# Predictive Analytics Foundations

Lecture 3

# Lecture 3

Data Types

- **Numbers, Strings, and Booleans**
- Types
- Lists, Tuples, Sets, and Dictionaries

# Numbers

(Demo)

# Ints and Floats

Python has two real number types

- **int**: an integer of any size
- **float:** a number with an optional fractional part

# Ints and Floats

An **int** never has a decimal point; a **float** always does

A **float** might be printed using scientific notation

Three limitations of float values:

- They have limited size (but the limit is huge)
- They have limited precision of 15-16 decimal places
- After arithmetic, the final few decimal places can be wrong

# Strings

(Demo)

# Text and Strings

A string value is a snippet of text of any length

- 'a'
- 'word'
- "there can be 2 sentences. Here's the second!"

# Text and Strings

Strings consisting of numbers can be converted to numbers

- int('12')
- float('1.2')

Any value can be converted to a string

- str(5)

# Discussion Question

## Assume you have run the following statements:

- x = 3
- y = '4'
- z = '5.6'

## What's the source of the error in each example?

- x + y
- x + int(y + z)
- str(x) + int(y)
- y + float(z)

# Lecture 3

Data Types

- Numbers, Strings, and Booleans
- **Types**
- Lists, Tuples, Sets, and Dictionaries

# Types

(Demo)

# Every value has a type

- We’ve seen 3 types so far:
  - int: 2
  - float: 2.2
  - str: 'Identity theft is not a joke'
- The type function can tell you the type of a value
  - type(2)
  - type(2 + 2)
- An expression’s “type” is based on its value, not how it looks
  - x = 2
  - type(x)

# What Does “Type” Mean?

- In Python variables, literals, and constants have a “type”
- Python knows the difference between an integer number and a string
- For example, “+” means “addition” if something is a number and “concatenate” if something is a string

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

concatenate = put together

# Type Matters

Python knows what “type” everything is

Some operations are prohibited

You cannot “add 1” to a string

We can ask Python what type something is by using the type() function

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):  File "<stdin>", line 1, in
<module>TypeError: Can't convert 'int' object to str
implicitly

>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

# String Conversions

- You can also use int() and float() to convert between strings and integers
- You will get an error if  the string does not contain numeric characters

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str
implicitly
```

```
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>
ValueError: invalid literal for int() with
base 10: 'x'
```

# User Input

We can instruct Python to pause and read data from the user using the input() function

The input() function returns a string

```
name = input('Who are you? ')
print('Welcome', name)
```

# Converting User Input

If we want to read a number from the user, we must convert it from a string to a number using a type conversion function.

```
input = input('Europe floor?')
usf = int(input) + 1
print('US floor', usf)
```

```
Europe floor? 0
US floor 1
```

# Lecture 3

Data Types

- Numbers, Strings, and Booleans
- Types
- **Lists, Tuples, Sets, and Dictionaries**

# Lists & Tuples

(Demo)

# Lists

- Lists allow us to store multiple things (“elements”) in a single object.
- The elements are *ordered* (we’ll explore what that means a little later).
- Lists are defined with square brackets [ ].

# Tuples

- Tuples allow us to store multiple things (“elements”) in a single object.

- Tuples look similar to lists but has a key difference. They are immutable.

- Tuples are defined with parentheses ( ).

# Arrays

(Demo)

# Arrays

An array contains a sequence of values

- All elements of an array should have the same type
- Arithmetic is applied to each element individually
- Adding two arrays adds the corresponding elements (but the arrays must be the same length!)
- A column of a table is an array