python data types

python data types

- 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!

python data types

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**

python data types

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**

python data types

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**

python data types

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**

python data types

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**

python data types

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**

python data types

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**

python data types

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**

python data types

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**

python data types

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**

numbrs

`python data types - numbers`

- There are two main number types we will work with:
  - Integers which are whole numbers.
  - Floating Point numbers which are numbers with a decimal.

Numbers and more in Python!

In this lecture, we will learn about numbers in Python and how to use them.

We'll learn about the following topics:

1.) Types of Numbers in Python
2.) Basic Arithmetic
3.) Differences between classic division and floor division
4.) Object Assignment in Python

variable assignments

python data types - variable assignments

- 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**

python data types - variable assignments

- **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 :'",<>/?|\()!@#$%^&*~-+

python data types - variable assignments

- 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"

python data types - variable assignments

- 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”**

`python data types - variable assignments`

- 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()**

python data types - variable assignments

```
int my_dog = 1;

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

Example of Static Typing
(C++)

python data types - variable assignments

```
my_dogs = 2
```

```
my_dogs = [ “Sammy” ,  “Frankie” ]
```

ERROR
in other
Languages!

python data types - variable assignments

```
my_dogs = 2
```

```
my_dogs = [ “Sammy” , “Frankie” ]
```

This is okay in Python!

Variable Assignment ¶

Rules for variable names

- names can not start with a number
- names can not contain spaces, use _ intead
- names can not contain any of these symbols:

```
:'",<>/?|\!@#%^&*~-+
```

- it's considered best practice (PEP8) that names are lowercase with underscores
- avoid using Python built-in keywords like `list` and `str`
- avoid using the single characters `l` (lowercase letter el), `O` (uppercase letter oh) and `I` (uppercase letter eye) as they can be confused with `1` and `0`

strings

python data types - strings

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

python data types - strings

- 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...

python data types - strings

- 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

python data types - strings

- 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.

Strings

Strings are used in Python to record text information, such as names. Strings in Python are actually a *sequence*, which basically means Python keeps track of every element in the string as a sequence. For example, Python understands the string "hello' to be a sequence of letters in a specific order. This means we will be able to use indexing to grab particular letters (like the first letter, or the last letter).

This idea of a sequence is an important one in Python and we will touch upon it later on in the future.

In this lecture we'll learn about the following:

```
1.) Creating Strings
2.) Printing Strings
3.) String Indexing and Slicing
4.) String Properties
5.) String Methods
6.) Print Formatting
```

lists

python data types - lists

- 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.

Lists

Earlier when discussing strings we introduced the concept of a *sequence* in Python. Lists can be thought of the most general version of a *sequence* in Python. Unlike strings, they are mutable, meaning the elements inside a list can be changed!

In this section we will learn about:

```
1.) Creating lists
2.) Indexing and Slicing Lists
3.) Basic List Methods
4.) Nesting Lists
5.) Introduction to List Comprehensions
```

Lists are constructed with brackets [] and commas separating every element in the list.

Let's go ahead and see how we can construct lists!

dictionaries

python data types - dictionaries

- 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.

python data types - dictionaries

- 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?

python data types - dictionaries

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

Unordered and can not be sorted.

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

Ordered Sequence can be indexed or sliced.

Dictionaries

We've been learning about *sequences* in Python but now we're going to switch gears and learn about *mappings* in Python. If you're familiar with other languages you can think of these Dictionaries as hash tables.

This section will serve as a brief introduction to dictionaries and consist of:

```
1.) Constructing a Dictionary
2.) Accessing objects from a dictionary
3.) Nesting Dictionaries
4.) Basic Dictionary Methods
```

So what are mappings? Mappings are a collection of objects that are stored by a *key*, unlike a sequence that stored objects by their relative position. This is an important distinction, since mappings won't retain order since they have objects defined by a key.

A Python dictionary consists of a key and then an associated value. That value can be almost any Python object.

tuples

python data types - tuples

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)**

ave written a note here.

Tuples

In Python tuples are very similar to lists, however, unlike lists they are *immutable* meaning they can not be changed. You would use tuples to present things that shouldn't be changed, such as days of the week, or dates on a calendar.

In this section, we will get a brief overview of the following:

```
1.) Constructing Tuples
2.) Basic Tuple Methods
3.) Immutability
4.) When to Use Tuples
```

You'll have an intuition of how to use tuples based on what you've learned about lists. We can treat them very similarly with the major distinction being that tuples are immutable.

sets

`python data types - sets`

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

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

Let’s see some examples!

booleans

python data types - strings

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!

Set and Booleans

There are two other object types in Python that we should quickly cover: Sets and Booleans.

Sets

Sets are an unordered collection of *unique* elements. We can construct them by using the set() function. Let's go ahead and make a set to see how it works