NETWORK PROGRAMMING - SERVER & CLIENT B : FILE TRANSFER

Note

In this chapter, we're going to extend Python Network Programming I - Basic Server / Client A (/python/python_network_programming_server_client.php), and try to file transfer from a server to numerous clients. The main purpose is to check the performance of the server from which clients download files.

Local file transfer

Here is the code to send a file from a local server to a local client.

Python tutorial

```
# server.py

import socket                   # Import socket module

port = 60000                    # Reserve a port for your service.
s = socket.socket()             # Create a socket object
host = socket.gethostname()     # Get local machine name
s.bind((host, port))            # Bind to the port

s.listen(5)                     # Now wait for client connection.

print 'Server listening....'

while True:
    conn, addr = s.accept()     # Establish connection with client.
    print 'Got connection from', addr
    data = conn.recv(1024)
    print('Server received', repr(data))

    filename='mytext.txt'
    f = open(filename,'rb')
    l = f.read(1024)
    while (l):
       conn.send(l)
       print('Sent ',repr(l))
       l = f.read(1024)
    f.close()

    print('Done sending')
    conn.send('Thank you for connecting')
    conn.close()


# client.py

import socket                   # Import socket module

s = socket.socket()             # Create a socket object
host = socket.gethostname()     # Get local machine name
port = 60000                    # Reserve a port for your service.

s.connect((host, port))
s.send("Hello server!")

with open('received_file', 'wb') as f:
    print 'file opened'
    while True:
        print('receiving data...')
        data = s.recv(1024)
        print('data=%s', (data))
        if not data:
            break
        # write data to a file
        f.write(data)

f.close()
print('Successfully get the file')
s.close()
print('connection closed')
```

Output on a local server:

```
Server listening....
Got connection from ('192.168.56.10', 62854)
('Server received', "'Hello server!'")
('Sent ', "'1 1234567890\\n
...
('Sent ', "'4567890\\n105
...
('Sent ', "'300 1234567890\\n'")
Done sending
```

Output on a local client:

```
file opened
receiving data...
data=1 1234567890
2 1234567890
...
103 1234567890
104 123
receiving data...
data=4567890
105 1234567890
106 1234567890
...
299 1234567890

receiving data...
data=300 1234567890
Thank you for connecting
receiving data...
data=
Successfully get the file
connection closed
```

multithread tcp file transfer on localhost

Our server code above can only interact with one client. If we try to connect with a second client, however, it simply won't reply to the new client. To let the server interact with multiple clients, we need to use multi-threading. Here is the new server script to accept multiple client connections:

plagiarism (/python/python_sets_union_inte

Hashing (Hash tables and hashlib) (/python/python_hash_tables_ha

Dictionary Comprehension with zip (/python/python_dictionary_com

The yield keyword (/python/python_function_with_y

Generator Functions and Expressions (/python/python_generators.php

generator.send() method (/python/python_function_with_g

Iterators (/python/python_iterators.php)

Classes and Instances (__init__, __call__, etc.) (/python/python_classes_instanc

if__name__ == '__main__' (/python/python_if__name__equa

argparse (/python/python_argparse.php)

Exceptions (/python/python_try_except_final

@static method vs class method (/python/python_differences_bet

Private attributes and private methods (/python/python_private_attribut

bits, bytes, bitstring, and constBitStream (/python/python_bits_bytes_bitst

json.dump(s) and json.load(s) (/python/python-json-dumps-loads-file-read-write.php)

Python Object Serialization -

```
# server2.py
import socket
from threading import Thread
from SocketServer import ThreadingMixIn

TCP_IP = 'localhost'
TCP_PORT = 9001
BUFFER_SIZE = 1024


class ClientThread(Thread):

    def __init__(self,ip,port,sock):
        Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.sock = sock
        print " New thread started for "+ip+":"+str(port)

    def run(self):
        filename='mytext.txt'
        f = open(filename,'rb')
        while True:
            l = f.read(BUFFER_SIZE)
            while (l):
                self.sock.send(l)
                #print('Sent ',repr(l))
                l = f.read(BUFFER_SIZE)
            if not l:
                f.close()
                self.sock.close()
                break

tcpsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpsock.bind((TCP_IP, TCP_PORT))
threads = []

while True:
    tcpsock.listen(5)
    print "Waiting for incoming connections..."
    (conn, (ip,port)) = tcpsock.accept()
    print 'Got connection from ', (ip,port)
    newthread = ClientThread(ip,port,conn)
    newthread.start()
    threads.append(newthread)

for t in threads:
    t.join()
```

```
# client2.py
#!/usr/bin/env python

import socket

TCP_IP = 'localhost'
TCP_PORT = 9001
BUFFER_SIZE = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
with open('received_file', 'wb') as f:
```

pickle and json (/python/python_serialization_pic

Python Object Serialization - yaml and json (/python/python_yaml_json_conv

Priority queue and heap queue data structure (/python/python_PriorityQueue_h

Graph data structure (/python/python_graph_data_stru

Dijkstra's shortest path algorithm (/python/python_Dijkstras_Short

Prim's spanning tree algorithm (/python/python_Prims_Spanning

Closure (/python/python_closure.php)

Functional programming in Python (/python/python_functional_prog

Remote running a local file using ssh (/python/python_ssh_remote_run

SQLite 3 - A. Connecting to DB, create/drop table, and insert data into a table (/python/python_sqlite_connect_

SQLite 3 - B. Selecting, updating and deleting data (/python/python_sqlite_select_up

MongoDB with PyMongo I - Installing MongoDB ... (/python/MongoDB_PyMongo/py

Python HTTP Web Services - urllib, httplib2 (/python/python_http_web_servic

Web scraping with Selenium for checking domain availability (/python/python_Web_scraping_w

REST API : Http Requests for

```
    print 'file opened'
    while True:
        #print('receiving data...')
        data = s.recv(BUFFER_SIZE)
        print('data=%s', (data))
        if not data:
            f.close()
            print 'file close()'
            break

        # write data to a file
        f.write(data)

print('Successfully get the file')
s.close()
print('connection closed')
```

Below is the output from the server console when we run two clients simultaneously:

```
$ python server2.py
Waiting for incoming connections...
Got connection from  ('127.0.0.1', 55184)
 New thread started for 127.0.0.1:55184
Waiting for incoming connections...
Got connection from  ('127.0.0.1', 55185)
 New thread started for 127.0.0.1:55185
Waiting for incoming connections...
```

tcp file download from EC2 to local

In the following codes, we made two changes:

1. ip switched to amazon ec2 ip
2. To calculate the time to take download a file, we import **time** module.

Humans with Flask (/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php)

Blog app with Tornado (/python/Tornado/Python_Torna

Multithreading ... (/python/Multithread/python_mu

Python Network Programming I - Basic Server / Client : A Basics (/python/python_network_progra

Python Network Programming I - Basic Server / Client : B File Transfer (/python/python_network_progra

Python Network Programming II - Chat Server / Client (/python/python_network_progra

Python Network Programming III - Echo Server using socketserver network framework (/python/python_network_progra

Python Network Programming IV - Asynchronous Request Handling : ThreadingMixIn and ForkingMixIn (/python/python_network_progra

Python Interview Questions I (/python/python_interview_quest

Python Interview Questions II (/python/python_interview_quest

Python Interview Questions III (/python/python_interview_quest

Python Interview Questions IV (/python/python_interview_quest

Python Interview Questions V (/python/python_interview_quest

Image processing with Python image library Pillow (/python/python_image_processi

```
# server3.py on EC2 instance
import socket
from threading import Thread
from SocketServer import ThreadingMixIn

# TCP_IP = 'localhost'
TCP_IP = socket.gethostbyaddr("your-ec2-public_ip")[0]
TCP_PORT = 60001
BUFFER_SIZE = 1024

print 'TCP_IP=',TCP_IP
print 'TCP_PORT=',TCP_PORT

class ClientThread(Thread):

    def __init__(self,ip,port,sock):
        Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.sock = sock
        print " New thread started for "+ip+":"+str(port)

    def run(self):
        filename='mytext.txt'
        f = open(filename,'rb')
        while True:
            l = f.read(BUFFER_SIZE)
            while (l):
                self.sock.send(l)
                #print('Sent ',repr(l))
                l = f.read(BUFFER_SIZE)
            if not l:
                f.close()
                self.sock.close()
                break

tcpsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpsock.bind((TCP_IP, TCP_PORT))
threads = []

while True:
    tcpsock.listen(5)
    print "Waiting for incoming connections..."
    (conn, (ip,port)) = tcpsock.accept()
    print 'Got connection from ', (ip,port)
    newthread = ClientThread(ip,port,conn)
    newthread.start()
    threads.append(newthread)

for t in threads:
    t.join()


# client3.py on local machine
#!/usr/bin/env python

#!/usr/bin/env python

import socket
import time

#TCP_IP = 'localhost'
```

Python and C++ with SIP (/python/python_cpp_sip.php)

PyDev with Eclipse (/python/pydev_eclipse_plugin_ir

Matplotlib (/python/python_matplotlib.php)

Redis with Python (/python/python_redis_with_pyth

NumPy array basics A (/python/python_numpy_array_tu

NumPy Matrix and Linear Algebra (/python/python_numpy_matrix_

Pandas with NumPy and Matplotlib (/python/python_Pandas_NumPy

Celluar Automata (/python/python_cellular_automa

Batch gradient descent algorithm (/python/python_numpy_batch_g

Longest Common Substring Algorithm (/python/python_longest_commc

Python Unit Test - TDD using unittest.TestCase class (/python/python_unit_testing.php

Simple tool - Google page ranking by keywords (/python/python_site_page_ranki

Google App Hello World (/python/GoogleApp/python_Goo

Google App webapp2 and WSGI (/python/GoogleApp/python_Goo

Uploading Google App Hello World (/python/GoogleApp/python_Goo

Python 2 vs Python 3

```
TCP_IP = 'ip-ec2-instance'
TCP_PORT = 60001

BUFFER_SIZE = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))

clock_start = time.clock()

time_start = time.time()

with open('received_file', 'wb') as f:
    print 'file opened'
    while True:
        #print('receiving data...')
        data = s.recv(1024)
        #print('data=%s', (data))
        if not data:
            f.close()
            print 'file close()'
            break
        # write data to a file
        f.write(data)

print('Successfully get the file')
s.close()
print('connection closed')

clock_end = time.clock()
time_end = time.time()

duration_clock = clock_end - clock_start
print 'clock:  start = ',clock_start, ' end = ',clock_end
print 'clock:  duration_clock = ', duration_clock

duration_time = time_end - time_start
print 'time:  start = ',time_start, ' end = ',time_end
print 'time:  duration_time = ', duration_time
```

Server console shows the following output after a connection from my local home machine:

```
$ python server3.py
TCP_IP= ec2-...
TCP_PORT= 60001
Waiting for incoming connections...
Got connection from  ('108.239.135.40', 56742)
 New thread started for 108.239.135.40:56742
```

The ip is isp's:

Geolocation data from IP2Location (Product: DB4 updated on 6/1/2015)

IP Address	Country	Region	City	ISP
108.239.135.40	United States	California	Fair Oaks	At&t Internet Services

On my local mac:

(/python/python_differences_Pyt

virtualenv and virtualenvwrapper (/python/python_virtualenv_virtu

Uploading a big file to AWS S3 using boto module (/DevOps/AWS/aws_S3_uploadin

Scheduled stopping and starting an AWS instance (/DevOps/AWS/aws_stopping_sta

Cloudera CDH5 - Scheduled stopping and starting services (/Hadoop/BigData_hadoop_CDH5

Removing Cloud Files - Rackspace API with curl and subprocess (/python/python_Rackspace_API_

Checking if a process is running/hanging and stop/run a scheduled task on Windows (/python/python-Windows-Check-if-a-Process-is-Running-Hanging-Schtasks-Run-Stop.php)

Apache Spark 1.3 with PySpark (Spark Python API) Shell (/Hadoop/BigData_hadoop_Apac

Apache Spark 1.2 Streaming (/Hadoop/BigData_hadoop_Apac

bottle 0.12.7 - Fast and simple WSGI-micro framework for small web-applications ... (/python/Bottle/Python_Bottle_Fr

Flask app with Apache WSGI on Ubuntu14/CentOS7 ... (/python/Flask/Python_Flask_Blog

Selenium WebDriver (/python/python_Selenium_WebD

Fabric - streamlining the use of SSH for application deployment (/python/Fabric/python_Fabric.ph

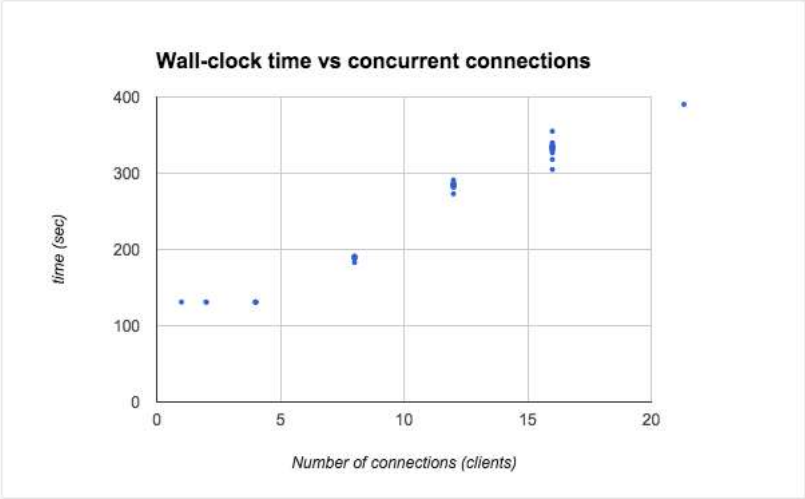
```
$ python client3.py
file opened
file close()
Successfully get the file
connection closed
clock:  start =  0.018806  end =  0.038608
clock:  duration_clock =  0.019802
time:  start =  1434991840.37  end =  1434991840.42
time:  duration_time =  0.0457620620728
```

File downloaded from EC2, **received_file** is simple, and it looks like this:

```
From EC2
1
2
3
4
5
6
7
8
9
```

Download time vs number of clients

Here is the output showing the wall-clock time depending on the number of concurrent connections:



Our server is located in California, and the following picture compares the download speed between US and Japan:

Ansible Quick Preview - Setting up web servers with Nginx, configure enviroments, and deploy an App (/DevOps/Ansible/Ansible_Setting

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networ

NLP - NLTK (Natural Language Toolkit) ... (/python/NLTK/NLTK_install.php)

RabbitMQ(Message broker server) and Celery(Task queue) ... (/python/RabbitMQ_Celery/pythc

OpenCV3 and Matplotlib ... (/python/OpenCV_Python/pythor

Simple tool - Concatenating slides using FFmpeg ... (/FFMpeg/ffmpeg_fade_in_fade_c

iPython - Signal Processing with NumPy (/python/OpenCV_Python/pythor

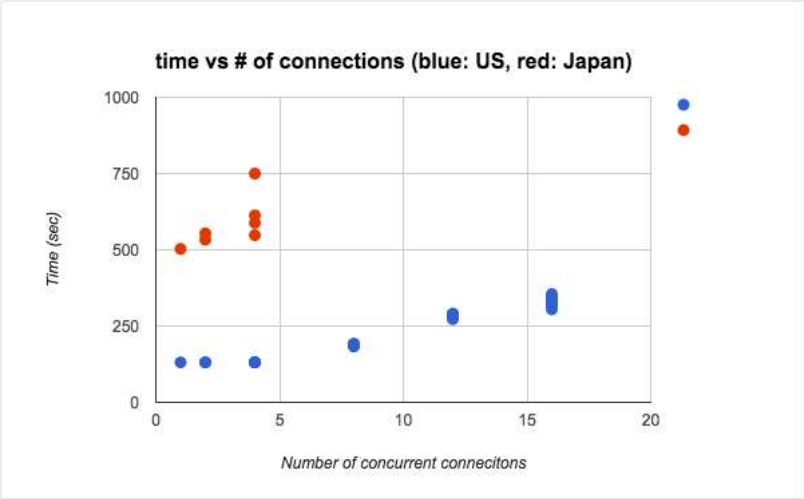
iPython and Jupyter - Install Jupyter, iPython Notebook, drawing with Matplotlib, and publishing it to Github (/python/IPython/IPython_Jupyte

iPython and Jupyter Notebook with Embedded D3.js (/python/IPython/iPython_Jupyte

Downloading YouTube videos using youtube-dl embedded with Python (/VideoStreaming/YouTube/youtu dl-embedding.php)

Machine Learning : scikit-learn ... (/python/scikit-learn/scikit_machine_learning_Su

Django 1.6/1.8 Web Framework ... (/python/Django/Python_Django_



Python Network Programming

Network Programming - Server & Client A : Basics
(/python/python_network_programming_server_client.php)

Network Programming - Server & Client B : File Transfer
(/python/python_network_programming_server_client_file_transfer.php)

Network Programming II - Chat Server & Client
(/python/python_network_programming_tcp_server_client_chat_server_chat_client_select.php)

Network Programming III - SocketServer
(/python/python_network_programming_socketserver_framework_for_network_servers.php)

Network Programming IV - SocketServer Asynchronous request
(/python/python_network_programming_socketserver_framework_for_network_servers_asynchronous_request_ThreadingMixIn_ForkingMi

Python tutorial

Python Home (/python/pytut.php)

OpenCV 3 image and video processing with Python

OpenCV 3 with Python
(/python/OpenCV_Python/pythor

Iterators (/python/python_iterators.php)

Classes and Instances (__init__, __call__, etc.) (/python/python_classes_instances.php)

if__name__ == '__main__' (/python/python_if__name__equals__main__.php)

argparse (/python/python_argparse.php)

Exceptions (/python/python_try_except_finally_raise_syntax_error.php)

@static method vs class method (/python/python_differences_between_static_method_and_class_method_instance_method.php)

Private attributes and private methods (/python/python_private_attributes_methods.php)

bits, bytes, bitstring, and constBitStream (/python/python_bits_bytes_bitstring_constBitStream.php)

json.dump(s) and json.load(s) (/python/python-json-dumps-loads-file-read-write.php)

Python Object Serialization - pickle and json (/python/python_serialization_pickle_json.php)

Python Object Serialization - yaml and json (/python/python_yaml_json_conversion.php)

Priority queue and heap queue data structure (/python/python_PriorityQueue_heapq_Data_Structure.php)

Graph data structure (/python/python_graph_data_structures.php)

Dijkstra's shortest path algorithm (/python/python_Dijkstras_Shortest_Path_Algorithm.php)

Prim's spanning tree algorithm (/python/python_Prims_Spanning_Tree_Data_Structure.php)

Closure (/python/python_closure.php)

Functional programming in Python (/python/python_functional_programming.php)

Remote running a local file using ssh (/python/python_ssh_remote_run.php)

SQLite 3 - A. Connecting to DB, create/drop table, and insert data into a table (/python/python_sqlite_connect_create_drop_table.php)

SQLite 3 - B. Selecting, updating and deleting data (/python/python_sqlite_select_update_delete.php)

MongoDB with PyMongo I - Installing MongoDB ... (/python/MongoDB_PyMongo/python_MongoDB_pyMongo_tutorial_installing.php)

Python HTTP Web Services - urllib, httplib2 (/python/python_http_web_services.php)

Web scraping with Selenium for checking domain availability (/python/python_Web_scraping_with_selenium_for_domain_availability.php)

REST API : Http Requests for Humans with Flask (/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php)

Watershed Algorithm : Marker-based Segmentation II (/python/OpenCV_Python/pythor

Image noise reduction : Non-local Means denoising algorithm (/python/OpenCV_Python/pythor local_Means_Denoising_Algorithr

Image object detection : Face detection using Haar Cascade Classifiers (/python/OpenCV_Python/pythor

Image segmentation - Foreground extraction Grabcut algorithm based on graph cuts (/python/OpenCV_Python/pythor

Image Reconstruction - Inpainting (Interpolation) - Fast Marching Methods (/python/OpenCV_Python/pythor

Video : Mean shift object tracking (/python/OpenCV_Python/pythor

Machine Learning : Clustering - K-Means clustering I (/python/OpenCV_Python/pythor Means_Clustering_Vector_Quanti

Machine Learning : Clustering - K-Means clustering II (/python/OpenCV_Python/pythor Means_Clustering_Vector_Quanti

Machine Learning : Classification - k-nearest neighbors (k-NN) algorithm (/python/OpenCV_Python/pythor nearest_neighbors_k-NN.php)

Machine Learning with

Blog app with Tornado (/python/Tornado/Python_Tornado_Blog_App.php)

Multithreading ... (/python/Multithread/python_multithreading_creating_threads.php)

Python Network Programming I - Basic Server / Client : A Basics (/python/python_network_programming_server_client.php)

Python Network Programming I - Basic Server / Client : B File Transfer (/python/python_network_programming_server_client_file_transfer.php)

Python Network Programming II - Chat Server / Client (/python/python_network_programming_tcp_server_client_chat_server_chat_client_select.php)

Python Network Programming III - Echo Server using socketserver network framework (/python/python_network_programming_socketserver_framework_for_network_servers.php)

Python Network Programming IV - Asynchronous Request Handling : ThreadingMixIn and ForkingMixIn (/python/python_network_programming_socketserver_framework_for_network_servers_asynchronous_request_ThreadingMixIn_ForkingMix

Python Interview Questions I (/python/python_interview_questions.php)

Python Interview Questions II (/python/python_interview_questions_2.php)

Python Interview Questions III (/python/python_interview_questions_3.php)

Python Interview Questions IV (/python/python_interview_questions_4.php)

Python Interview Questions V (/python/python_interview_questions_5.php)

Image processing with Python image library Pillow (/python/python_image_processing_with_Pillow_library.php)

Python and C++ with SIP (/python/python_cpp_sip.php)

PyDev with Eclipse (/python/pydev_eclipse_plugin_install_python_IDE.php)

Matplotlib (/python/python_matplotlib.php)

Redis with Python (/python/python_redis_with_python.php)

NumPy array basics A (/python/python_numpy_array_tutorial_basic_A.php)

NumPy Matrix and Linear Algebra (/python/python_numpy_matrix_tutorial.php)

Pandas with NumPy and Matplotlib (/python/python_Pandas_NumPy_Matplotlib.php)

Celluar Automata (/python/python_cellular_automata.php)

Batch gradient descent algorithm (/python/python_numpy_batch_gradient_descent_algorithm.php)

Longest Common Substring Algorithm (/python/python_longest_common_substring_lcs_algorithm_generalized_suffix_tree.php)

Python Unit Test - TDD using unittest.TestCase class (/python/python_unit_testing.php)

scikit-learn

scikit-learn installation (/python/scikit-learn/scikit-learn_install.php)

scikit-learn : Features and feature extraction - iris dataset (/python/scikit-learn/scikit_machine_learning_fe

scikit-learn : Machine Learning Quick Preview (/python/scikit-learn/scikit_machine_learning_qu

scikit-learn : Data Preprocessing I - Missing / Categorical data (/python/scikit-learn/scikit_machine_learning_Da Missing-Data-Categorical-Data.php)

scikit-learn : Data Preprocessing II - Partitioning a dataset / Feature scaling / Feature Selection / Regularization (/python/scikit-learn/scikit_machine_learning_Da II-Datasets-Partitioning-Feature-scaling-Feature-Selection-Regularization.php)

scikit-learn : Data Preprocessing III - Dimensionality reduction vis Sequential feature selection / Assessing feature importance via random forests (/python/scikit-learn/scikit_machine_learning_Da III-Dimensionality-reduction-via-Sequential-feature-selection-Assessing-feature-importance-via-random-forests.php)

Data Compression via Dimensionality Reduction I - Principal component analysis (PCA) (/python/scikit-learn/scikit_machine_learning_Da _PCA.php)

Simple tool - Google page ranking by keywords (/python/python_site_page_ranking_by_keywords.php)

Google App Hello World (/python/GoogleApp/python_GoogleApp_HelloWorld.php)

Google App webapp2 and WSGI (/python/GoogleApp/python_GoogleApp_WebApp2_WSGI.php)

Uploading Google App Hello World (/python/GoogleApp/python_GoogleApp_Uploading_HelloWorld.php)

Python 2 vs Python 3 (/python/python_differences_Python2_vs_Python3_port.php)

virtualenv and virtualenvwrapper (/python/python_virtualenv_virtualenvwrapper.php)

Uploading a big file to AWS S3 using boto module (/DevOps/AWS/aws_S3_uploading_large_file.php)

Scheduled stopping and starting an AWS instance (/DevOps/AWS/aws_stopping_starting_instances.php)

Cloudera CDH5 - Scheduled stopping and starting services (/Hadoop/BigData_hadoop_CDH5_stop_start_services.php)

Removing Cloud Files - Rackspace API with curl and subprocess (/python/python_Rackspace_API_curl_subprocess_Cloud_Files.php)

Checking if a process is running/hanging and stop/run a scheduled task on Windows (/python/python-Windows-Check-if-a-Process-is-Running-Hanging-Schtasks-Run-Stop.php)

Apache Spark 1.3 with PySpark (Spark Python API) Shell (/Hadoop/BigData_hadoop_Apache_Spark_PySpark.php)

Apache Spark 1.2 Streaming (/Hadoop/BigData_hadoop_Apache_Spark_Streaming.php)

bottle 0.12.7 - Fast and simple WSGI-micro framework for small web-applications ... (/python/Bottle/Python_Bottle_Framework.php)

Flask app with Apache WSGI on Ubuntu14/CentOS7 ... (/python/Flask/Python_Flask_Blog_App_Production_with_MongoDB_and_Apache_WSGI.php)

Fabric - streamlining the use of SSH for application deployment (/python/Fabric/python_Fabric.php)

Ansible Quick Preview - Setting up web servers with Nginx, configure enviroments, and deploy an App (/DevOps/Ansible/Ansible_SettingUp_Webservers_Nginx_Install_Env_Configure_Deploy_App.php)

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networks_Backpropagation_for_XOR_using_one_hidden_layer.php)

NLP - NLTK (Natural Language Toolkit) ... (/python/NLTK/NLTK_install.php)

RabbitMQ(Message broker server) and Celery(Task queue) ... (/python/RabbitMQ_Celery/python_Installing_RabbitMQ_Celery.php)

OpenCV3 and Matplotlib ... (/python/OpenCV_Python/python_opencv3_matplotlib_rgb_brg_image_load_display_save.php)

scikit-learn : Data Compression via Dimensionality Reduction II - Linear Discriminant Analysis (LDA) (/python/scikit-learn/scikit_machine_learning_Da

scikit-learn : Data Compression via Dimensionality Reduction III - Nonlinear mappings via kernel principal component (KPCA) analysis (/python/scikit-learn/scikit_machine_learning_Da nonlinear-mappings-via-kernel-principal-component-analysis.php)

scikit-learn : Logistic Regression, Overfitting & regularization (/python/scikit-learn/scikit-learn_logistic_regression.php)

scikit-learn : Supervised Learning & Unsupervised Learning - e.g. Unsupervised PCA dimensionality reduction with iris dataset (/python/scikit-learn/scikit_machine_learning_Su

scikit-learn : Unsupervised_Learning - KMeans clustering with iris dataset (/python/scikit-learn/scikit_machine_learning_Ur

scikit-learn : Linearly Separable Data - Linear Model & (Gaussian) radial basis function kernel (RBF kernel) (/python/scikit-learn/scikit_machine_learning_Li

scikit-learn : Decision Tree Learning I - Entropy, Gini, and Information Gain (/python/scikit-learn/scikt_machine_learning_De

scikit-learn : Decision Tree Learning II - Constructing the Decision Tree (/python/scikit-learn/scikit_machine_learning_Cc

Simple tool - Concatenating slides using FFmpeg ... (/FFMpeg/ffmpeg_fade_in_fade_out_transitions_effects_filters_slideshow_concat.php)

iPython - Signal Processing with NumPy (/python/OpenCV_Python/python_opencv3_NumPy_Arrays_Signal_Processing_iPython.php)

iPython and Jupyter - Install Jupyter, iPython Notebook, drawing with Matplotlib, and publishing it to Github (/python/IPython/IPython_Jupyter_Install_iPython_Notebook_Matplotlib_Publishing_it_to_Github.php)

iPython and Jupyter Notebook with Embedded D3.js (/python/IPython/iPython_Jupyter_Notebook_with_Embedded_D3.php)

Downloading YouTube videos using youtube-dl embedded with Python (/VideoStreaming/YouTube/youtube-dl-embedding.php)

Machine Learning : scikit-learn ... (/python/scikit-learn/scikit_machine_learning_Supervised_Learning_Unsupervised_Learning.php)

Django 1.6/1.8 Web Framework ... (/python/Django/Python_Django_tutorial_introduction.php)

scikit-learn : Random Decision Forests Classification (/python/scikit-learn/scikit_machine_learning_Ra

scikit-learn : Support Vector Machines (SVM) (/python/scikit-learn/scikit_machine_learning_Su

scikit-learn : Support Vector Machines (SVM) II (/python/scikit-learn/scikit_machine_learning_Su

Flask with Embedded Machine Learning I : Serializing with pickle and DB setup (/python/Flask/Python_Flask_Emb

Flask with Embedded Machine Learning II : Basic Flask App (/python/Flask/Python_Flask_Emb

Flask with Embedded Machine Learning III : Embedding Classifier (/python/Flask/Python_Flask_Emb

Flask with Embedded Machine Learning IV : Deploy (/python/Flask/Python_Flask_Emb

Flask with Embedded Machine Learning V : Updating the classifier (/python/Flask/Python_Flask_Emb

scikit-learn : Sample of a spam comment filter using SVM - classifying a good one or a bad one (/python/scikit-learn/scikit_learn_Support_Vecto

MACHINE LEARNING ALGORITHMS AND CONCEPTS

Batch gradient descent algorithm

(/python/python_numpy_batch_g

Single Layer Neural Network - Perceptron model on the Iris dataset using Heaviside step activation function (/python/scikit-learn/Perceptron_Model_with_Iri

Batch gradient descent versus stochastic gradient descent (/python/scikit-learn/scikit-learn_batch-gradient-descent-versus-stochastic-gradient-descent.php)

Single Layer Neural Network - Adaptive Linear Neuron using linear (identity) activation function with batch gradient descent method (/python/scikit-learn/Single-Layer-Neural-Network-Adaptive-Linear-Neuron.php)

Single Layer Neural Network : Adaptive Linear Neuron using linear (identity) activation function with stochastic gradient descent (SGD) (/python/scikit-learn/Single-Layer-Neural-Network-Adaptive-Linear-Neuron-with-Stochastic-Gradient-Descent.php)

Logistic Regression (/python/scikit-learn/logistic_regression.php)

VC (Vapnik-Chervonenkis) Dimension and Shatter (/python/scikit-learn/scikit_machine_learning_VC

Bias-variance tradeoff (/python/scikit-learn/scikit_machine_learning_Bi variance-Tradeoff.php)

Maximum Likelihood Estimation (MLE) (/python/scikit-learn/Maximum-Likelyhood-Estimation-

MLE.php)

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networ

minHash (/Algorithms/minHash_Jaccard_Si

tf-idf weight (/Algorithms/tf_idf_term_frequen

Natural Language Processing (NLP): Sentiment Analysis I (IMDb & bag-of-words) (/Algorithms/Machine_Learning_I

Natural Language Processing (NLP): Sentiment Analysis II (tokenization, stemming, and stop words) (/Algorithms/Machine_Learning_I

Natural Language Processing (NLP): Sentiment Analysis III (training & cross validation) (/Algorithms/Machine_Learning_I

Natural Language Processing (NLP): Sentiment Analysis IV (out-of-core) (/Algorithms/Machine_Learning_I

Locality-Sensitive Hashing (LSH) using Cosine Distance (Cosine Similarity) (/Algorithms/Locality_Sensitive_H

ARTIFICIAL NEURAL NETWORKS (ANN)

[Note] Sources are available at Github - Jupyter notebook files (https://github.com/Einsteinish/A Neural-Networks-with-Jupyter.git)

1. Introduction (/python/scikit-learn/Artificial-Neural-Network-

ANN-1-Introduction.php)

2. Forward Propagation (/python/scikit-learn/Artificial-Neural-Network-ANN-2-Forward-Propagation.php)

3. Gradient Descent (/python/scikit-learn/Artificial-Neural-Network-ANN-3-Gradient-Descent.php)

4. Backpropagation of Errors (/python/scikit-learn/Artificial-Neural-Network-ANN-4-Backpropagation.php)

5. Checking gradient (/python/scikit-learn/Artificial-Neural-Network-ANN-5-Checking-Gradient.php)

6. Training via BFGS (/python/scikit-learn/Artificial-Neural-Network-ANN-6-Training-via-BFGS-Broyden-Fletcher-Goldfarb-Shanno-algorithm-a-variant-of-gradient-descent.php)

7. Overfitting & Regularization (/python/scikit-learn/Artificial-Neural-Network-ANN-7-Overfitting-Regularization.php)

8. Deep Learning I : Image Recognition (Image uploading) (/python/scikit-learn/Artificial-Neural-Network-ANN-8-Deep-Learning-1-Image-Recognition-Image-Uploading.php)

9. Deep Learning II : Image Recognition (Image classification) (/python/scikit-learn/Artificial-Neural-Network-ANN-9-Deep-Learning-2-Image-Recognition-Image-Classification.php)

10 - Deep Learning III : Deep Learning III : Theano, TensorFlow, and Keras (/python/scikit-learn/Artificial-

Neural-Network-ANN-10-Deep-Learning-3-Theano-TensorFlow-Keras.php)

CONTACT

BogoToBogo
contactus@bogotobogo.com (mailto:#)

FOLLOW BOGOTOBOGO

(https://www.facebook.com/KHongSanFrancisco) (https://twitter.com/KHongTwit) (https://plus.google.com/u/0/+KHongSanFrancisco/posts)

ABOUT US (/ABOUT_US.PHP)

contactus@bogotobogo.com (mailto:#)

Golden Gate Ave, San Francisco, CA 94115

Golden Gate Ave, San Francisco, CA 94115