Computer Systems and Networks

ECPE 170 – University of the Pacific

Python Introduction

Lab Schedule

Activities

- **This Week**
  - Python introduction
  - Networking introduction
  - Endianness
  - **Lab 8** (HTTP, TCP sockets)

Assignments Due

- **Lab 7**
  - **Due by OCT 21st 5:00am**
- **Lab 8**
  - **Due by Oct 28th 5:00am**

Python

What is Python?

- Interpreted language for scripting and many other uses
- Features:
  - Objects
  - Dynamic types
  - A rich set of libraries
  - Extensibility through C (for speed critical code)
- It is most notorious for its indentation rules, using whitespace or tabs (and it is *very picky*)

Python Datatypes

- Python supports many datatypes from C or C++:
  - Integers, floats, strings, booleans
- Recent Python versions support other useful types:
  - Complex numbers
  - Sequences (tuples, lists)
  - Dictionaries
  - Sets
  - Bytes and bytearrays

Runtime evaluation

- Python is interpreted and has dynamic typing
- Implications:
  - Syntax is checked when code is first encountered
  - Variable types (or even their existence) aren't checked until the code is executed
- Result: Code can execute correctly for a while until either an undefined variable is encountered, or it is used incorrectly (i.e., trying to access an integer as a sequence)

Python Tuples

- A *tuple* is an immutable collection of objects in a sequence
- Tuples are denoted by parenthesis

```
t = (1,2,3)
```

- The objects in a tuple do not need to be of the same type

```
t = (1,2,3, 'ECPE 170', 3.1415)
```

Problem 1 – Indexing Tuples

Open the terminal. Type `python3` to open the interpreter. Create a tuple:

```
t = (1,2,3,'ECPE 170 rocks!', 'bye')
```

Write the output for:

```
>>>print(t[0])  #Pound is for comment, btw
>>>print(t[3])
>>>print(t[7])
>>>print(t[-3]) # Access sequence from end using neg indices
>>>print(t[-8])
>>>t[2]=t[3]
```

P1

Problem 2 – Slicing Tuples

Slices (subsets of sequences) are accessed by using a ":"

What does the following print?

```
>>>t[2:4]
>>>t[0:4:2]
```

P2

Python Lists

- A *list* is a mutable collection of objects in a sequence
- Lists are denoted by square brackets

```
l = [1.5, 'a', (3,True)]
```

Problem 3 - Lists

Declare the list:

```
list = [1.5, 'a', (3,True)]
```

Write the output for the following operations:

a. `>>>list.append('Hello!')`
   `>>>print(list)`
b. Try slicing like in tuples to see if it works:
   `>>>list[1:3]`
c. `>>>list.insert(4,'Scarlett Popapill')`
   `>>>print(list)`
d. `>>>list.pop(-2)`

P3

Python Dictionaries

- A *dictionary* is an associative array of keys and value pairs

```
d={'a':1, 'b':2, 3:'c'}
print(d)
print(d.keys())
print(d.values())
print(d['a'])
print(d['c'])
```

Output:

```
{'a': 1, 3: 'c', 'b': 2}
dict_keys(['a', 3, 'b'])
dict_values([1, 'c', 2])
1
KeyError: 'c'
```

P4

Problem 5 – Strings Sequences

- String sequences are versatile with many built-in operations.

Perform the following and write the output:

a.

```
>>>string="Programming in C is "
>>>print(string)
```

b.

```
>>>string=string + "a lot of fun!" #concatenation
>>>print(string)
```

P5

Problem 6 – String Splitting

- ‘Split’ a string to divide it into a list based on a delimiter

```
<name of string>.split(delimiter,maxsplits)
```

- Returns a list of separated items
- `delimiter` is the delimiting sequence about which you would like to split.
- `maxsplits` is the number of splits to perform. The output list will have `maxsplits+1` items

What is the output:

```
>>>string=“Python is the best language, ever!”
>>>newlist=string.split(' ',2);
>>>newlist=string.split(' ');
```

P6

Problem 7 – String Striping

- ‘Strip’ a string to remove all characters in [chars]

```
<name of string>.strip([chars])
```

- Strips the string from front and back by removing all characters in `[chars]`
- Stops strip when a character is encountered that is not in `[chars]`

What is the output:

```
>>>website=“www.pacific.edu”
>>>hostname=website.strip(‘wedu.’)
```

P7

Python Error Handling

- Python handles errors using the `try` and `except` statements

```
try:
    d['c']
except:
    print("Key 'c' is not present")
```

Output:

```
Key 'c' is not present
```

Python Blocks

- Python uses whitespace and ":" to denote blocks
  - **Note: tabs and spaces are not interchangeable!**
- Within a block, all lines are indented exactly the same amount

```
print(l)
    print(l)
```

Output:

```
[1.5, 'a', (3, True)]
IndentationError: unexpected indent
```

Python Statements and Flow Control

- Python supports these statements:
  - `if`
  - `elif`
  - `else`
  - `for`
  - `while`

```
if 1 > 2:
    print(a)
elif 3 > 2:
    print(t)
else:
    print("Neither")
```

Output:

```
(1, 2, 3)
```

Python Statements and Flow Control

- The `for` statement takes a sequence as its input
- This works for any sequence type
  - Tuples, lists, strings, etc…

```
for x in (1,3,5,'a'):
    print(x)
```

Output:

```
1
3
5
a
```

Python Statements and Flow Control

- For the equivalent of a C `for` loop, use the `range` class

```
for i in range(0,9,3):
    print(i)
```

Output:

```
0
3
6
```

This is equivalent to:

```
for (int i=0; i < 9; i += 3)
```

Using Python Libraries

- Libraries (modules) are accessed using the import statement

```
import math
print(math.sin(2))
```

Output:

```
0.9092974268256817
```