Python Object and Data Structure Basics

PIERIAN DATA

Basic Data Types

Complete Python 3 Bootcamp

- In this section of the course we will cover the key data types in Python.
- These are your basic building blocks when constructing larger pieces of code.
- Let’s quickly discuss all of the possible data types, then we’ll have lectures that go into more detail about each one!

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Complete Python 3 Bootcamp

Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

Let’s get started!

Numbers

Complete Python 3 Bootcamp

- There are two main number types we will work with:
    - Integers which are whole numbers.
    - Floating Point numbers which are numbers with a decimal.
- Let’s explore basic math with Python!
- We will also discuss how to create variables and assign them values.

Variable Assignments

Complete Python 3 Bootcamp

- We just saw how to work with numbers, but what do these numbers represent?
- It would be nice to assign these data types a variable name to easily reference them later on in our code!
- For example:
  - `my_dogs = 2`

Complete Python 3 Bootcamp

- **Rules for variable names**
  - Names can not start with a number.
  - There can be no spaces in the name, use _ instead.
  - Can't use any of these symbols :'",<>/?|\()!@#$%^&*~-+

Complete Python 3 Bootcamp

- Rules for variable names
  - It's considered best practice (PEP8) that names are lowercase.
  - Avoid using words that have special meaning in Python like "list" and "str"

Complete Python 3 Bootcamp

- Python uses **Dynamic Typing**
- This means you can reassign variables to different data types.
- This makes Python very flexible in assigning data types, this is different than other languages that are **“Statically-Typed”**

Complete Python 3 Bootcamp

my_dogs = 2

my_dogs = [“Sammy” , “Frankie”]

This is okay in Python!

Complete Python 3 Bootcamp

my_dogs = 2

my_dogs = [“Sammy” , “Frankie”]

ERROR in other Languages!

Complete Python 3 Bootcamp

```
int my_dog = 1;


my_dog = “Sammy” ;  //RESULTS IN ERROR
```

Example of Static Typing (C++)

Complete Python 3 Bootcamp

- Pros of Dynamic Typing:
  - Very easy to work with
  - Faster development time
- Cons of Dynamic Typing:
  - May result in bugs for unexpected data types!
  - You need to be aware of **type()**

Let’s explore these concepts!

Strings

Complete Python 3 Bootcamp

- Strings are sequences of characters, using the syntax of either single quotes or double quotes:
  - **'hello'**
  - **"Hello"**
  - **" I don't do that "**

Complete Python 3 Bootcamp

- Because strings are **ordered sequences** it means we can using **indexing** and **slicing** to grab sub-sections of the string.
- Indexing notation uses [ ] notation after the string (or variable assigned the string).
- Indexing allows you to grab a single character from the string...

Complete Python 3 Bootcamp

- These actions use [ ] square brackets and a number index to indicate positions of what you wish to grab.

Character :	**h**	**e**	**l**	**l**	**o**
Index :	**0**	**1**	**2**	**3**	**4**

Complete Python 3 Bootcamp

- These actions use [ ] square brackets and a number index to indicate positions of what you wish to grab.

Character :	**h**	**e**	**l**	**l**	**o**
Index :	**0**	**1**	**2**	**3**	**4**
Reverse Index:	**0**	**-4**	**-3**	**-2**	**-1**

Complete Python 3 Bootcamp

- Slicing allows you to grab a subsection of multiple characters, a “slice” of the string.
- This has the following syntax:
  - **[start:stop:step]**
- **start** is a numerical index for the slice start

Complete Python 3 Bootcamp

- Slicing allows you to grab a subsection of multiple characters, a “slice” of the string.
- This has the following syntax:
  - **[start:stop:step]**
- **start** is a numerical index for the slice start
- **stop** is the index you will go up to (but not include)
- **step** is the size of the “jump” you take.

Let’s explore these concepts!

String Indexing and Slicing

String Properties and Methods

String Formatting for Printing

Complete Python 3 Bootcamp

- Often you will want to “inject” a variable into your string for printing. For example:
  - **my_name = “Jose”**
  - **print(“Hello ” + my_name)**
- There are multiple ways to format strings for printing variables in them.
- This is known as string interpolation.

Complete Python 3 Bootcamp

- Let’s explore two methods for this:
  - **.format()** method
  - **f-strings** (formatted string literals)

Lists

Complete Python 3 Bootcamp

- Lists are ordered sequences that can hold a variety of object types.
- They use [] brackets and commas to separate objects in the list.
  - **[1,2,3,4,5]**
- Lists support indexing and slicing. Lists can be nested and also have a variety of useful methods that can be called off of them.

Dictionaries

Complete Python 3 Bootcamp

- Dictionaries are unordered mappings for storing objects. Previously we saw how lists store objects in an ordered sequence, dictionaries use a key-value pairing instead.
- This key-value pair allows users to quickly grab objects without needing to know an index location.

Complete Python 3 Bootcamp

- Dictionaries use curly braces and colons to signify the keys and their associated values.

{'key1':'value1','key2':'value2'}

- So when to choose a list and when to choose a dictionary?

Complete Python 3 Bootcamp

- **Dictionaries:** Objects retrieved by key name.

Unordered and can not be sorted.

- **Lists:** Objects retrieved by location.

Ordered Sequence can be indexed or sliced.

Tuples

Complete Python 3 Bootcamp

Tuples are very similar to lists. However they have one key difference - **immutability.**

Once an element is inside a tuple, it can not be reassigned.

Tuples use parenthesis: **(1,2,3)**

Sets

Complete Python 3 Bootcamp

Sets are unordered collections of **unique** elements.

Meaning there can only be one representative of the same object.

Let’s see some examples!

Booleans

Complete Python 3 Bootcamp

Booleans are operators that allow you to convey **True** or **False** statements.

These are very important later on when we deal with control flow and logic!

Files

Complete Python 3 Bootcamp

Before we finish this section, let's quickly go over how to perform simple I/O with basic .txt files.

We'll also discuss file paths on your computer.

Let's get started!

Objects and Data Structures Assessment Test

PIERIAN DATA

Complete Python 3 Bootcamp

Let’s have a quick overview of your first test.

You can download the notebooks from GitHub or as a zip file from the Course Overview Lecture.

Objects and Data Structures Assessment Test SOLUTIONS

PIERIAN DATA

Complete Python 3 Bootcamp

- Numbers: Store numerical information and come in two forms:
  - Integers - Whole Numbers
  - Floating Point - Numbers with a decimal

Complete Python 3 Bootcamp

- Strings: Ordered sequence of characters
- Lists: Ordered sequence of objects (mutable)
- Tuples: Ordered sequence of objects (immutable)
- Dictionary: Key-Value pairing that is unordered.

Python Documentation