CEI.

Python Data Types

CEI.

Python Data Types

CEI.

- 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

CEI.

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

CEI.

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

CEI.

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

CEI.

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

CEI.

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

CEI.

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

CEI.

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

CEI.

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

CEI.

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

CEI.

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 - Numbers

CEI.

Python Data Types - Numbers

CEI.

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


jupyter

Python Data Types - Variable Assignments

CEI.

Python Data Types - Variable Assignments

CEI.

- 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

CEI.

- **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

CEI.

- 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

CEI.

- 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

CEI.

- 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

CEI.

```
int my_dog = 1;
```

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

Example of Static Typing
(C++)

Python Data Types - Variable Assignments

CEI.

```
my_dogs = 2
```

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

ERROR
in other
Languages!

Python Data Types - Variable Assignments

CEI.

```
my_dogs = 2
```

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

This is okay in Python!


jupyter

Python Data Types - Strings

CEI.

Python Data Types - Strings

CEI.

- 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

CEI.

- 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...

- 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**

- 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.


jupyter

Python Data Types - Lists

CEI.

Python Data Types - Lists

CEI.

- 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.


jupyter

Python Data Types - Dictionaries

CEI.

Python Data Types - Dictionaries

CEI.

- 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

CEI.

- 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

CEI.

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

Unordered and can not be sorted.

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

Ordered Sequence can be indexed or sliced.


jupyter

Python Data Types - Tuples

CEI.

Python Data Types - Tuples

CEI.

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)**


jupyter

Python Data Types - Sets

CEI.

Python Data Types - Sets

CEI.

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

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

Let's see some examples!

Python Data Types - Booleans

CEI.

Python Data Types - Booleans

CEI.

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!

c.