basics

```
height = 1.75
```

String Variable:

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

Boolean Variable:

```
is_valid = True
```

Data Type Inference:

Python infers data types automatically:

```
num = 42 # Integer  
pi = 3.14 # Float  
message = "Hello" # String
```

Type Conversion:

Implicit Type Conversion:

```
num_int = 42  
num_float = num_int + 0.5 # Implicitly converts integer to float
```

- Implicit type conversion occurs when different data types are combined in operations. Here, `num_int` is implicitly converted to a float when added to 0.5.

Explicit Type Conversion:

```
num_str = "123"  
num_int = int(num_str) # Explicitly converts string to integer
```

- Explicit type conversion is done using built-in functions like `int()`, `float()`, `str()`, etc. Here, `num_str` is converted from a string to an integer using `int()`.

Python Basics

Variable Naming:

Following naming conventions:

```
first_name = "Alice" # snake_case for variable names
num_attempts = 5 # descriptive variable names
```

- Variable names should follow conventions like using lowercase letters and underscores for readability (snake_case). Descriptive names like first_name and num_attempts enhance code clarity.

Checking Data Types:

Using the type() function:

```
num = 42
print(type(num)) # Output: <class 'int'>
```

The `type()` function is used to determine the data type of a variable. Here, it confirms that num is of type integer (int).

Composite Data Types:

Lists:

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

- Lists are used to store collections of items. Here, numbers is a list containing integers from 1 to 5.

Tuples:

```
point = (10, 20)
```

- Tuples are similar to lists but are immutable (cannot be changed). point is a tuple containing coordinates (10, 20).

Dictionaries:

```
person = {"name": "Alice", "age": 30}
```

- Dictionaries store key-value pairs. person is a dictionary with keys "name" and "age", mapping to values "Alice" and 30, respectively.

Python Basics

Sets:

```
unique_numbers = {1, 2, 3, 4, 5}
```

- Sets are unordered collections of unique items. `unique_numbers` is a set containing unique integers from 1 to 5.

Operators:

Arithmetic Operators

```
+ # addition  
- # subtraction  
* # multiplication  
/ # division  
** # power of  
% # modulus  
// # floor division
```

Assignment Operators

```
= # assigns a value to a variable  
+= # adds to the existing variable's value  
-= # subtracts from the existing variable's value  
*= # multiplies the existing variable's value by a value  
/= # divides the existing variable's value by a value  
%= # returns the modulus of the existing variable value  
//= # returns floor division with the existing variable value  
**= # raises the existing variable's value to a power
```

Comparison Operators

```
!= # checks if values are not equal  
== # checks if values are equal  
> # checks if first value is greater than second  
< # checks if first value is less than second  
>= # checks if first value is greater than or equal to second  
<= # checks if first value is less than or equal to second
```

Logical Operators

```
is # checks if objects share a memory address  
is not # checks if objects do not share a memory address
```

Python Basics

and # this lets you require multiple logical checks to be true in order to continue code execution

or # this lets you specify multiple logical checks in which only one need be true to continue code execution

Membership Operators

in # returns True if the object is in a collection

not in # returns True if the object is not in a collection

User Input and Output

Input

```
user_input = input("Enter your name: ")
```

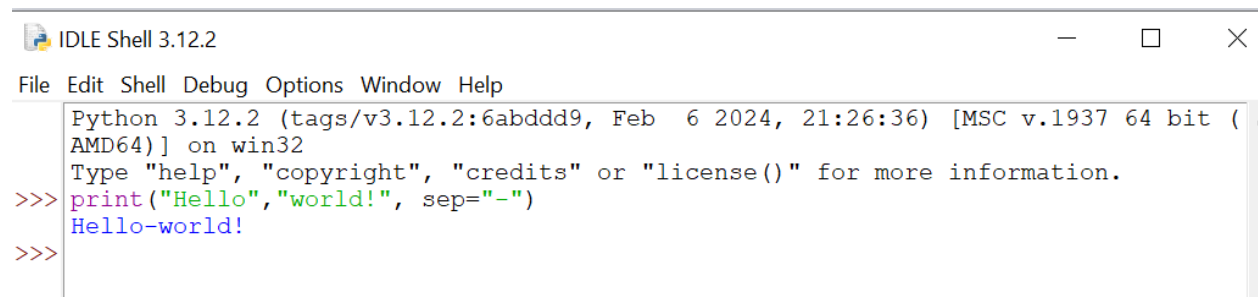
Print

To display data in the console you use the "print" function

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

If you want to change the separator between objects you can set a value called "sep" to your desired separator in the function arguments

```
print("Hello", "world!", sep="-")
```

A screenshot of the IDLE Shell 3.12.2 window. The title bar says "IDLE Shell 3.12.2". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell area shows the following text: "Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32", "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and the command prompt ">>>". The user has entered the command "print('Hello', 'world!', sep='-')", and the output "Hello-world!" is displayed. The prompt ">>>" is shown again on the next line.

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello", "world!", sep="-")
Hello-world!
>>>
```

Finally, the default behavior for the function is to end the line with a new line character, which you can change by setting an "end" value in the argument to your desired end character

```
print("Hello", "world", sep="-", end="!")
```


Python Basics

```
print("Hello", "world", sep="-", end="!")  
Hello-world!
```

Thank you