

# LECTURE 10: PYTHON TUTORIAL

## Computer Systems and Networks

---

Dr. Pallipuram

([vpallipuramkrishnamani@pacific.edu](mailto:vpallipuramkrishnamani@pacific.edu))

# Today's Class

---

- Python Tutorial 8:15 AM – 9:00 AM
- Work on Labs 6 and 7

# What is Python?

---

Interpreted language for scripting and many other uses

Features: objects, dynamic types, a rich set of libraries and more!

Very notorious for indentation rules



# Install Python3

---

SUDO APT-GET INSTALL PYTHON3



# Python Datatypes

---

Supports ints, floats, booleans, etc.

Other types:

Complex numbers

sequences (tuples and 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 Sequences -- Tuples

---

A *tuple* is an immutable collection of objects

Tuples are denoted by parenthesis

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

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

# Exercise 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])
>>>print(t[-8])
>>>t[2]=t[3]
```



# Exercise 2 – Slicing in tuples

---

What does the following print?

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

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

this method is called slicing. Very useful  
when you only want selected items

# Python Lists

---

A *list* is a mutable collection of objects

Lists are denoted by square brackets

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

# Exercise 3

---

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. Does it work?

```
c. >>> list.insert(4, 'Hello world')
```

```
d. >>> list.pop(-2)
```



# Exercise 4 -- 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:

# Exercise 5 -- String Sequences

---

String sequences are versatile with several associated functions that let you perform neat stuff!

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)
```

# Exercise 6 – Splitting a string 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(' ');
```

# Exercise 7 – Stripping a string

---

`<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.')
```

# 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")
```



# 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
```

# Exercise 8 – What is the output?

---

a.

```
>>> mystring="I"+"\\n"+"love"+"\\n"+"python"
>>> print(mystring)
```

b.

```
>>> newlist=mystring.split('\\n') #split on all
\\n
>>> for word in newlist:
    print(word)
```

# In-Class Participation – Push your work to lab08 folder

---

Pull the boilerplate code and find a folder called Practice. There is a \*.py script waiting for you to be edited. Save it in lab08 folder. The file is executed as:

```
python3 URLanalyzer.py --url  
http://www.google.com/images/srpr/logo3w.png
```

In this example, the host name is [www.google.com](http://www.google.com)

The file name is: /images/srpr/logo3w.png



# In-Class Participation – Push your work to lab08 folder

---

Your goal is to modify this file to automatically extract the hostname and filename and print the message string:

The Hostname is: [www.google.com](http://www.google.com)

The requested File name is: /images/srpr/logo3w.png

COMMIT and PUSH your work!

In lab 08 you will do something similar and send the message string as a stream of bytes!