Predictive Analytics Foundations

Lecture 4

Lecture 4

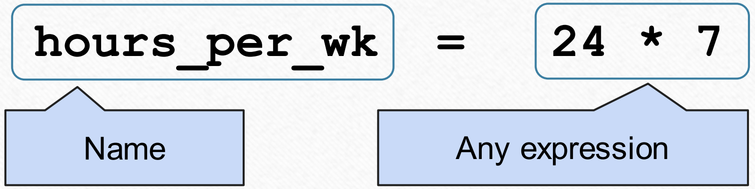
Tables

- **Review**
- Functions
- Array
- Files

Review of Python Concepts

- An expression evaluates to a value
- Values can be numbers or strings (text)
- The syntax (format) of the language is very rigid — even an extra space can cause a syntax error
- There is particular behavior associated with built-in operators that you need to learn
  (e.g., dividing produces 8.0 instead of 8)

Assignment Statements



```
hours_per_wk = 24 * 7
```

- An assignment statement changes the meaning of the name to the left of the = symbol
- The name is bound to the value of the expression to the right of the = symbol (its current value; not the equation)

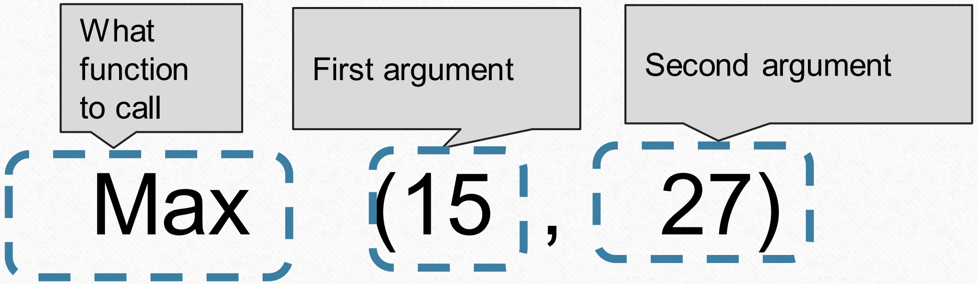
(Demo)

Lecture 4

Tables

- Review
- **Functions**
- Array
- Files

Anatomy of a Call Expression



(Demo)

Review of Function Concepts

- Some functions require a particular number of arguments (e.g., **abs** must be called on one value)
- Arguments can be named in the call expression:
  round(number=12.34)
  But the names must match the documentation
- Type a ? after a function name to see its documentation

Write a Function

- As a rule of thumb, if you anticipate needing to repeat the same or very similar code more than once, it may be worth writing a reusable function.

```
In [173]: def my_function(x, y):
     .....:        return x + y
```

(Demo)

Lecture 4

Tables

- Review
- Functions
- **Array**
- Files

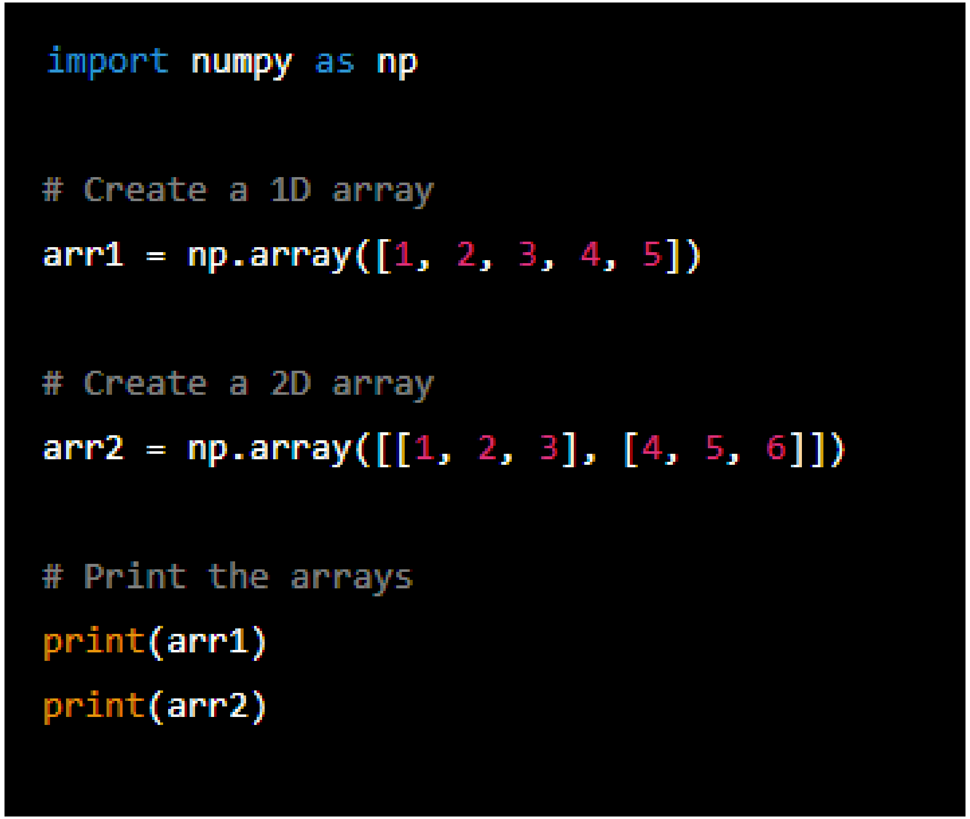
NumPy Basics: Arrays

- NumPy is a Python library for numerical computing. NumPy is widely used in scientific computing, data analysis, and machine learning.

- One of the key features of NumPy is its powerful array data structure, which provides efficient storage and manipulation of large numerical data sets. The array can be of any dimension, from a simple one-dimensional array to a multidimensional array, and can store elements of any numerical data type.

Arrays

One of the key features of NumPy is its N-dimensional array object, or ndarray, which is a fast, flexible container for large datasets in Python.



```
import numpy as np

# Create a 1D array
arr1 = np.array([1, 2, 3, 4, 5])

# Create a 2D array
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

# Print the arrays
print(arr1)
print(arr2)
```

Lecture 4

Tables

- Review
- Functions
- Array
- **Files**

Opening a File

Before we can read the contents of the file, we must tell Python which file we are going to work with and what we will be doing with the file

This is done with the open() function

Using open()

handle = open(filename, mode)

- returns a handle use to manipulate the file
- filename is a string
- mode is optional and should be 'r' if we are planning to read the file and 'w' if we are going to write to the file

Reading Files in Python

File Handle as a Sequence

A file handle open for read can be treated as a sequence of strings where each line in the file is a string in the sequence

We can use the for statement to iterate through a sequence

Remember - a sequence is an ordered set

```
xfile = open('poem.txt')

for cheese in xfile:
    print(cheese)
```

Counting Lines in a File

Open a file read-only

Use a for loop to read each line

Count the lines and print out the number of lines

```
fhand = open('mbox.txt')
count = 0
for line in fhand:
    count = count + 1
print('Line Count:', count)
```

Reading the *Whole* File

We can read the whole file (newlines and all) into a single string

```
>>> fhand = open('mbox-short.txt')
>>> inp = fhand.read()
>>> print(len(inp))
94626
```

Searching Through a File

We can put an if statement in the for loop to only print lines that meet some criteria

```
fhand = open('mbox-short.txt')
for line in fhand:
    if line.startswith('From:') :
        print(line)
```