Lecture 08: Python For Traditional Programmers

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

TABLE OF CONTENTS

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Introduction

- Highlight some differences
  - Interpreter vs compiler: program runs on-the-fly
  - No semicolon at the end of commands: new line character
  - No brackets for block of commands: indent is very important
- Highlight some advantages of Python for Business Intelligence
  - Very easy input / output text files
  - Very easy / flexible when working with collection (array, list, tuples)
  - Built-in data types for advanced collection (dictionaries)

Variables and data types

- Naming rules: same as other programming languages
- Declaration: no need to declare. Data type is based on value
  - a = 5 => int
  - b = 5.0 => float
  - c = “5” => string
- Flexible data type: changes as value changes
  - a = 5 # a is integer
  - a = “5” # a changes to string

Input

- Syntax: varName = input(“prompt string: “)
- Input a string
  - str = input(“Enter a string: “)
- Input an integer:
  - a = int(input(“Enter an integer: “)
- Input a float:
  - x = float(intput(“Enter a float: “)

Output

- Function print
  - Syntax: print(varName)
- Format output
  - Similar to C (Java)

```
str1 = "hello"
str2 = "world"

print('[{0:10s} {1:>10s}]'.format(str2, str1))
```

```
print('E. {0:d} {1:d}'.format(int1, int2))
print('F. {0:8d} {1:10d}'.format(int1, int2))
print(' ***')
print('G. {0:0.3f}'.format(float1))   # 3 decimal places
print('H. {0:6.3f}'.format(float1))   # 6 spaces, 3 decimals
print('I. {0:8.3f}'.format(float1))   # 8 spaces, 3 decimals
```

Collection: list

- Lists are defined by a pair of *square* brackets
  - Elements can be of any types
  - Indexing: start from 0

```
a = [0, 1, 1, 2, 3, 5, 8, 13]
b = [5., "girl", 2+0j, "horse", 21]
```

  - Concatenate 2 lists: c = a + b

```
[0, 1, 1, 2, 3, 5, 8, 13, 5.0, 'girl', (2+0j), 'horse', 21]
```

Collection: list

- Slicing list: You can access pieces of lists using the slicing feature

```
In [18]: b
Out[18]: [10.0, 'girls & boys', (2+0j), 3.14159, 21]

In [19]: b[1:4]
Out[19]: ['girls & boys', (2+0j), 3.14159]

In [20]: b[3:5]
Out[20]: [3.14159, 21]
```

Collection: list

- Slicing list: You can access pieces of lists using the slicing feature

```
In [18]: b
Out[18]: [10.0, 'girls & boys', (2+0j), 3.14159, 21]

In [21]: b[2:]
Out[21]: [(2+0j), 3.14159, 21]

In [22]: b[:3]
Out[22]: [10.0, 'girls & boys', (2+0j)]

In [23]: b[:]
Out[23]: [10.0, 'girls & boys', (2+0j), 3.14159, 21]
```

Collection: list

- Slicing list: How to get up to (exclusive) last element of the list?
  - Using len function:
  - last = len(b)
  - b[5:last-1]

  - More fancy way:
  - b[5:-1]

Collection: tuple

- Tuples are lists that are *immutable :* the individual elements of a tuple cannot be changed

```
In [43]: c = (1, 1, 2, 3, 5, 8, 13)
In [44]: c[4]
Out[44]: 5

In [45]: c[4] = 7
```

Collection: multi-dimensional lists & tuples

- We can make multidimensional lists, or lists of lists (same as tuples, but immutable).
  - Here is a list of 3 lists

```
In [40]: a = [[3, 9], [8, 5], [11, 1]]
```

```
In [47]: a[1]
Out[47]: [8, 5]

In [48]: a[1][0]
Out[48]: 8
```

  - Can have list of different length lists a = [[1], [2, 3, 4], [5, 6]]

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Exercises

1. Create a list of 20 integers

a. Slice 1st half and 2nd half of the list

b. Slice list to get a sublist that removes n elements at begin and n elements at end of list (n from keyboard)

c. With n from keyboard, get n first elements and n last elements, join them to make a new list of 2n elements

d. With n from keyboard, get nth element from list. What happen if n is out of index range?

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Exercise

2. Create a 5 x 5 list of int numbers

a) Get 3 middle rows in as many ways as you can

b) Get 2 last rows in as many ways as you can

c) Enter n from keyboard, get max, min of n^{th} row and print them (hint: using function max, min)

d) Enter n from keyboard, get sum of n^{th} row and print it (hint: using function sum)

e) Enter n from keyboard, sort n^{th} row and print it (hint: using function sorted)

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Collection: numpy array

- “NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more”. – scipy.org
- Install NumPy: ask Mr. Google
- Using NumPy: import numpy as np

Collection: numpy array

- For more complicated operations on data, using array is much more convenient than using list
- Array vs List
  - Elements of a NumPy array must all be of the same type
  - Arrays allow Boolean indexing; lists do not
  - NumPy arrays support “vectorized” operations like element-by-element addition and multiplication
  - Adding one or more additional elements to a NumPy array creates a new array and destroys the old one.

Collection: numpy array

- Array creating

```
In [1]: a = [0, 0, 1, 4, 7, 16, 31, 64, 127]

In [2]: import numpy as np

In [3]: b = np.array(a)

In [4]: b
Out[4]: array([ 0, 0, 1, 4, 7, 16, 31, 64, 127])

In [5]: c = np.array([1, 4., -2, 7])

In [6]: c
Out[6]: array([ 1., 4., -2., 7.])
```

Collection: numpy array

- Array creating: Using linspace function
  - Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.linspace.html

```
In [7]: np.linspace(0, 10, 5)
Out[7]: array([ 0. , 2.5, 5. , 7.5, 10. ])
```

  - Some examples:
    - np.linspace(0,10, 20, endpoint = False)
    - a, step = np.linspace(0, 10, 20, retstep = True)

Collection: numpy array

- Array creating: Using logspace function

```
lg10 = np.logspace(0, 3, 4)
print(lg10)
lg2 = np.logspace(1, 10, 10, base = 2)
print(lg2)
```

```
[   1.   10.  100. 1000.]
[   2.    4.    8.   16.   32.   64.  128.  256.  512. 1024.]
```

Collection: numpy array

- Array creating: Using arange function (similar to range function)
  - Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.arange.html

```
In [10]: np.arange(0, 10, 2)
Out[10]: array([0, 2, 4, 6, 8])

In [11]: np.arange(0., 10, 2)
Out[11]: array([ 0., 2., 4., 6., 8.])

In [12]: np.arange(0, 10, 1.5)
Out[12]: array([ 0. , 1.5, 3. , 4.5, 6. , 7.5, 9. ])
```

Collection: numpy array

- Array creating: Using zeros and ones function

```
In [13]: np.zeros(6)
Out[13]: array([ 0., 0., 0., 0., 0., 0.])

In [14]: np.ones(8)
Out[14]: array([ 1., 1., 1., 1., 1., 1., 1., 1.])

In [15]: ones(8, dtype=int)
Out[15]: np.array([1, 1, 1, 1, 1, 1, 1, 1])
```

Mathematical operation with arrays

- Element-by-element operation

```
In [16]: a = np.linspace(-1., 5, 7)

In [17]: a
Out[17]: array([-1., 0., 1., 2., 3., 4., 5.])

In [18]: a*6
Out[18]: array([ -6., 0., 6., 12., 18., 24., 30.])
```

– Try: a+2, a*a, a**2, (a+3)/2, np.sin(a)

Mathematical operation with arrays

- Element-by-element operation

```
In [40]: a-b
Out[40]: array([-34., -17., -15.])

In [41]: a*b
Out[41]: array([ 2312., -60., 100.])

In [42]: a/b
Out[42]: array([ 0.5 , -2.4 , 0.25])
```

  – Try: a+2/b, (a+2)/b, a + np.sin(b)

Array: Boolean indexing

- Who needs for and if?

```
In [52]: b = 1.0/np.arange(0.2, 3, 0.2)

In [53]: b
Out[53]:
array([ 5.        ,  2.5       ,  1.66666667,  1.25      ,
        1.        ,  0.83333333,  0.71428571,  0.625     ,
        0.55555556,  0.5       ,  0.45454545,  0.41666667,
        0.38461538,  0.35714286])

In [54]: b[b > 1]
Out[54]:
array([ 5.        ,  2.5       ,  1.66666667,  1.25      ])
```

Array: Boolean indexing

- Cleaning data

```
In [62]: y = np.sin(np.linspace(0, 4*np.pi, 9))

In [63]: y
Out[63]:
array([  0.00000000e+00,   1.00000000e+00,   1.22464680e-16,
        -1.00000000e+00,  -2.44929360e-16,   1.00000000e+00,
         3.67394040e-16,  -1.00000000e+00,  -4.89858720e-16])

In [64]: y[np.abs(y) < 1.e-15] = 0

In [65]: y
Out[65]: array([ 0.,  1.,  0., -1.,  0.,  1.,  0., -1.,  0.])
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Exercise

1. Create array of 10 integers from 0 to 20 (exclusive), equally spacing by using linspace
2. Create array of 10 integers from 0 to 20 (exclusive), equally spacing by using arange
3. When should we use linspace vs arange?
4. Create array of 20 random integers (min = 0, max = 99)
   a) https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.random.randint.html
   b) Get sub-array from that, contains only even numbers, sort them ascending
   c) Re-arrange array created from a) so that 1st half contains only even numbers, 2nd half contains only odd numbers (hint: use concatenate function of numpy), each half is sorted

exercise

5. Suppose, for example, that we have two arrays y, and t for position *vs.* time of a falling object
   a) y = np.array([0., 1.3, 5. , 10.9, 18.9, 28.7, 40.])
   b) t = np.array([0., 0.49, 1. , 1.5 , 2.08, 2.55, 3.2])
   c) Calculate the velocity as a function of time

$$v_i = \frac{y_i - y_{i-1}}{t_i - t_{i-1}}$$

exercise

6. Create 2 arrays of same size for names and ages, including some empty names, invalid ages (less than 0, over 150)
   a) For any empty name, change it to “John Doe”
   b) For any invalid age, correct it to 0 or 150 correspondingly
   c) Find max age, min age
   d) *Find longest name, shortest name
   e) **Find length of longest name, length of shortest name (without knowing which it is)

Multi-dimensional arrays

- Creating matrix from list

```
In [66]: b = np.array([[1., 4, 5], [9, 7, 4]])

In [67]: b
Out[67]: array([[ 1.,  4.,  5.],
                [ 9.,  7.,  4.]])
```

- Reshape from 1d array

```
In [71]: c = np.arange(6)

In [72]: c
Out[72]: array([0, 1, 2, 3, 4, 5])

In [73]: c = np.reshape(c, (2, 3))
```

Indexing and slicing

- Indexing: same as list

```
In [75]: b[0][2]
Out[75]: 5.0
```

or new way

```
In [76]: b[0, 2]
Out[76]: 5.0
```

- Slicing: same as list
  column

```
a =
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
a[1] =  [4 5 6]
```

but can slide

```
a =
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
a[:,1] =  [2 5 8]
```

Dictionaries

- A dictionary is also a collection of objects that is indexed by strings or numbers

```
room = {"Emma":309, "Jake":582, "Olivia":764}
```

  - room.keys()
  - room.values()

```
keys =  ['Emma', 'Jake', 'Olivia']
values =  [309, 582, 764]
```

- Access key to get value

```
In [2]: room["Olivia"]
Out[2]: 764
```

- Other methods: clear(), copy(), pop(), update(), …
  - https://www.w3schools.com/python/python_dictionaries.asp

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Exercise

1. Perform the following tasks with NumPy arrays.
   a) Create an 8 × 8 array with ones on all the edges and zeros everywhere else.
   b) Create an 8 × 8 array of integers with a checkerboard pattern of ones and zeros.
   c) Create an 8 x 8 array of random integers (0, 99), make all the numbers not divisible by 3 negative.
   d) Split the matrix above into 4 submatrices 4x4
   e) Find the size, shape, mean, and standard deviation of the matrices you created in part d

UNIVERSITY of GREENWICH
Alliance with FPT Education

Pearson BTEC

Exercise

2. Create a dictionary with keys as student names and values as student GPA (float)
   a) Enter some new items (key, value) to the dictionary
   b) Print all items of the dictionary
   c) Find student who has max GPA
   d) Find student who has min GPA
   e) Find mean GPA of students
   f) Enter a student name, print his/her GPA
   g) Enter a student name, a new GPA and update it
   h) Enter a student name, remove him/her from the dictionary

Loop: For

- The for loop syntax
- Loop through a list

```
for <itervar> in <sequence>:
    <body>
```

```
for dogname in ["Molly", "Max", "Buster", "Lucy"]:
    print(dogname)
    print(" Arf, arf!")
print("All done.")
```

- Loop through a range

```
s = 0
for i in range(1, 100, 2):
    print(i, end=' ')
    s = s+i
print('\n{}'.format(s))
```

Loop: for

- Loop through a list of pairs

```
room = {"Emma": 309, "Jake": 582, "Olivia": 764}

lp = list(room.items())
print(lp)

for k, v in lp:
    if v > 500:
        print(k, end=" ")
```

```
[('Emma', 309), ('Jake', 582), ('Olivia', 764)]
Jake Olivia
```

Loop: for

- Loop through a string

```
a = 'There are places I remember all my life'
```

```
for letter in a:
    print(letter)
```

- Using enumerate function (which generates a list of pairs)

```
for i, letter in enumerate(a):
        if i % 3 == 0:
            print(letter, end=' ')
```

LOOP: FOR

- When should not use for?

```
a = np.linspace(0, 32, 10000000) # 10 million

for i in range(len(a)):
    a[i] = a[i]*a[i]

print(a)
```

```
a = np.linspace(0, 32, 10000000) # 10 million
a = a*a

print(a)
```

100 times faster!

Loop: for

- List comprehension: Suppose we have a matrix

```
A = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

- How to get the diagonal from A?

```
diag = []
for i in [0, 1, 2]:
    diag.append(A[i][i])
```

Traditional for way

```
diagLC = [A[i][i] for i in [0, 1, 2]]
```

List comprehension way

How to slice a column by using list comprehension?

exercises

1. Create a **list** of 20 random integers less than 100
   a) Find elements that are greater than 50
   b) Find elements that are greater than mean of list
   c) Create a list of even numbers from that list
   d) Re-arrange the list so that 1st half contains only even numbers and 2nd half contains only odd numbers
   e) Create a list of even index numbers from that list, a list of odd index numbers from that list

exercises

2. Reuse the dictionary of (student, GPA) from previous exercise
   a) Print Distinction students who have GPA > 8.0
   b) Print Merit students who have GPA > 6.5 but <= 8.0
   c) Print Pass students who have GPA >= 4.0 but <= 6.5
   d) Print Failed student who have GPA < 4.0
   e) Print numbers of Distinction students, Merit students, Pass students and Failed students
   f) Print pass rate and check if it is greater than 70% then print successful semester