Computer Systems and Networks

ECPE 170 – University of the Pacific

Crash Dive into Python
Lab Schedule

Activities

- **Today**
  - Python

- **Thursday**
  - **Midterm Exam**

Assignments Due

- Lab 7
  - Due by Mar 20th 5:00am

- Lab 8
  - Due by Mar 27th 5:00am
Person of the Day: Guido van Rossum

- Author of the Python programming language
- Self-appointed “Benevolent Dictator For Life”
- Chose the name because he was “in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)”
- Has worked in numerous organizations, including NIST, Google and Dropbox
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 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)
A tuple is an immutable collection of objects

Tuples are denoted by parenthesis

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

The objects in a tuple do not need to be of the same type
A list is an mutable collection of objects

Lists are denoted by square brackets

```python
l = [1.5, 'a', (3,True)]
```
Tuples and lists are both types of sequences: individual items can be accessed in various ways.

To access a particular item in a sequence:

```python
t = (1,2,3)
l = [1.5, 'a', (3,True)]
print(t[0],l[1])
```

Output:

```
1 a
```
Sequences can also be accessed from the end (instead of beginning) using *negative* indices.

```python
t = (1, 2, 3)
l = [1.5, 'a', (3, True)]
print(t[-2], l[-1])
```

Output:
```
2 (3, True)
```
Slices (subsets of sequences) are accessed by using a “:”

Note that the second index (if supplied) is one greater than actual last object in the slice

```python
t = (1,2,3)
l = [1.5, 'a', (3,True)]
print(t[0:2])
print(l[1:1])
```

Output:

```
(1,2)
['a', (3, True)]
```
A dictionary is an associative array of keys and value pairs.

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

Output:

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

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

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

Output:

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

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

Output:

```
[1.5, 'a', (3, True)]
IndentationError: unexpected indent
```
Python supports these statements:

- if
- elif
- else
- for
- while

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

Output:

```
(1, 2, 3)
```
The `for` statement takes a sequence as its input

This works for any sequence type

- Tuples, lists, strings, etc...

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

Output:

```
1
3
5
a
```
For the equivalent of a C for loop, use the range class

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

Output:

```
0
3
6
```

This is equivalent to:

```python
for (int i=0; i < 9; i += 3)
```
Libraries (modules) are accessed using the import statement

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

Output:
```
0.9092974268256817
```