Python Fundamentals

Dr. Ram Prasad K
VisionCog R&D

ram.krish@visioncog.com
https://www.visioncog.com

Python Fundamentals

Materials adapted from course notes of:
Stanford CS231n: Convolutional Neural Networks for Visual Recognition

Python Fundamentals

Basic data types

Numbers (integers and floats)

```
x = 3

print(type(x))
# Prints "<class 'int'>"

print(x)
# Prints "3"


y = 2.5

print(type(y))
# Prints "<class 'float'>"

print(y, y + 1, y * 2, y ** 2)
# Prints "2.5 3.5 5.0 6.25"
```

no need to define type of data

Python Fundamentals

Basic data types

Booleans

```
t = True
f = False

print(type(t))
# Prints "<class 'bool'>"

print(t and f)
# Logical AND; prints "False"

print(t or f)
# Logical OR; prints "True"

print(not t)
# Logical NOT; prints "False"

print(t != f)
# Logical XOR; prints "True"
```

Python uses words instead of symbols like &&, || for boolean operations

Python Fundamentals

Basic data types

Strings

```
# String literals can use single quotes
str1 = 'hello'

# or double quotes; it does not matter.
str2 = "world"

print(str1)
# "hello"

print(len(str1))
# String length; prints "5"
```

When concatenating, remember to concatenate objects of same type.

e.g., string and int cannot be concatenated

```
# String concatenation
hw = str1 + ' ' + str2

print(hw)
# "hello world"

# sprintf style string formatting
hw12 = '%s %s %d' % (str1, str2, 12)

print(hw12)
# "hello world 12"
```

Python Fundamentals

Basic data types

Strings

```
s = "hello"

print(s.capitalize())
# Capitalize a string; prints "Hello"

print(s.upper())
# Convert a string to uppercase; prints "HELLO"

print(s.replace('l','(ell)'))
# Replace all instances of one substring with another;
# prints "he(ell)(ell)o"
```

string object contains many useful methods

Python Fundamentals

Containers

List

```
# Create a list
list1 = [3, 1, 2]

print(list1, list1[2])
# [3, 1, 2] 2

# Negative indices count from
# the end of the list;
print(list1[-1])
# 2

# Lists can contain elements
# of different types
list1[2] = 'foo'
print(list1)
# [3, 1, 'foo']

# Add a new element to
# the end of the list
list1.append('bar')

print(list1)
# [3, 1, 'foo', 'bar']

# Remove and return the
# last element of the list
list2 = list1.pop()

print(list2, list1)
# bar [3, 1, 'foo']
```

Python Fundamentals

Containers

List slicing

```
                0       1       2       3       4
animals = ['cat', 'dog', 'monkey', 'tiger', 'lion']

print(animals)
# ['cat', 'dog', 'monkey', 'tiger', 'lion']


# Get a slice from index 2 to 4 (exclusive);
print(animals[2:4])
# ['monkey', 'tiger']


# Get a slice from index 2 to the end;
print(animals[2:])
# ['monkey', 'tiger', 'lion']


# Get a slice from the start to index 2 (exclusive);
print(animals[:2])
# ['cat', 'dog']
```

Python Fundamentals

Containers

List slicing

```
# Get a slice of the whole list;
print(animals[:])
# ['cat', 'dog', 'monkey', 'tiger', 'lion']


# Slice indices can be negative;
print(animals[:-1])
# ['cat', 'dog', 'monkey', 'tiger']


# Assign a new sublist to a slice
animals[2:4] = ['deer', 'horse']

print(animals)
# ['cat', 'dog', 'deer', 'horse', 'lion']
```

Python Fundamentals

Containers

List loops

```
animals = ['cat', 'dog', 'monkey']

for elems in animals:
    print(elems)

# cat
# dog
# monkey
```

To access only the elements

Python Fundamentals

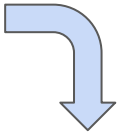
Containers

List loops

```
animals = ['cat', 'dog', 'monkey']

for idx, elem in enumerate(animals):
    print('%d: %s' % (idx, elem))

# 0: cat
# 1: dog
# 2: monkey
```



To access also index within 'for' loop

Python Fundamentals

Containers

List comprehension

```
nums = [0, 1, 2, 3, 4]
squares = []

for x in nums:
    squares.append(x ** 2)

print(squares)
# Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]

squares = [x ** 2 for x in nums]
print(squares)
# Prints [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]

even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)
# Prints "[0, 4, 16]"
```

List comprehensions can also contain conditions

Python Fundamentals

Containers

Dictionary

```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data

print(d['cat'])
# Get an entry from a dictionary; prints "cute"

print('cat' in d)
# Check if a dictionary has a given key; prints "True"

d['fish'] = 'wet'
# Set an entry in a dictionary

print(d['fish'])
# Prints "wet"

# print(d['monkey'])  # KeyError: 'monkey' not a key of d
```

Dictionaries are like associative arrays

Index can be strings, unlike list

Python Fundamentals

Containers

Dictionary loops

```
d = {'person': 2, 'cat': 4, 'spider': 8}

for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))

# A person has 2 legs
# A cat has 4 legs
# A spider has 8 legs
```

iterating only the keys

Python Fundamentals

Containers

Dictionary loops

```
d = {'person': 2, 'cat': 4, 'spider': 8}

for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))

# A person has 2 legs
# A cat has 4 legs
# A spider has 8 legs
```

iterating both key and its value

Python Fundamentals

Containers

Dictionary comprehension

```
nums = [0, 1, 2, 3, 4]

even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}

print(even_num_to_square)
# {0: 0, 2: 4, 4: 16}
```

Python Fundamentals

Tuples

```
# Create a tuple
t = (8, 10, 25)

print(type(t))
# <class 'tuple'>

print(t[0], t[1], t[2])
# 8 10 25

for i in range(len(t)):
    print(t[i])

# 8
# 10
# 25
```

Python Fundamentals

Functions

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
# Prints "negative", "zero", "positive"
```

```
def hello(name, loud=False):
    if loud:
        print('HELLO, %s!' % name.upper())
    else:
        print('Hello, %s' % name)

hello('Bob')
# Prints "Hello, Bob"

hello('Fred', loud=True)
# Prints "HELLO, FRED!"
```

Python Fundamentals

Classes

```
class Greeter(object):

    # Constructor
    def __init__(self, name):
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, loud=False):
        if loud:
            print('HELLO, %s!' % self.name.upper())
        else:
            print('Hello, %s' % self.name)
```

```
g = Greeter('Fred')
# Construct an instance of the Greeter class

g.greet()
# Call an instance method; prints "Hello, Fred"

g.greet(loud=True)
# Call an instance method; prints "HELLO, FRED!"
```

Exercise

Create a list X = [1, 2, 3, 4, 5]

Using list comprehension, create three more lists containing square, cube and quad of X.

Store these lists in a dictionary with keys ‘square’, ‘cube’ and ‘quad’.

Print the dictionary to show keys and its values.