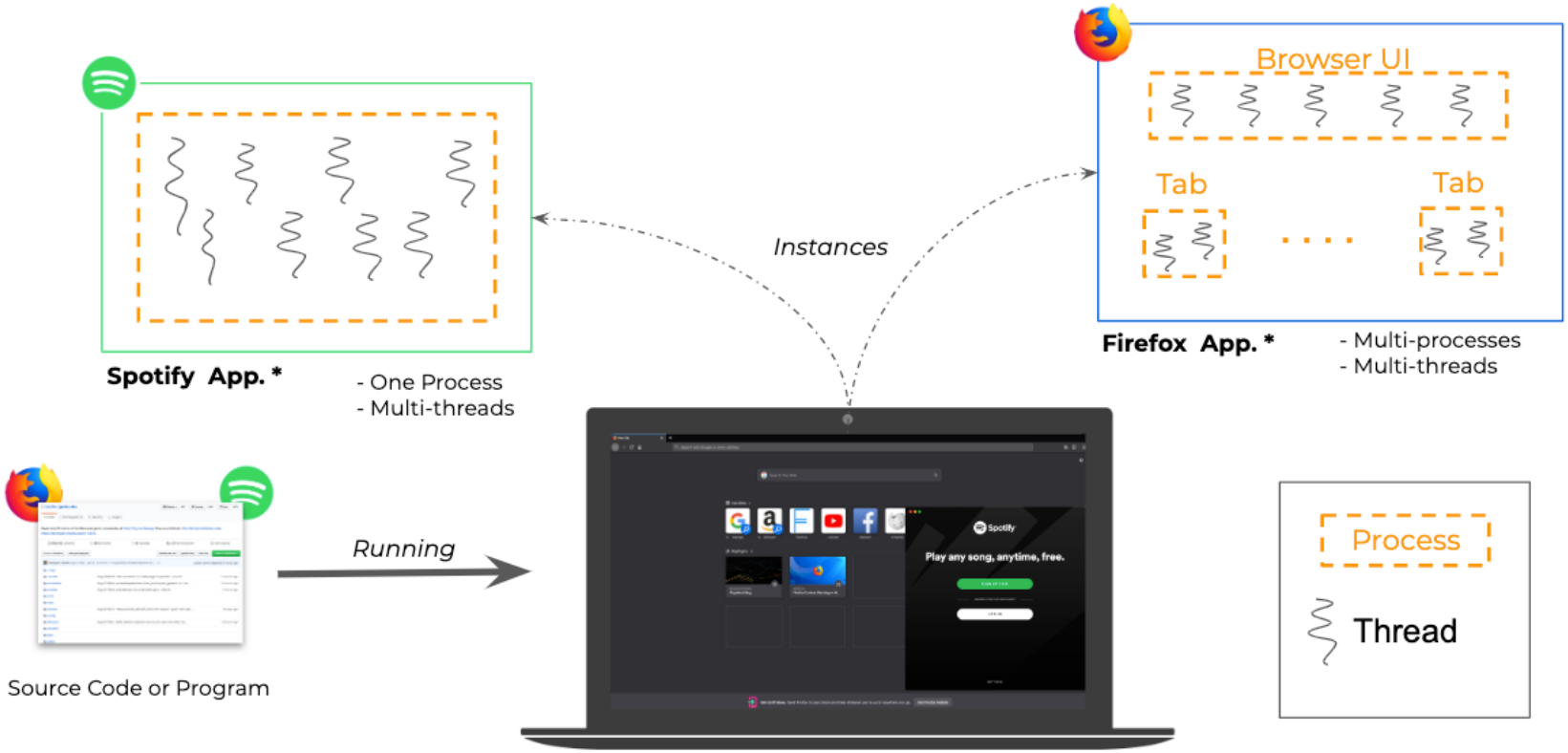
Đa luồng, đa tiến trình

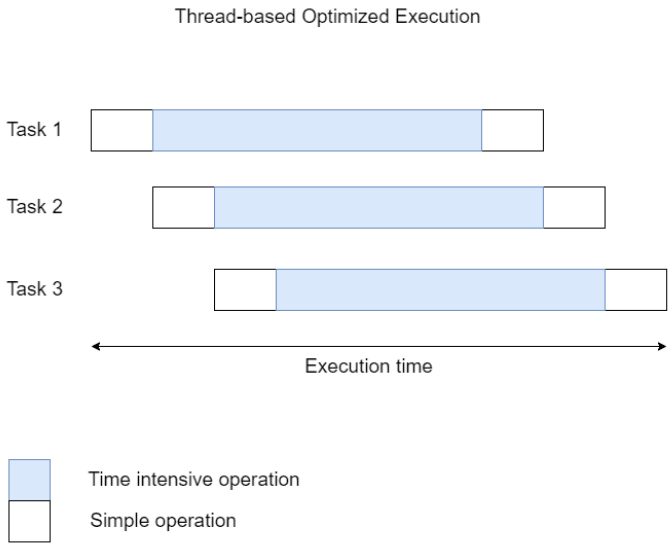
Process, Thread

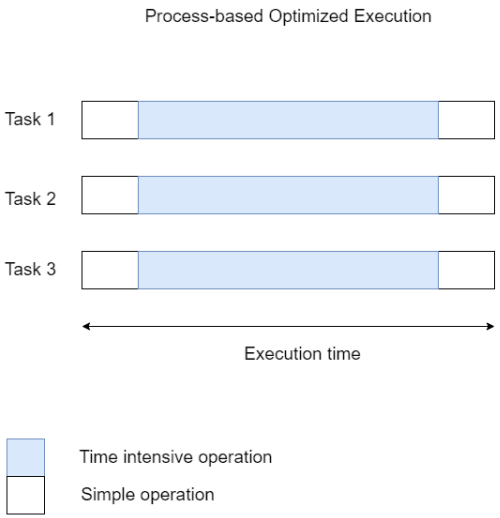
Programs, Apps, Processes & Threads



** this image may not reflect the reality for the show-cased apps*

Process, Thread





Differences Between threading.Thread and multiprocessing.Process

Thread	Process
• Uses Native Threads	• Uses native Processes
• Belongs to a Process	• Has Threads and Child Processes
• Lightweight, fast to start	• Heavyweight, slow to start
• Shared Memory	• Inter-Process Communication
• Subject to the GIL	• Not Subject to the GIL
• Suited to IO-bound Tasks	• Suited to CPU-bound Tasks
• Create 10s to 1000s Workers	• Create 10s of Workers

Thread

- Chạy một hàm trong **Thread** mới

```
import threading
# a target function that does something
def work()
    # do something...

# create a thread to execute the work() function
thread = threading.Thread(target=work)
# start the thread
thread.start()
```

```
# create a thread to execute the work()
function
thread = Thread(target=work, args=(123,))
```

- Xây dựng lớp kế thừa Thread class và override phương thức run().

Thread

- Xây dựng lớp kế thừa Thread class và override phương thức run().

```
import threading
# define a custom thread
class CustomThread(threading.Thread):
    # custom run function
    def run():
        # do something...
# create the custom thread
thread = CustomThread()
# start the thread
thread.start()
```

Process

- Chạy một hàm trong **Process** mới

```
import multiprocessing
# a target function that does something
def work()
    # do something...

# protect entry point
if __name__ == '__main__':
    # create a process to execute the work()
function
    process = multiprocessing .Process(target=work)
    # start the process
    process.start()
```

```
...
# create a process to execute the work() function
process = multiprocessing.Process(target=work, args=(123,))
```

- Xây dựng lớp kế thừa **Process** class và override phương thức run()

Process

- Xây dựng lớp kế thừa **Process** class và override phương thức run().

```
import multiprocessing
# define a custom process
class CustomProcess(multiprocessing.Process):
    # custom run function
    def run():
        # do something...

# protect entry point
if __name__ == '__main__':
    # create the custom process
    process = CustomProcess()
    # start the process
    process.start()
```

```python
import multiprocessing #importing the module
import time
def even(n): #function to print all even numbers till n
    for i in range(0,n,2):
        print("even:",i)
        time.sleep(1)
def odd(n): #function to print all odd numbers till n
    for i in range(1,n,2):
        print("odd:",i)
        time.sleep(1)
if __name__=="__main__":
    # creating processes for each of the functions
    prc1 = multiprocessing.Process(target=even, args=(15, ))
    prc2 = multiprocessing.Process(target=odd, args=(15, ))
    # starting the 1st process
    prc1.start()
    # starting the 2nd process
    prc2.start()
```

Kết quả

```
even: 0
odd: 1
even: 2
odd: 3
even: 4
odd: 5
even: 6
odd: 7
even: 8
odd: 9
even: 10
odd: 11
even: 12
odd: 13
even: 14
```

Introduction to Parallel and Concurrent Programming in Python (tutsplus.com)

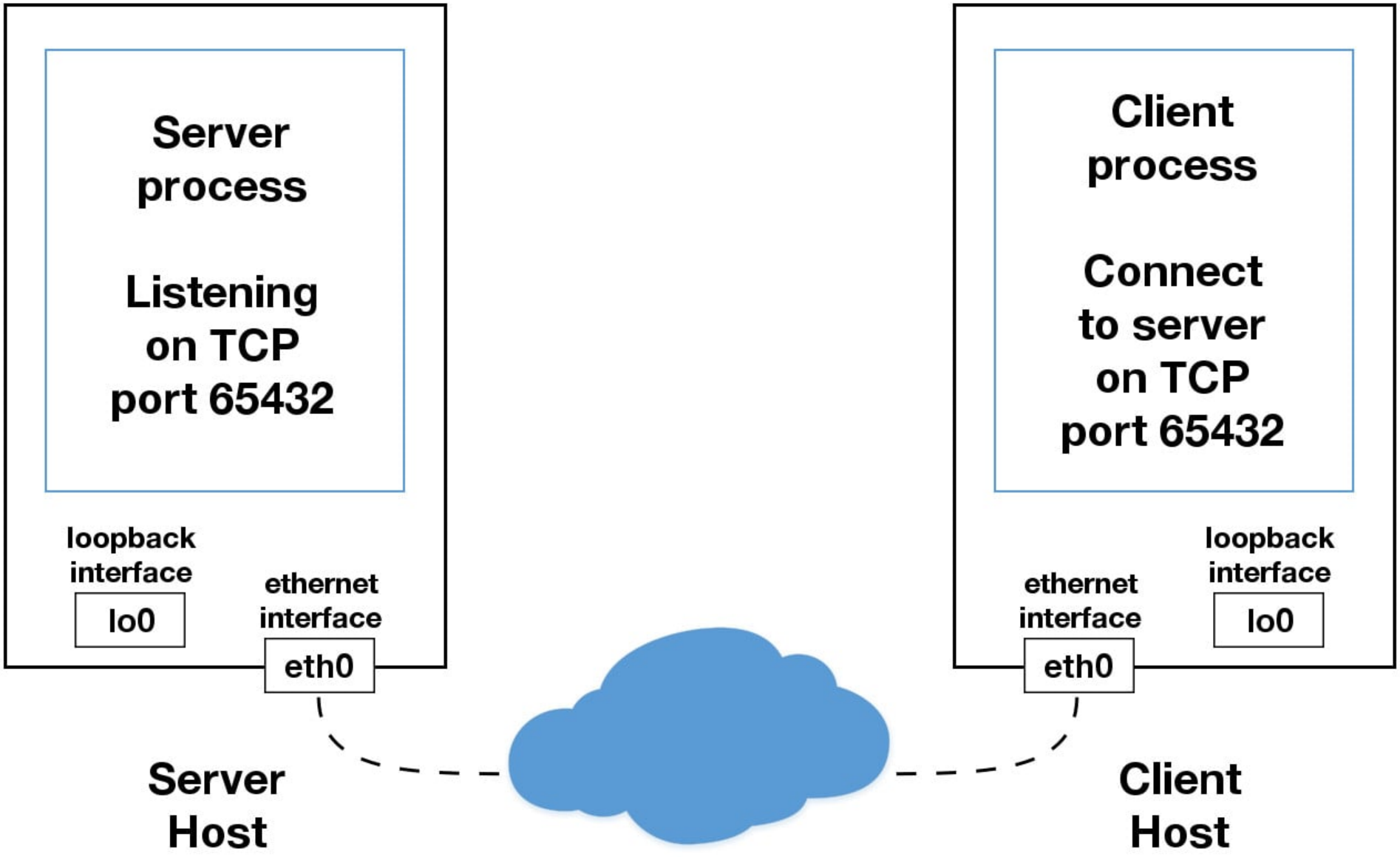
```python
import time
import concurrent.futures
import math
PRIMES = [112272535095293,112582705942171111111111111111111111,
    112272535095293,115280095190773,115797848077099, 1099726899285419]
def is_prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    sqrt_n = int(math.floor(math.sqrt(n)))
    for i in range(3, sqrt_n + 1, 2):
        if n % i == 0:
            return False
    return True
def main():
    with concurrent.futures.ProcessPoolExecutor() as executor:
        for number, prime in zip(PRIMES, executor.map(is_prime,
PRIMES)):
            print('%d is prime: %s' % (number, prime))

if __name__ == '__main__':
    main()
```

Kết quả

112272535095293 is prime: True
1125827059421711111111111111111111 is prime: False
112272535095293 is prime: True
115280095190773 is prime: True
115797848077099 is prime: True
1099726899285419 is prime: False

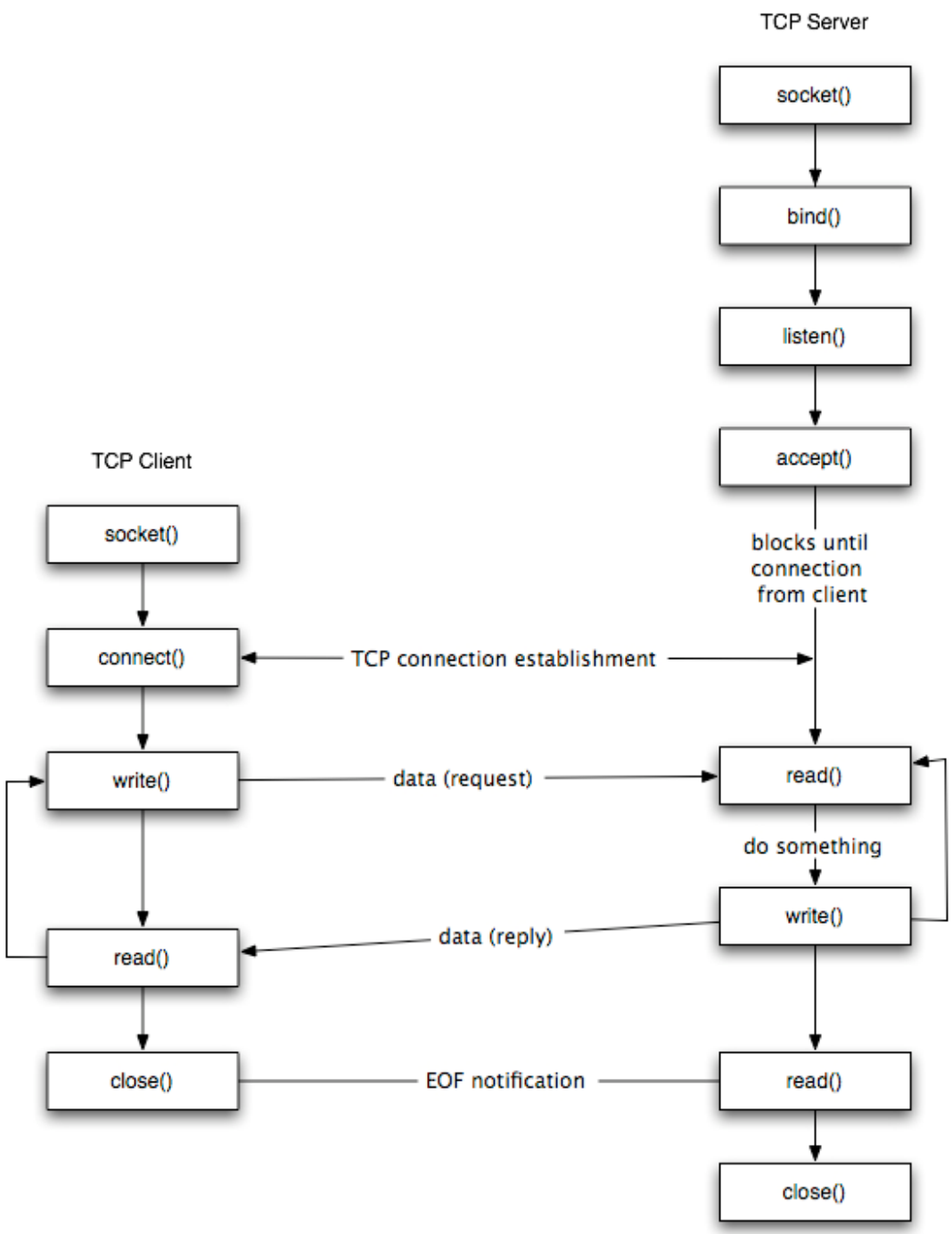
```python
from concurrent.futures import ProcessPoolExecutor
import concurrent.futures
import math
def is_prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    sqrt_n = int(math.floor(math.sqrt(n)))
    for i in range(3, sqrt_n + 1, 2):
        if n % i == 0:
            return False
    return True
def main():
    with ProcessPoolExecutor(max_workers=2) as executor:
        future1 = executor.submit(is_prime, 2222222222222222222)
        future2 = executor.submit(is_prime, 87178291199)
    print(future1.result(), future2.result())
if __name__ == '__main__':
    main()
```

Socket

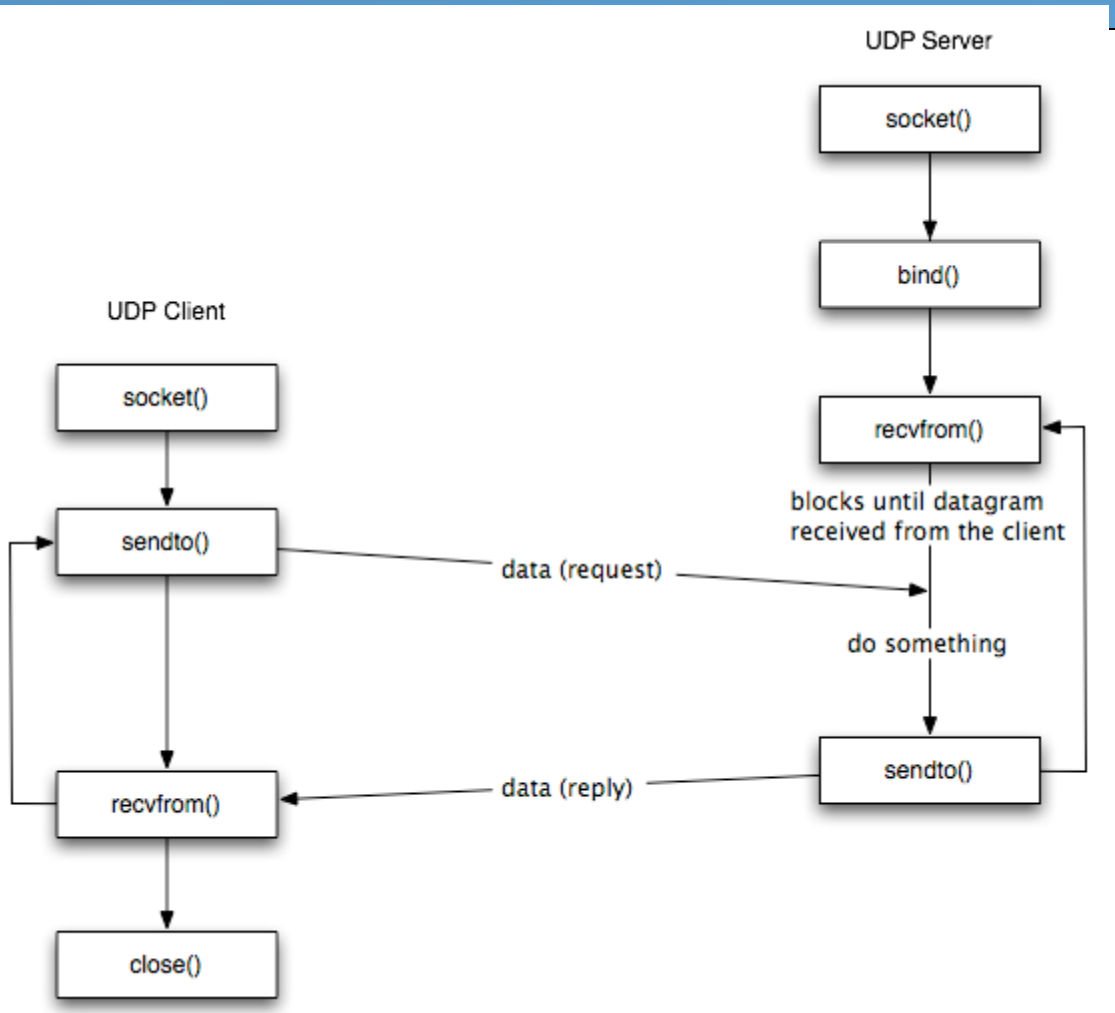

Server process
Listening on TCP port 65432
loopback interface
lo0
ethernet interface
eth0
Server Host
Client process
Connect to server on TCP port 65432
ethernet interface
eth0
loopback interface
lo0
Client Host

TCP

- Transmission Control Protocol
- Reliable: có sữa lỗi và truyền lại gói tin bị lỗi/mất, bảo đảm thứ tự truyền nhận.



UDP


UDP Server
socket()
bind()
recvfrom()
blocks until datagram
received from the client
do something
sendto()
UDP Client
socket()
sendto()
recvfrom()
close()
data (request)
data (reply)

Socket

- Module:

```
import socket
```

- Khởi tạo

```
s = socket.socket(family, type, protocol=0)
```

  - **socket_family**: AF_UNIX hoặc AF_INET (Unix domain, Internet domain)
  - **socket_type:** SOCK_STREAM hoặc SOCK_DGRAM (TCP, UDP)
  - **protocol:** mặc định 0.

Socket

- Phương thức socket (s) phía server
  - `s.bind((hostname, port))`
  - `s.listen(backlog)`
  - `s.accept()`
- Phương thức socket (s) phía client
  - `s.connect(hostname, port)`
- Phương thức socket (s) cả hai phía
  - `s.recv(buflen[, flags])`
  - `s.send(data[, flags])`
  - `s.recvfrom(buflen[, flags])`
  - `s.sendto(data[,flags], (addr, port))`
  - `s.close()`
  - `socket.gethostname()`
  - `socket.gethostbyname(hostname)`

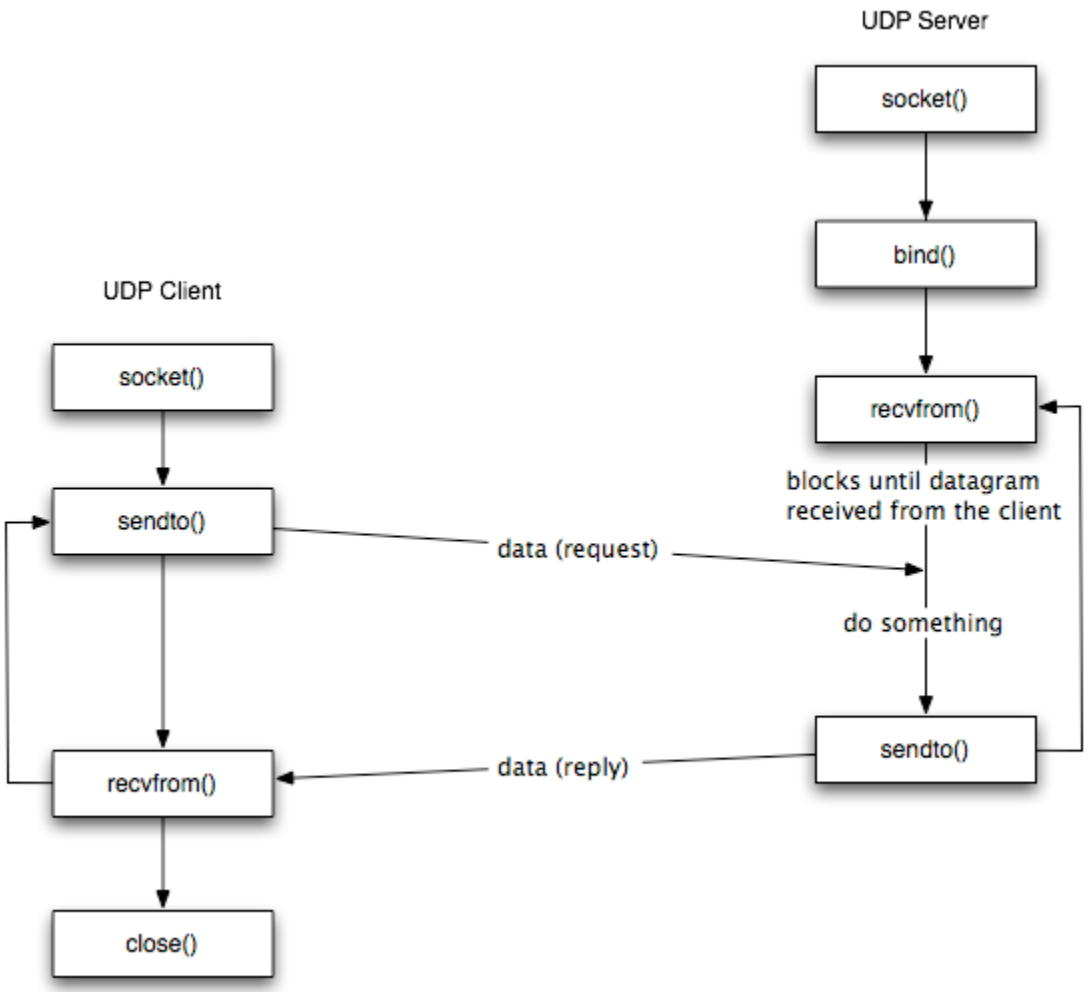
time_server.py

```
import socket
import datetime
# initializing socket
s = socket.socket()
host = socket.gethostname()
port = 12345
# binding port and host
s.bind((host, port))
# waiting for a client to connect
s.listen(5)
while True:
   # accept connection
   c, addr = s.accept()
   print ('got connection from addr', addr)
   date = datetime.datetime.now()
   d = str(date)
   # sending data type should be string and encode before
sending
   c.send(d.encode())
   c.close()
```

time_client.py

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 12345

# connect to host
s.connect((host, port))

# recv message and decode here 1024 is buffer size.
print (s.recv(1024).decode())
s.close()
```

udp_server.py



```
import socket
localIP = "127.0.0.1“, localPort = 20001, bufferSize = 1024
UDPServerSocket = socket.socket(family = socket.AF_INET, type
= socket.SOCK_DGRAM)
UDPServerSocket.bind((localIP, localPort))
print("UDP server up and listening")
di={'17BIT0382':'vivek','17BEC0647':'shikhar'}
while(True):
   # receiving from client
   name, addr1 = UDPServerSocket.recvfrom(bufferSize)
   pwd, addr1 = UDPServerSocket.recvfrom(bufferSize)
   name = name.decode() ,   pwd = pwd.decode() ,   msg =''
   if name not in di:
       msg ='name does not exists’, flag = 0
   for i in di:
      if i == name:
          if di[i]== pwd:
              msg ="pwd match“, flag = 1
          else:
              msg ="pwd wrong“,  bytesToSend = str.encode(msg)
      # sending encoded status of name and pwd
      UDPServerSocket.sendto(bytesToSend, addr1)
```

udp_client.py

```
import socket
# user input
name = input('enter your username : ')
bytesToSend1 = str.encode(name)
password = input('enter your password : ')
bytesToSend2 = str.encode(password)
serverAddrPort = ("127.0.0.1", 20001)
bufferSize = 1024
# connecting to hosts
UDPClientSocket = socket.socket(family = socket.AF_INET, type
= socket.SOCK_DGRAM)
# sending username by encoding it
UDPClientSocket.sendto(bytesToSend1, serverAddrPort)
# sending password by encoding it
UDPClientSocket.sendto(bytesToSend2, serverAddrPort)
# receiving status from server
msgFromServer = UDPClientSocket.recvfrom(bufferSize)
msg = "Message from Server
{}".format(msgFromServer[0].decode())
print(msg)
```

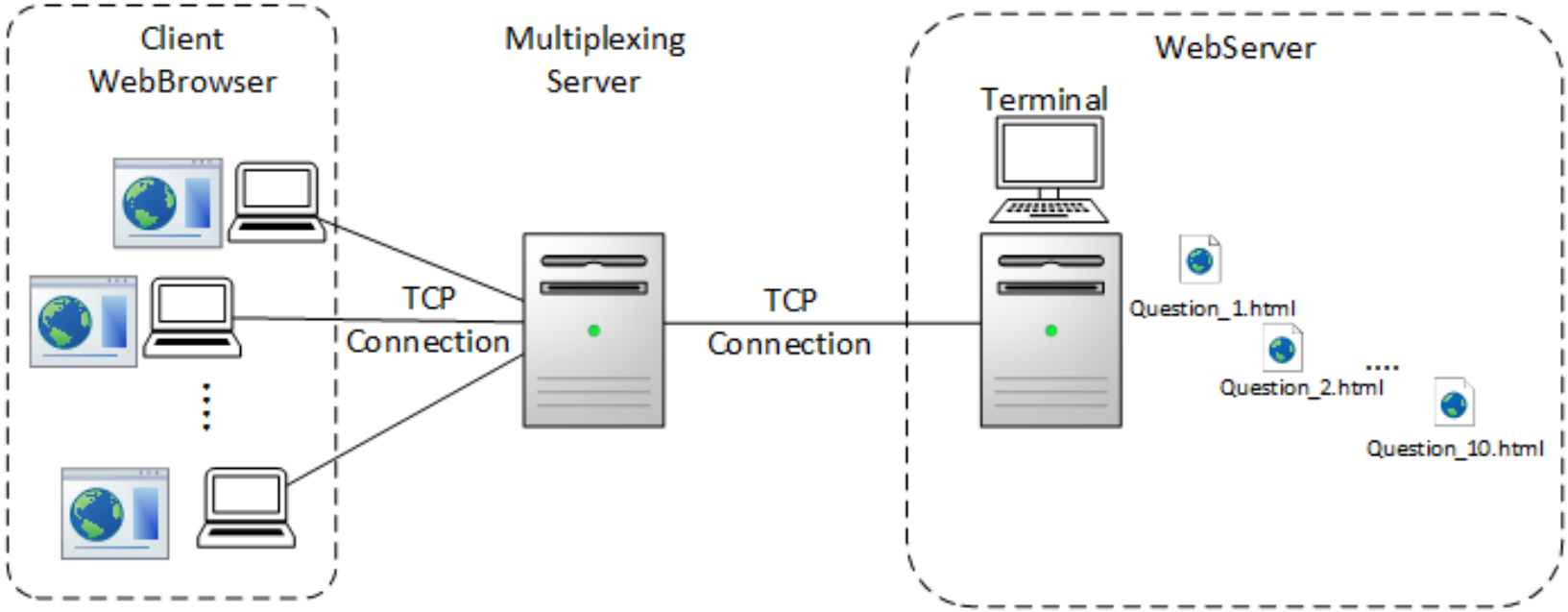
echo_server.py

```
import socket
PORT = 65432  # Port to listen on (non-privileged ports are >
1023)
HOST = "127.0.0.1"
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

echo_client.py

```
import socket

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b"Hello, world")
    data = s.recv(1024)

print(f"Received {data!r}")
```

Phục vụ song song – Multi Thread



echo_server.py

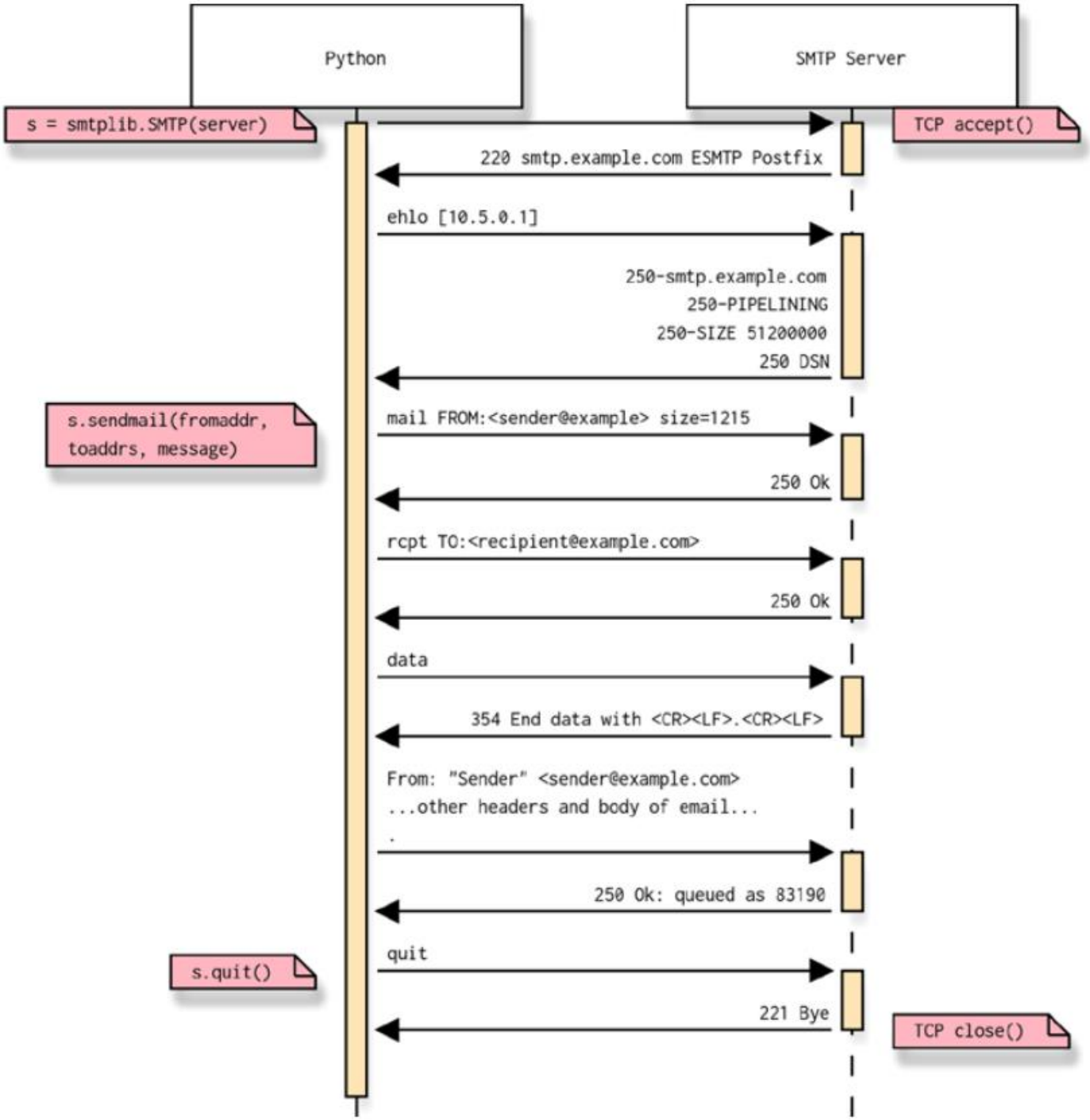
```
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self,clientAddress,clientsocket):
        threading.Thread.__init__(self)
        self.csocket = clientsocket
        print ("New connection added: ", clientAddress)
    def run(self):
        print ("Connection from : ", clientAddress)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
              break
            print ("from client", msg)
            self.csocket.send(bytes(msg,'UTF-8'))
        print ("Client at ", clientAddress , "
disconnected...")
```

echo_server.py

```
LOCALHOST = "127.0.0.1"
PORT = 8080
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind((LOCALHOST, PORT))
print("Server started")
print("Waiting for client request..")
while True:
    server.listen(1)
    clientsock, clientAddress = server.accept()
    newthread = ClientThread(clientAddress, clientsock)
    newthread.start()
```

echo_client.py

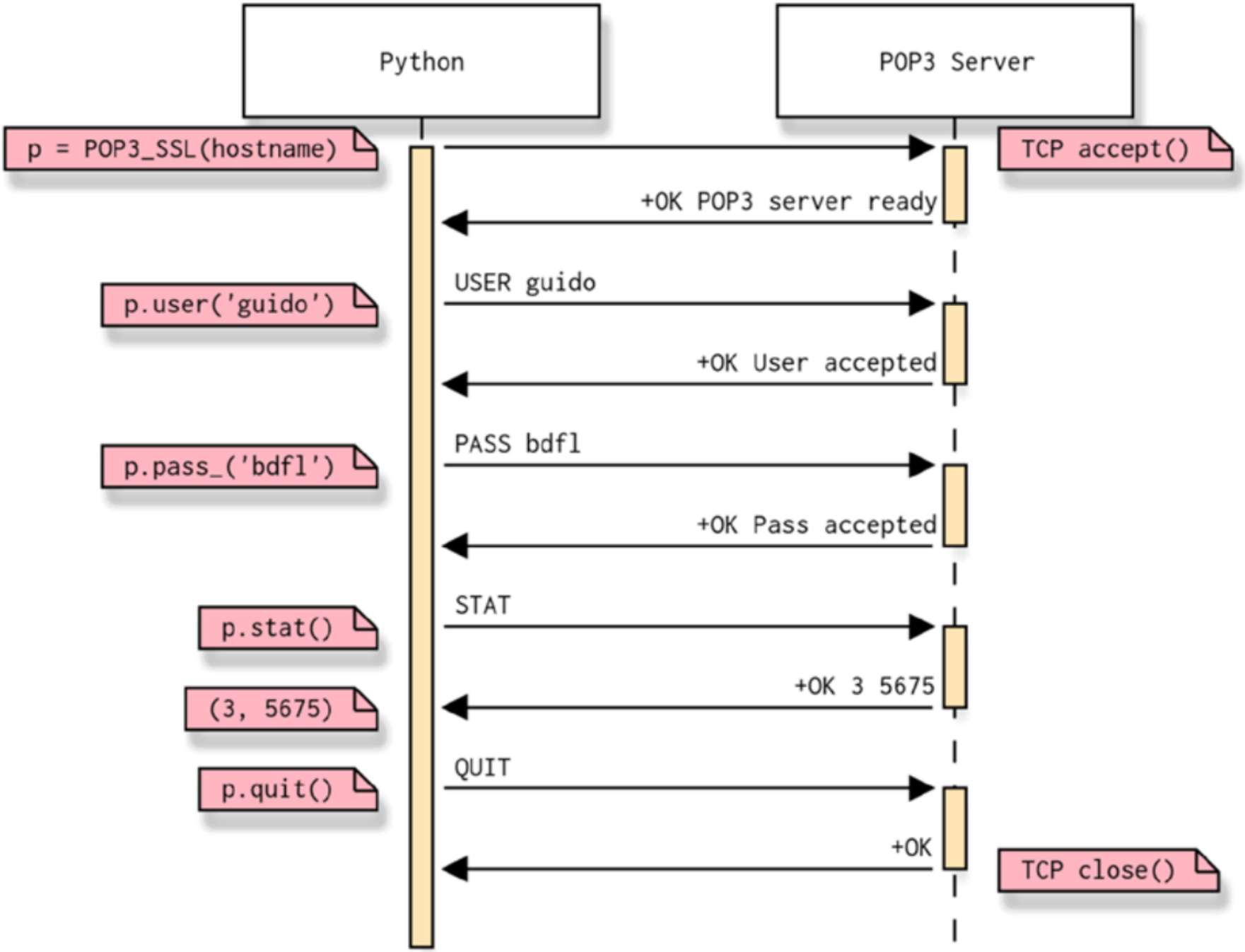
```
import socket
SERVER = "127.0.0.1"
PORT = 8080
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((SERVER, PORT))
client.sendall(bytes("This is from Client",'UTF-8'))
while True:
  in_data =  client.recv(1024)
  print("From Server :" ,in_data.decode())
  out_data = input()
  client.sendall(bytes(out_data,'UTF-8'))
  if out_data=='bye':
    break
client.close()
```


Python
SMTP Server
s = smtplib.SMTP(server)
TCP accept()
220 smtp.example.com ESMTP Postfix
ehlo [10.5.0.1]
250-smtp.example.com
250-PIPELINING
250-SIZE 51200000
250 DSN
s.sendmail(fromaddr, toaddrs, message)
mail FROM:<sender@example> size=1215
250 Ok
rcpt TO:<recipient@example.com>
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
From: "Sender" <sender@example.com>
...other headers and body of email...
.
250 Ok: queued as 83190
s.quit()
quit
221 Bye
TCP close()

```python
import argparse
import os
import getpass
import re
import sys
import smtplib
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587
def send_email(sender, recipient):
    """ Send email message """
    msg = MIMEMultipart()
    msg['Subject'] = 'Python Emaill Test'
    msg['To'] = recipient
    msg['From'] = sender
    subject = 'Python email Test'
```

```
    message = 'Images attached.'
    # attach imgae files
    files = os.listdir(os.getcwd())
    gifsearch = re.compile(".gif", re.IGNORECASE)
    files = filter(gifsearch.search, files)
    for filename in files:
        path = os.path.join(os.getcwd(), filename)
        if not os.path.isfile(path):
            continue
        img = MIMEImage(open(path, 'rb').read(),
_subtype="gif")
        img.add_header('Content-Disposition', 'attachment',
filename=filename)
        msg.attach(img)
    part = MIMEText('text', "plain")
    part.set_payload(message)
    msg.attach(part)
    # create smtp session
```

```
    session = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    session.ehlo()
    session.starttls()
    session.ehlo
    password = getpass.getpass(prompt="Enter your Google
password: ")
    session.login(sender, password)
    session.sendmail(sender, recipient, msg.as_string())
    print ("Email sent.")
    session.quit()
if __name__ == '__main__':
    send_email( "dqbao@dthu.edu.vn", "baodhspdt@gmail.com")
```


Python
POP3 Server
p = POP3_SSL(hostname)
TCP accept()
+OK POP3 server ready
USER guido
p.user('guido')
+OK User accepted
PASS bdfl
p.pass_('bdfl')
+OK Pass accepted
STAT
p.stat()
+OK 3 5675
(3, 5675)
QUIT
p.quit()
+OK
TCP close()

Lưu ý

Cho phép less security app access Gmail để test các ví dụ đăng nhập Gmail bằng Python

https://myaccount.google.com/u/1/lesssecureapps?pli=1&rapt=AEjHL4NEF7LRQySkoM3pu_QOKaD0Puyi4A8n0QacBP7nNhadyrF9nS06B9U6S9-0-IL2Z_tyl8mibfAtA_In0dOD

Google Account

← Less secure app access

Some apps and devices use less secure sign-in technology, which makes your account vulnerable. You can turn off access for these apps, which we recommend, or turn it on if you want to use them despite the risks. Google will automatically turn this setting OFF if it's not being used. Learn more

Allow less secure apps: ON

Settings

General | Labels | Inbox | Accounts | Filters and Blocked Addresses | Forwarding and POP/IMAP | Add-ons | Chat and Meet | Advanced | Offline | Themes

Forwarding:
Learn more

Add a forwarding address

Tip: You can also forward only some of your mail by creating a filter!

POP download:
Learn more

1. Status: POP is enabled for all mail that has arrived since 11/8/12
- Enable POP for **all mail** (even mail that's already been downloaded)
- Enable POP for **mail that arrives from now on**
- **Disable** POP

2. When messages are accessed with POP keep Trường Đại học Đồng Tháp Mail's copy in the Inbox

3. Configure your email client (e.g. Outlook, Eudora, Netscape Mail)
Configuration instructions

IMAP access:
(access Trường Đại học Đồng Tháp Mail from other clients using IMAP)
Learn more

Status: IMAP is enabled
- Enable IMAP
- Disable IMAP

When I mark a message in IMAP as deleted:
- Auto-Expunge on - Immediately update the server. (default)
- Auto-Expunge off - Wait for the client to update the server.

Http Server đơn giản

Server.py

```python
import http.server
import socketserver

PORT = 8080
Handler = http.server.SimpleHTTPRequestHandler

with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print("serving at port", PORT)
    httpd.serve_forever()
```

Index.htm

```html
<html>
    <head>
        <title>Python is awesome!</title>
    </head>
    <body>
        <h1>Afternerd</h1>
        <p>Congratulations! The HTTP Server is working!</p>
    </body>
</html>
```

Http Server đơn giản

Windows PowerShell

```
D:\app_net> python .\webserver.py
ving at port 8080
```

Python is awesome!

localhost:8080

Afternerd

Congratulations! The HTTP Server is working!

Download google email bằng pop3

```
#!/usr/bin/env python
import getpass
import poplib
GOOGLE_POP3_SERVER = 'pop.googlemail.com'
def download_email(username):
    mailbox = poplib.POP3_SSL(GOOGLE_POP3_SERVER, '995')
    mailbox.user(username)
    password = getpass.getpass(prompt="Enter your Google
password: ")
    mailbox.pass_(password)
    num_messages = len(mailbox.list()[1])
    print ("Total emails: %s" %num_messages)
    print ("Getting last message")
    for msg in mailbox.retr(num_messages)[1]:
        print (msg)
    mailbox.quit()
if __name__ == '__main__':
    username = "dqbao@dthu.edu.vn"
    download_email(username)
```

Remote Procedure Call (RPC)

```
import operator, math
from xmlrpc.server import SimpleXMLRPCServer
from functools import reduce
def main():
    server = SimpleXMLRPCServer(('127.0.0.1', 7001))
    server.register_introspection_functions()
    server.register_multicall_functions()
    server.register_function(addtogether)
    server.register_function(quadratic)
    server.register_function(remote_repr)
    print("Server ready")
    server.serve_forever()
def addtogether(*things):
        return reduce(operator.add, things)
def quadratic(a, b, c):
    """Determine `x` values satisfying: `a` * x*x + `b` * x + c == 0"""
    b24ac = math.sqrt(b*b - 4.0*a*c)
    return list(set([ (-b-b24ac) / 2.0*a,
                      (-b+b24ac) / 2.0*a ]))
def remote_repr(arg):
    return arg
if __name__ == '__main__':
    main()
```

Remote Procedure Call (RPC)

```python
#!/usr/bin/env python3
# Foundations of Python Network Programming, Third Edition
# XML-RPC client performing a multicall
import xmlrpc.client

def main():
    proxy = xmlrpc.client.ServerProxy('http://127.0.0.1:7001')
    multicall = xmlrpc.client.MultiCall(proxy)
    multicall.addtogether('a', 'b', 'c')
    multicall.quadratic(2, -4, 0)
    multicall.remote_repr([1, 2.0, 'three'])
    for answer in multicall():
        print(answer)

if __name__ == '__main__':
    main()
```

Remote Procedure Call (RPC)

```
#jsonrpc_server.py
# JSON-RPC server needing "pip install jsonrpclib-pelix"
from jsonrpclib.SimpleJSONRPCServer import SimpleJSONRPCServer
def lengths(*args):
    """Measure the length of each input argument.
    results = []
    for arg in args:
        try:
            arglen = len(arg)
        except TypeError:
            arglen = None
        results.append((arglen, arg))
    return results

def main():
    server = SimpleJSONRPCServer(('localhost', 7002))
    server.register_function(lengths)
    print("Starting server")
    server.serve_forever()
if __name__ == '__main__':
    main()
```

Remote Procedure Call (RPC)

```python
#!/usr/bin/env python3
# Foundations of Python Network Programming, Third Edition
# jsonrpc_client.py
# JSON-RPC client needing "pip install jsonrpclib-pelix"

from jsonrpclib import Server

def main():
    proxy = Server('http://localhost:7002')
    print(proxy.lengths((1,2,3), 27, {'Sirius': -1.46, 'Rigel': 0.12}))

if __name__ == '__main__':
    main()
```

Làm việc với Hệ QTCSDL mysql

```
D:\app_mysql>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.31-cp38-cp38-win_amd64.whl
(7.9 MB)
     |[redacted]| 7.9 MB 3.3 MB/s
Requirement already satisfied: protobuf<=3.20.1,>=3.11.0 in
c:\users\dangq\anaconda3\lib\site-packages (from mysql-connector-python)
(3.17.3)
Requirement already satisfied: six>=1.9 in c:\users\dangq\anaconda3\lib\site-
packages (from protobuf<=3.20.1,>=3.11.0->mysql-connector-python)
(1.15.0)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-8.0.31
```

Kết nối

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="dqbao“,
  charset='utf8'
)
print(mydb)
```

Tạo CSDL

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="dqbao"
)
mycursor = mydb.cursor()
#Tạo CSDL
mycursor.execute("CREATE DATABASE db1")
#Duyệt qua các tên database
mycursor.execute("SHOW DATABASES")
for x in mycursor:
  print(x)
```

Tạo CSDL

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="dqbao"
)
mycursor = mydb.cursor()
#Tạo CSDL
mycursor.execute("CREATE DATABASE db1")
#Duyệt qua các tên database
mycursor.execute("SHOW DATABASES")
for x in mycursor:
  print(x)
```

Làm việc với database

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="dqbao",
    database="db1"
)
mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name
VARCHAR(255), address VARCHAR(255))")
```

Làm việc với database

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="dqbao",
    database="db1"
)
mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES
(%s, %s)"
val = ("Dang Quoc Bao", "Cao Lanh")
mycursor.execute(sql, val)
val = ("Nguyen Van Ty", "TP Ho Chi Minh")
mycursor.execute(sql, val)
mydb.commit()
```

Làm việc với database

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="dqbao",
    database="db1"
)
mycursor = mydb.cursor()
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM customers")
myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

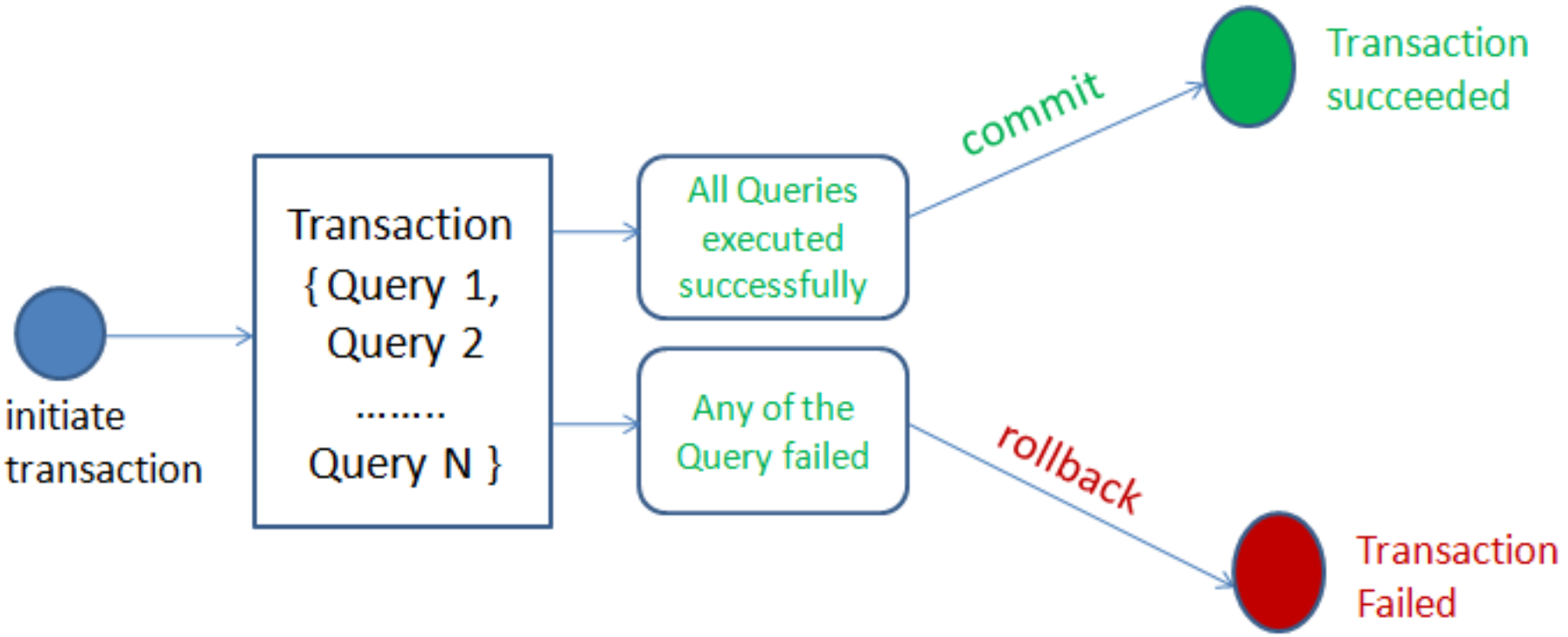
Làm việc với database

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="dqbao",
    database="db1"
)
mycursor = mydb.cursor()
sql = "UPDATE customers SET address = '140 Cao Lanh'
WHERE address = 'Cao Lanh'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) affected")
```

Làm việc với database

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="dqbao",
    database="db1"
)
mycursor = mydb.cursor()

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address =
'Mountain 21'"

mycursor.execute(sql)
```

Quản lý giao dịch transaction



Quản lý giao dịch transaction

- Một hay nhiều lệnh cập nhật CSDL.
- Xét một hoạt động chuyển tiền ngay lập tức trong một hệ thống ngân hàng.
  a) Withdrawal of money from account A
  b) Deposit Money to Account B

Nếu thao tác a) thành công nhưng b) thất bại do lỗi ngoại lệ xảy ra?=>cần quản lý transaction=> **ACID properties**

➔

- **Atomicity**: means all or nothing.
- **Consistency**: It ensures that the database remains in a consistent state after performing a transaction.
- **Isolation**: It ensures that the transaction is isolated from other transactions.
- **Durability**: It means once a transaction has been committed, it persists in the database irrespective of power loss, error, or restart system.

```
import mysql.connector
try:
    conn = mysql.connector.connect(
        host='localhost’,
       database=db2',user=‘root',  password=dqbao')
    conn.autocommit = False
    cursor = conn.cursor()
    # withdraw from account A
    sql_update_query = """Update account_A set balance = 1000
where id = 1"""
    cursor.execute(sql_update_query)
    # Deposit to account B
    sql_update_query = """Update account_B set balance = 1000
where id = 2"""
    cursor.execute(sql_update_query)
    print("Record Updated successfully ")
    # Commit your changes
    conn.commit()
Except:
    print("Failed to update record to database rollback:”)
    # reverting changes because of exception
    conn.rollback()
```

```
finally:
    # closing database connection.
    if conn.is_connected():
        cursor.close()
        conn.close()
        print("connection is closed")
```

```python
import mysql.connector

#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user =
"root",passwd = "google",database = "PythonDB")
myconn.autocommit = False
#creating the cursor object
cur = myconn.cursor()

try:
    cur.execute("delete from Employee where Dept_id = 201")
    myconn.commit()
    print("Deleted !")
except:
    print("Can't delete !")
    myconn.rollback()

myconn.close()
```

pygame

Cài đặt module

```
pip install pygame
```

http://pygame.org/
http://pygame.org/docs/ref/

- Sophisticated 2-D graphics drawing functions
- Media (images, sound F/X, music) nicely
- Respond to user input (keyboard, joystick, mouse)
- Built-in classes to represent common game objects

pygame

- pyGame consists of many **modules** of code to help you:

```
cdrom    cursors    display     draw    event
font     image      joystick    key     mouse
movie    sndarray   surfarray   time    transform
```

- To use a given module, **import** it. For example:

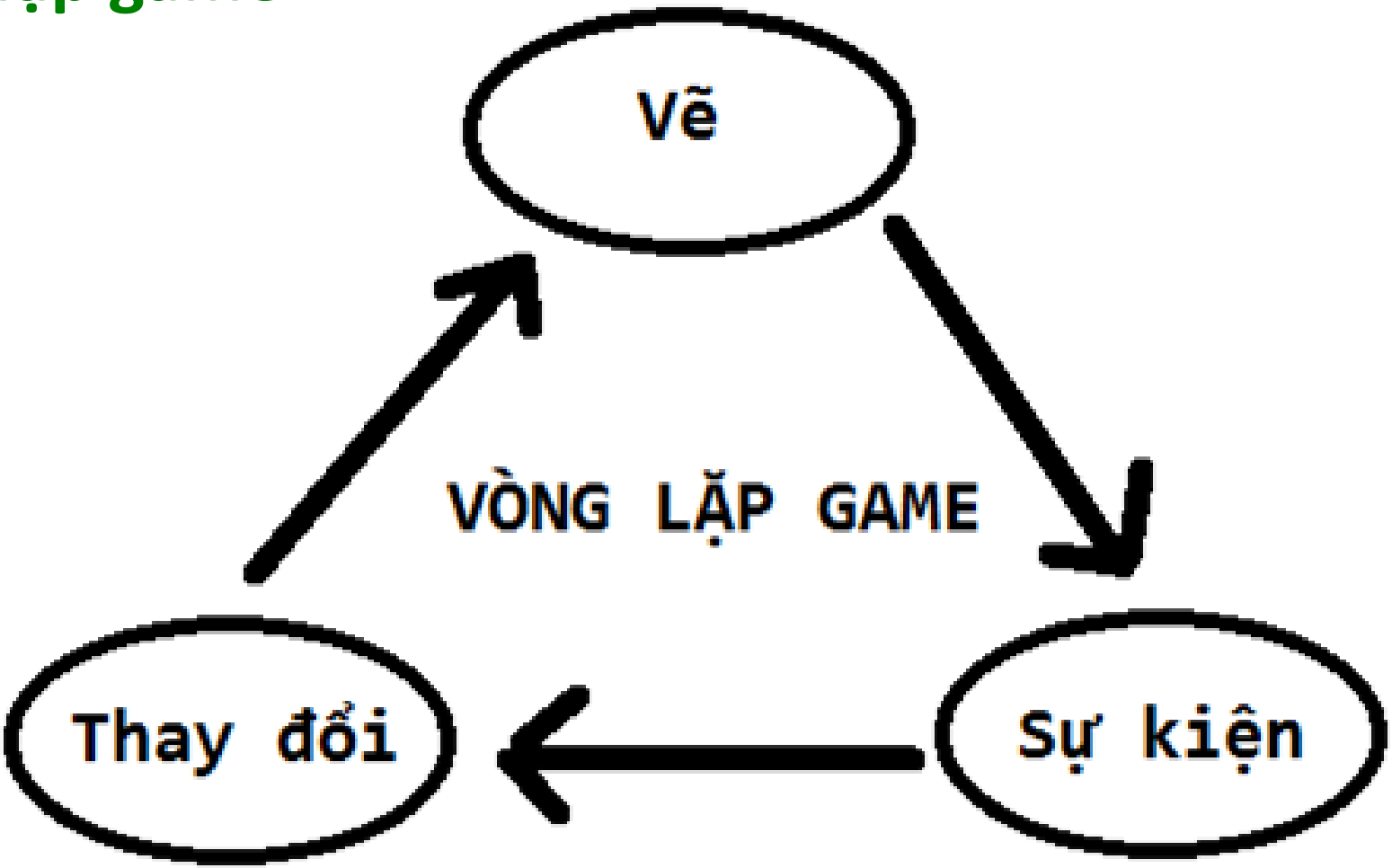
```
import pygame
from pygame import *
from pygame.display import *
```

pygame

Cài đặt module

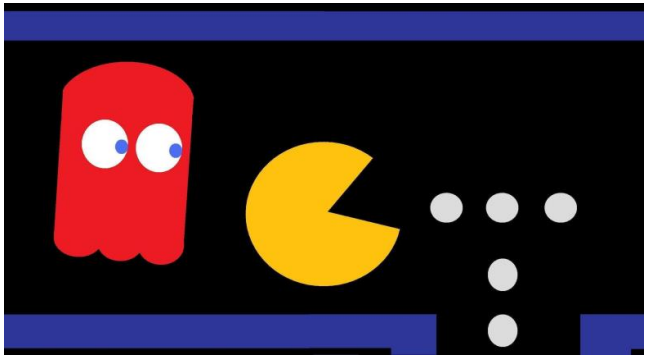
```
pip install pygame
```

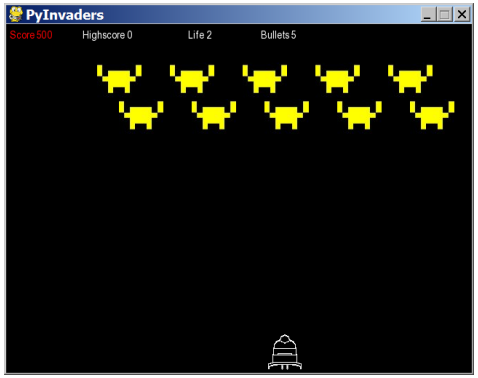
Vòng lặp game



Game fundamentals

- **sprites**: Onscreen characters or other moving objects.
- **collision detection**: Seeing which pairs of sprites touch.
- **event**: An in-game action such as a mouse or key press.
- **event loop**: Many games have an overall loop that:
  - **waits** for events to occur, **updates** sprites, **redraws** screen





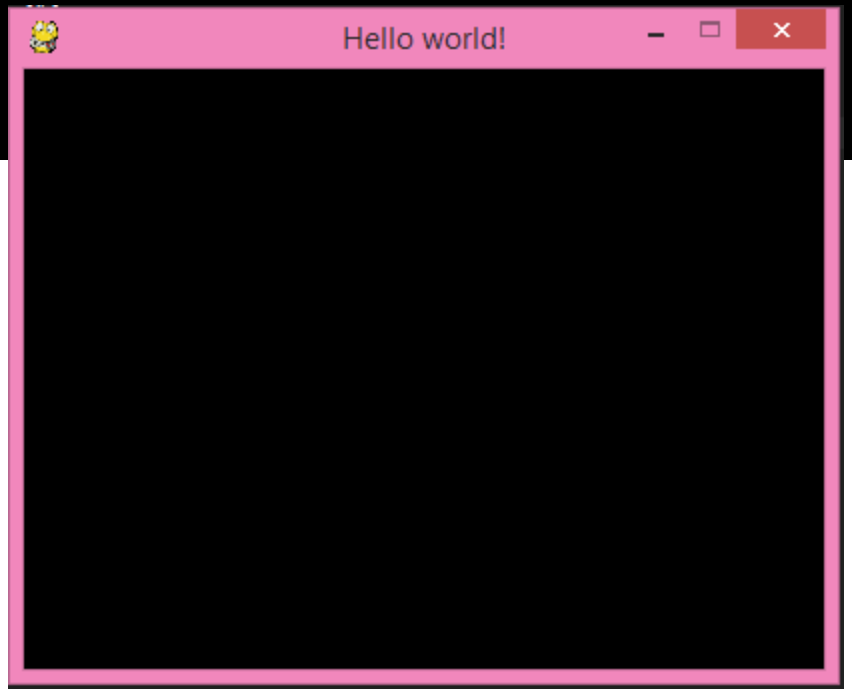
Cấu trúc CT cơ bản

pygame_template.py

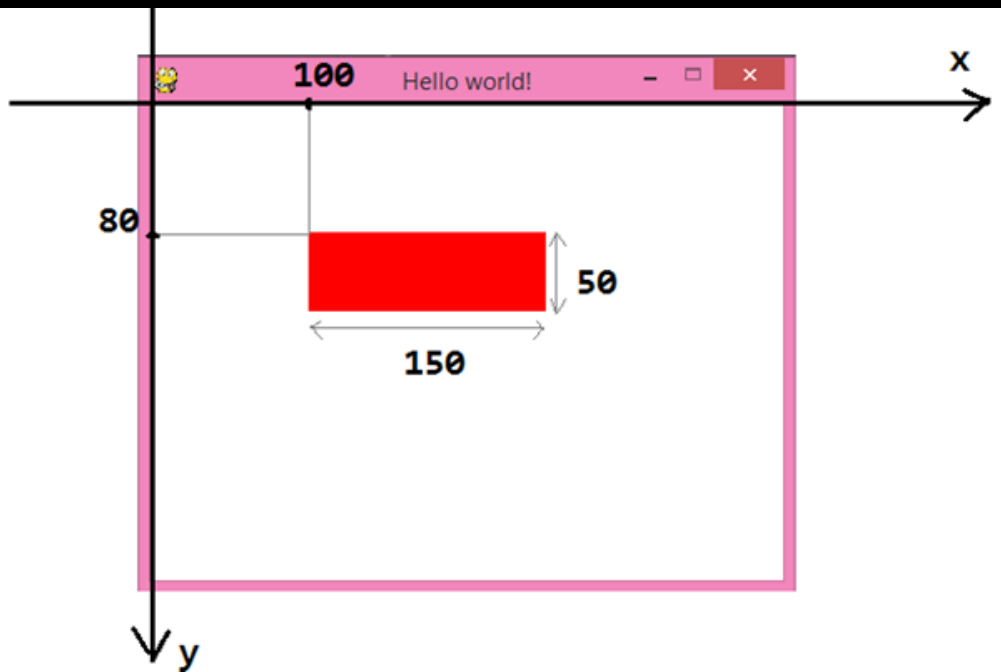
```
1   from pygame import *
2   from pygame.sprite import *
3
4   pygame.init()                          # starts up pyGame
5   screen = display.set_mode((width, height))
6   display.set_caption("window title")
7
8   create / set up sprites.
9
10  # the overall event loop
11  while True:
12      e = event.wait()                   # pause until event occurs
13      if e.type == QUIT:
14          pygame.quit()                  # shuts down pyGame
15          break
16
17      update sprites, etc.
18      screen.fill((255, 255, 255))       # white background
19      display.update()                   # redraw screen
```

pygame

```
import pygame, sys
from pygame.locals import *
pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello world!')
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
```



```python
import pygame, sys
from pygame.locals import *
pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello world!')
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    DISPLAYSURF.fill((255, 255, 255))
    pygame.draw.rect(DISPLAYSURF, (255, 0, 0), (100, 80, 150, 50))
    pygame.display.update()
```

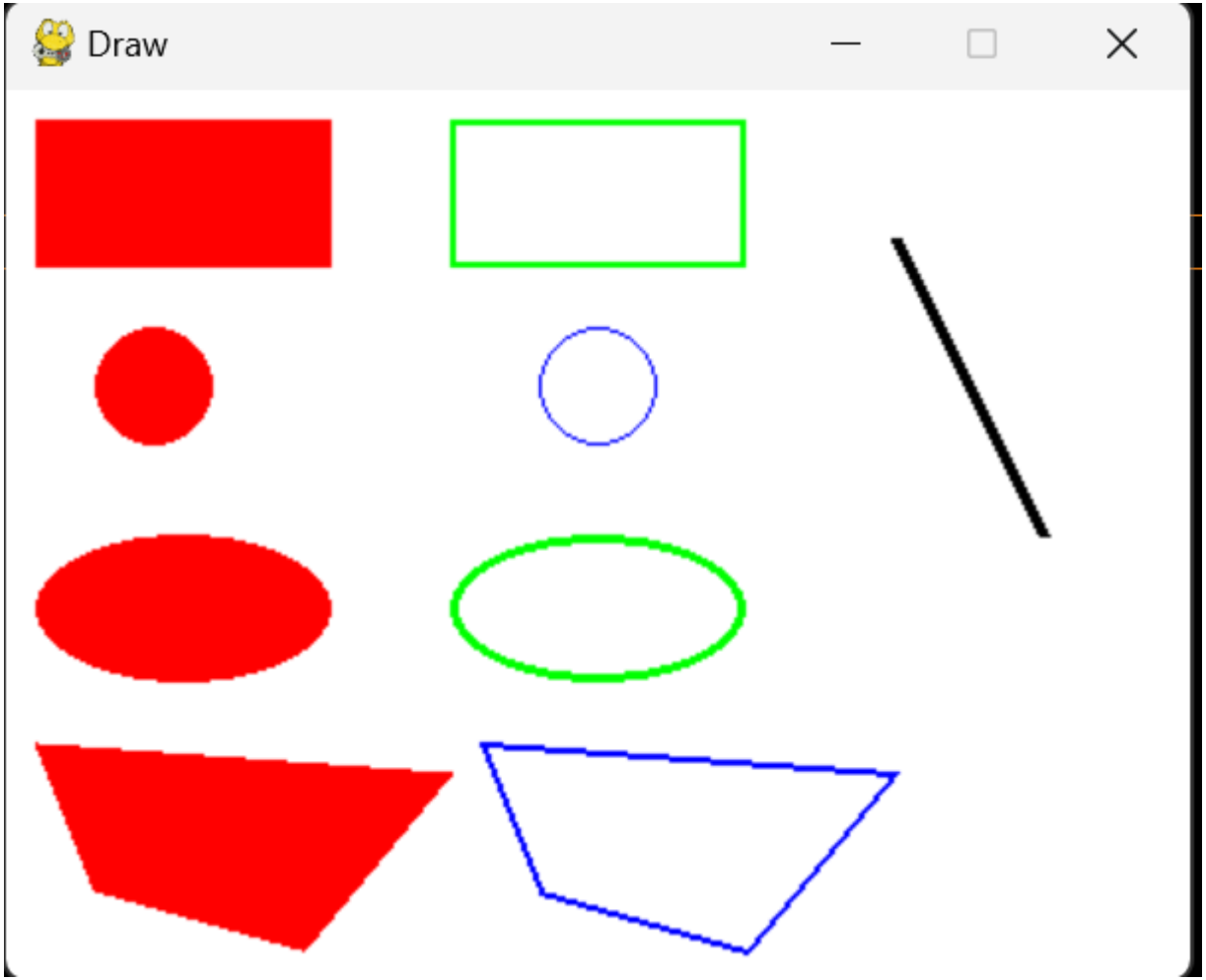


pygame

```
import pygame, sys
from pygame.locals import *

pygame.init()

DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Draw')

# Tạo sẵn các màu sắc
BLACK = (  0,   0,   0)
WHITE = (255, 255, 255)
RED   = (255,   0,   0)
GREEN = (  0, 255,   0)
BLUE  = (  0,   0, 255)
```

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    DISPLAYSURF.fill(WHITE)

    pygame.draw.rect(DISPLAYSURF, RED, (10, 10, 100, 50))# Hình chữ nhật
    pygame.draw.rect(DISPLAYSURF, GREEN, (150, 10, 100, 50), 2)# Hình
chữ nhật rỗng
    pygame.draw.circle(DISPLAYSURF, RED, (50, 100), 20) # Hình tròn
    pygame.draw.circle(DISPLAYSURF, BLUE, (200, 100), 20, 1)# Hình tròn
rỗng
    pygame.draw.ellipse(DISPLAYSURF, RED, (10, 150, 100, 50))# Hình elip
    pygame.draw.ellipse(DISPLAYSURF, GREEN, (150, 150, 100, 50), 3)#
Hình elip rỗng
    pygame.draw.polygon(DISPLAYSURF, RED, ((10, 220), (150, 230), (100
,290), (30, 270)))# Đa giác
    pygame.draw.polygon(DISPLAYSURF, BLUE, ((160, 220), (300, 230), (250
,290), (180, 270)), 2)# Đa giác rỗng
    pygame.draw.line(DISPLAYSURF, BLACK, (300, 50), (350, 150), 4)# Đoạn
thẳng
    pygame.display.update()
```

pygame



Text

```
import pygame, sys
from pygame.locals import *

WINDOWWIDTH = 400
WINDOWHEIGHT = 300

WHITE = (255, 255, 255)
RED   = (255,   0,   0)
GREEN = (  0, 255,   0)

pygame.init()

FPS = 60
fpsClock = pygame.time.Clock()

DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
pygame.display.set_caption('Text')

font = pygame.font.SysFont('consolas', 30)
textSurface = font.render('Hello world!', True, GREEN, RED)
```

Text

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    DISPLAYSURF.fill(WHITE)
    DISPLAYSURF.blit(textSurface, (50, 100))

    pygame.display.update()
    fpsClock.tick(FPS)
```

```python
from pygame.locals import *

WINDOWWIDTH = 400 # Chiều dài cửa sổ
WINDOWHEIGHT = 300 # Chiều cao cửa sổ

WHITE = (255, 255, 255)
RED   = (255,   0,   0)
GREEN = (  0, 255,   0)

pygame.init()

### Xác định FPS ###
FPS = 60
fpsClock = pygame.time.Clock()

DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
pygame.display.set_caption('Animation')

### Tạo surface và vẽ hình chiếc xe ###
car_x = 0 # Hoành độ của xe
carSurface = pygame.Surface((100, 50), SRCALPHA)
pygame.draw.polygon(carSurface, RED, ((15, 0), (65, 0), (85, 15),
(100, 15), (100, 40), (0, 40), (0, 15)))
pygame.draw.circle(carSurface, GREEN, (15, 40), 10)
pygame.draw.circle(carSurface, GREEN, (85, 40), 10)
```

Chuyển động

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    DISPLAYSURF.fill(WHITE)
       # Vẽ carSurface tai tọa độ(x, y)
    DISPLAYSURF.blit(carSurface, (car_x, 100))

    ### Thay đổi vị trí carSurface ###
    car_x += 2
    if car_x + 100 > WINDOWWIDTH:
        car_x = 0

    pygame.display.update()
    fpsClock.tick(FPS)
```

Bắt sự kiện

```
import pygame, sys
from pygame.locals import *
WINDOWWIDTH = 400 # Chiều dài cửa sổ
WINDOWHEIGHT = 300 # Chiều cao cửa sổ
WHITE = (255, 255, 255)
RED   = (255,   0,   0)
GREEN = (  0, 255,   0)
pygame.init()
### Xác định FPS ###
FPS = 60
fpsClock = pygame.time.Clock()
DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH,
WINDOWHEIGHT))
pygame.display.set_caption('Event')
```

Bắt sự kiện

```
class Car():
    def __init__(self):
        self.x = 100 # Vị trí của xe

        ## Tạo surface và thêm hình chiếc xe vào ##
        self.surface = pygame.image.load('car.png')

    def draw(self): # Hàm dùng để vẽ xe
        DISPLAYSURF.blit(self.surface, (self.x, 100))

    def update(self, moveLeft, moveRight): # Hàm dùng để thay
đổi vị trí xe
        if moveLeft == True:
            self.x -= 2
        if moveRight == True:
            self.x += 2

        if self.x + 100 > WINDOWWIDTH:
            self.x = WINDOWWIDTH - 100
        if self.x < 0:
```

```
car = Car()
moveLeft = False
moveRight = False
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

        if event.type == KEYDOWN:
            if event.key == K_LEFT:
                moveLeft = True
            if event.key == K_RIGHT:
                moveRight = True

        if event.type == KEYUP:
            if event.key == K_LEFT:
                moveLeft = False
            if event.key == K_RIGHT:
                moveRight = False

    DISPLAYSURF.fill(WHITE)
```

Tìm hiểu thêm

- https://realpython.com/pygame-a-primer/?fbclid=IwAR1OCewMu3kmpK2iRX89K0OKxnkTSVOgK4rgIvNVQP4qWc_qgD0PNru8Hhk#players
- https://pythonguides.com/python-pygame-tutorial/
- https://pythonprogramming.altervista.org/flappygame-made-with-pygame/

Surfaces

```
screen = display.set_mode((width, height))    # a surface
```

- In Pygame, every 2D object is an object of type `Surface`
  - The screen object, each game character, images, etc.
  - Useful methods in each `Surface` object:

`Surface((`**width**`, `**height**`))`	constructs new Surface of given size
`fill((`**red**`, `**green**`, `**blue**`))`	paints surface in given color *(rgb 0-255)*
`get_width(), get_height()`	returns the dimensions of the surface
`get_rect()`	returns a `Rect` object representing the x/y/w/h bounding this surface
`blit(`**surface**`, `**coords**`)`	draws another surface onto this surface at the given coordinates

  - after changing any surfaces, must call `display.update()`

Sprites

- **Sprites**: Onscreen characters or other moving objects.

- A sprite has data/behavior such as:
  - its **position** and **size** on the screen
  - an **image** or shape for its appearance
  - the ability to **collide** with other sprites
  - whether it is **alive** or on-screen right now
  - might be part of certain "**groups**" (enemies, food, ...)

- In pyGame, each type of sprite is represented as a subclass of the class `pygame.sprite.Sprite`

A rectangular sprite

```
from pygame import *
from pygame.sprite import *

class name(Sprite):
    def __init__(self):                    # constructor
        Sprite.__init__(self)
        self.image = Surface(width, height)
        self.rect = Rect(leftX, topY, width, height)

    other methods (if any)
```

- Important fields in every sprite:
  - `image` - the image or shape to draw for this sprite (a `Surface`)
    - as with screen, you can `fill` this or draw things onto it
  - `rect` - position and size of where to draw the sprite (a `Rect`)
- Important methods: `update`, `kill`, `alive`

Rect methods

`clip(`**rect**`)` *	crops this rect's size to bounds of given rect
`collidepoint(`**p**`)`	`True` if this `Rect` contains the point
`colliderect(`**rect**`)`	`True` if this `Rect` touches the rect
`collidelist(`**list**`)`	`True` if this `Rect` touches any rect in the list
`collidelistall(`**list**`)`	`True` if this `Rect` touches all rects in the list
`contains(`**rect**`)`	`True` if this `Rect` completely contains the rect
`copy()`	returns a copy of this rectangle
`inflate(`**dx**`,` **dy**`)` *	grows size of rectangle by given offsets
`move(`**dx**`,` **dy**`)` *	shifts position of rectangle by given offsets
`union(`**rect**`)` *	smallest rectangle that contains this and rect

* Many methods, rather than mutating, return a new rect.

- To mutate, use `_ip` (in place) version, e.g. `move_ip`

A Sprite using an image

```
from pygame import *
from pygame.sprite import *

class name(Sprite):
   def __init__(self):                # constructor
      Sprite.__init__(self)
      self.image = image.load("filename").convert()
      self.rect = self.image.get_rect().move(x, y)

   other methods (if any)
```

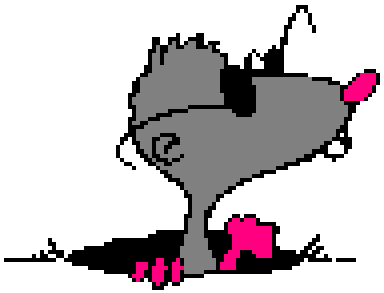
- When using an image, you load it from a file with `image.load` and then use its size to define the `rect` field
- Any time you want a sprite to move on the screen, you must change the state of its `rect` field.

Setting up sprites

- When creating a game, we think about the sprites.
  - What sprites are there on the screen?
  - What data/behavior should each one keep track of?
  - Are any sprites similar? (If so, maybe they share a class.)

- For our Whack-a-Mole game:

```
class Mole(Sprite):
    ...
```



Sprite groups

name = `Group(`**sprite1**, **sprite2**, **...**`)`

- To draw sprites on screen, put them into a `Group`
- Useful methods of each `Group` object:

  `draw(`**surface**`)` - draws all sprites in group on a `Surface`

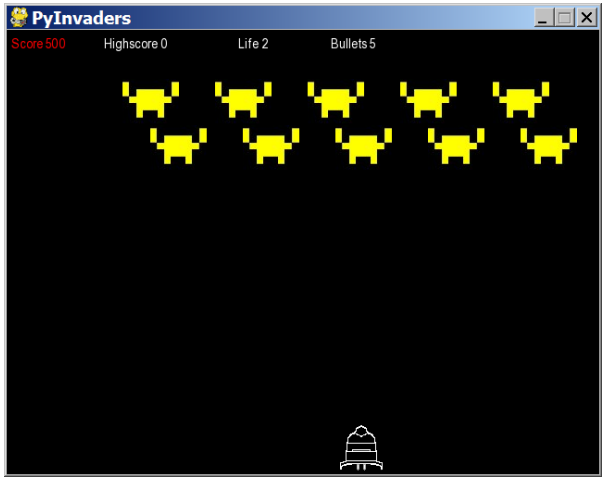
  `update()` - calls every sprite's `update` method

```
my_mole1 = Mole()     # create a Mole object
my_mole2 = Mole()
all_sprites = Group(my_mole1, other_mole2)
...
# in the event loop
all_sprites.draw(screen)
```

Events

- **event-driven programming**: When the overall program is a series of responses to user actions, or "events."

- **event loop** (aka "main loop", "animation loop"):
  Many games have an overall loop to do the following:
  - **wait** for an event to occur, or
    wait a certain interval of time
  - **update** all game objects (location, etc.)
  - **redraw** the screen
  - repeat



The event loop

- In an *event loop*, you wait for something to happen, and then depending on the kind of event, you process it:

```
while True:
    e = event.wait()        # wait for an event
    if e.type == QUIT:
        pygame.quit()       # exit the game
        break
    elif e.type == type:
        code to handle some other type of events;
    elif ...
```

Mouse events

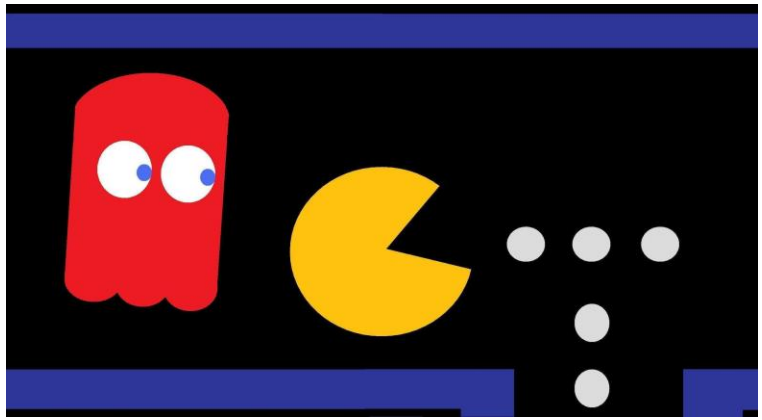
- Mouse actions lead to events with specific types:
  - **press** button down: `MOUSEBUTTONDOWN`
  - **release** button: `MOUSEBUTTONUP`
  - **move** the cursor: `MOUSEMOTION`

- At any point you can call `mouse.get_pos()` which returns the mouse's current position as an (x, y) tuple.

```
e = event.wait()
if e.type == MOUSEMOTION:
    pt = mouse.get_pos()
    x, y = pt
    ...
```

Collision detection

- **collision detection**: Examining pairs of sprites to see if they are touching each other.
  - e.g. seeing whether sprites' bounding rectangles intersect
  - usually done after events occur,
    or at regular timed intervals
  - can be complicated and error-prone
    - optimizations: *pruning* (only comparing some sprites, not all), ...



Collisions btwn. rectangles

- Recall: Each `Sprite` contains a `Rect` collision rectangle stored as a field named `rect`

- `Rect` objects have useful methods for detecting collisions between the rectangle and another sprite:

`collidepoint(`**p**`)`	returns `True` if this `Rect` contains the point
`colliderect(`**rect**`)`	returns `True` if this `Rect` touches the rect

```
if sprite1.rect.colliderect(sprite2.rect):
    # they collide!
    ...
```

Collisions between groups

global pyGame functions to help with collisions:

`spritecollideany(`**sprite**`, `**group**`)`

- Returns `True` if sprite has collided with any sprite in the group

`spritecollide(`**sprite**`, `**group**`, `**kill**`)`

- Returns a list of all sprites in **group** that collide with **sprite**
- If **kill** is `True`, a collision causes **sprite** to be deleted/killed

`groupcollide(`**group1**`, `**group2**`, `**kill1**`, `**kill2**`)`

- Returns list of all sprites in **group1** that collide with **group2**

Drawing text: Font

- Text is drawn using a `Font` object:

  name `= Font(`**filename**`,` **size**`)`

  - Pass `None` for the file name to use a default font.

- A `Font` draws text as a `Surface` with its `render` method:

  name`.render("`**text**`", True, (`**red**`,` **green**`,` **blue**`))`

  Example:

```
my_font = Font(None, 16)
text = my_font.render("Hello", True, (0, 0, 0))
```

Displaying text

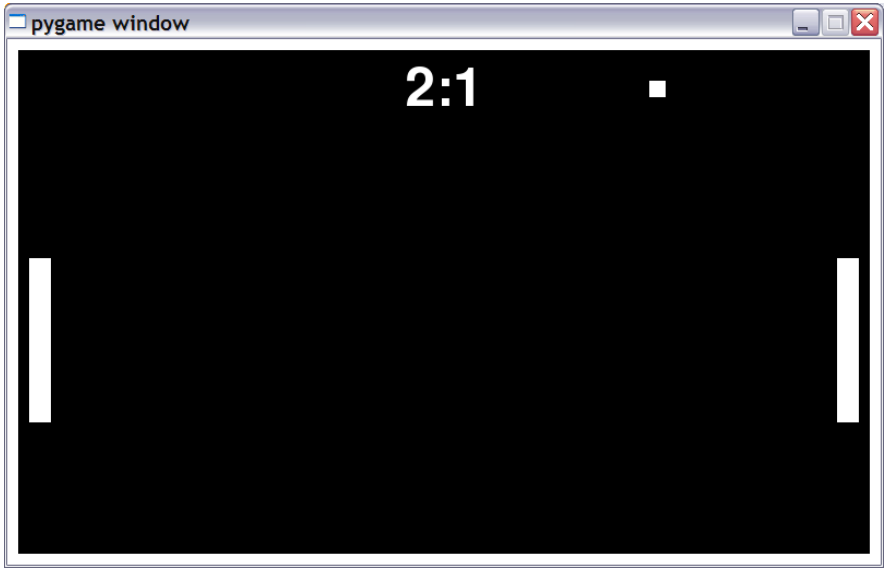
- A `Sprite` can be text by setting that text's `Surface` to be its `.image` property.

Example:

```
class Banner(Sprite):
    def __init__(self):
        my_font = Font(None, 24)
        self.image = my_font.render("Hello", True, \
                                    (0, 0, 0))
        self.rect = self.image.get_rect().move(50,70)
```

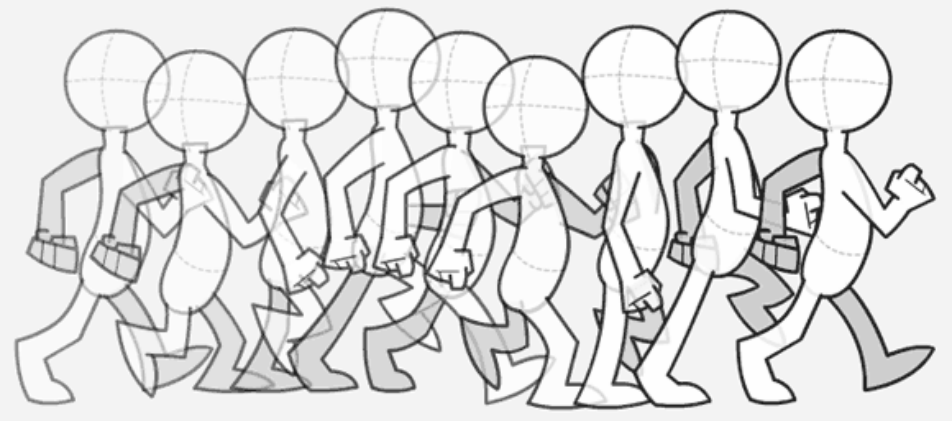
Exercise: Pong

- Let's create a Pong game with a bouncing ball and paddles.
  - 800x480 screen, 10px white border around all edges
  - 15x15 square ball bounces off of any surface it touches
  - two 20x150 paddles move when holding Up/Down arrows
  - game displays score on top/center of screen in a 72px font



Animation

- Many action games, rather than waiting for key/mouse input, have a constant **animation** timer.
  - The timer generates events at regular intervals.
  - On each event, we can move/update all sprites, look for collisions, and redraw the screen.



Timer events

`time.set_timer(USEREVENT, `**delayMS**`)`

- Animation is done using **timers**
  - Events that automatically occur every *delayMS* milliseconds; they will have a type of `USEREVENT`
  - Your event loop can check for these events. Each one is a "frame" of animation

```
while True:
    e = event.wait()
    if e.type == USEREVENT:
        # the timer has ticked
        ...
```

Key presses

- key presses lead to `KEYDOWN` and `KEYUP` events
- `key.get_pressed()` returns an array of keys held down
  - the array indexes are constants like `K_UP` or `K_F1`
  - values in the array are booleans (`True` means pressed)
  - Constants for keys: `K_LEFT`, `K_RIGHT`, `K_UP`, `K_DOWN`, `K_a` - `K_z`, `K_0` - `K_9`, `K_F1` - `K_F12`, `K_SPACE`, `K_ESCAPE`, `K_LSHIFT`, `K_RSHIFT`, `K_LALT`, `K_RALT`, `K_LCTRL`, `K_RCTRL`, ...

```
keys_down = key.get_pressed()
if keys_down[K_LEFT]:
    # left arrow is being held down
    ...
```

Updating sprites

```
class name(Sprite):
    def __init__(self):
        ...

    def update(self):  # right by 3px per tick
        self.rect = self.rect.move(3, 0)
```

- Each sprite can have an `update` method that describes how to move that sprite on each timer tick.
  - Move a rectangle by calling its `move(`**dx**`, `**dy**`)` method.
  - Calling `update` on a `Group` updates all its sprites.

Sounds

- Loading and playing a sound file:

```
from pygame.mixer import *
mixer.init()            # initialize sound system
mixer.stop()            # silence all sounds

Sound("filename").play()   # play a sound
```

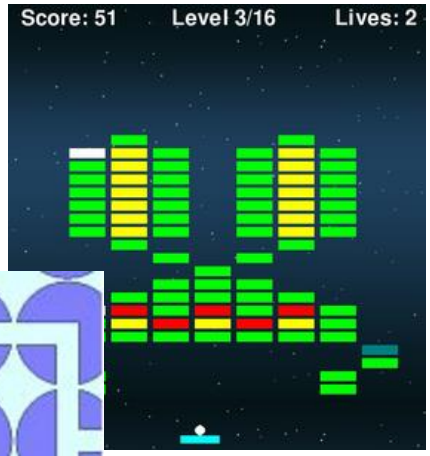
- Loading and playing a music file:

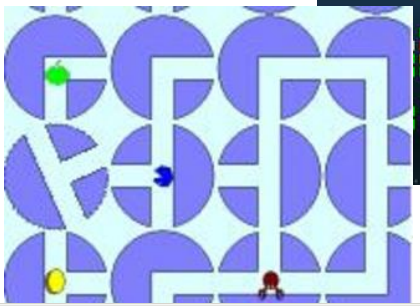
```
music.load("filename")     # load bg music file
music.play(loops=0)        # play/loop music
                         # (-1 loops == infinite)
```

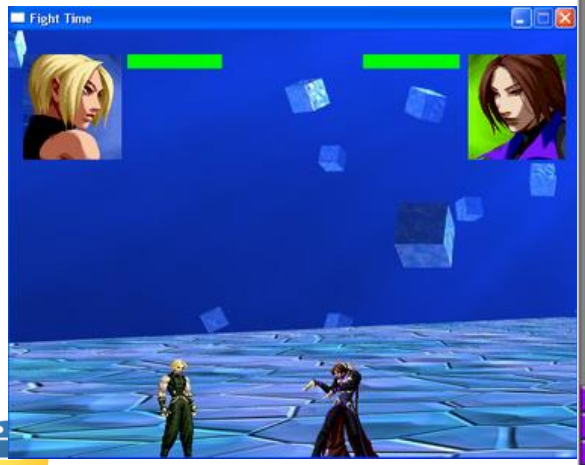
others: `stop`, `pause`, `unpause`, `rewind`, `fadeout`, `queue`

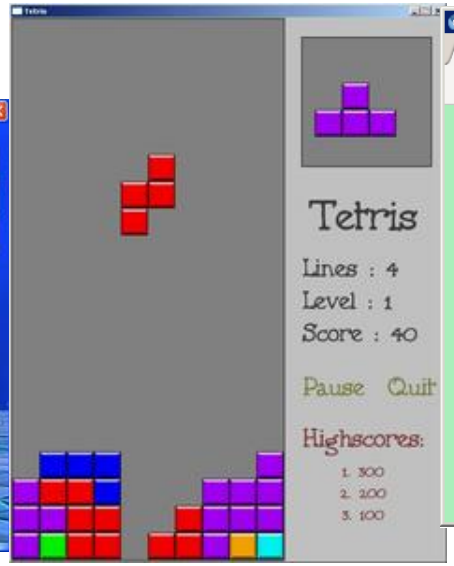
The sky's the limit!

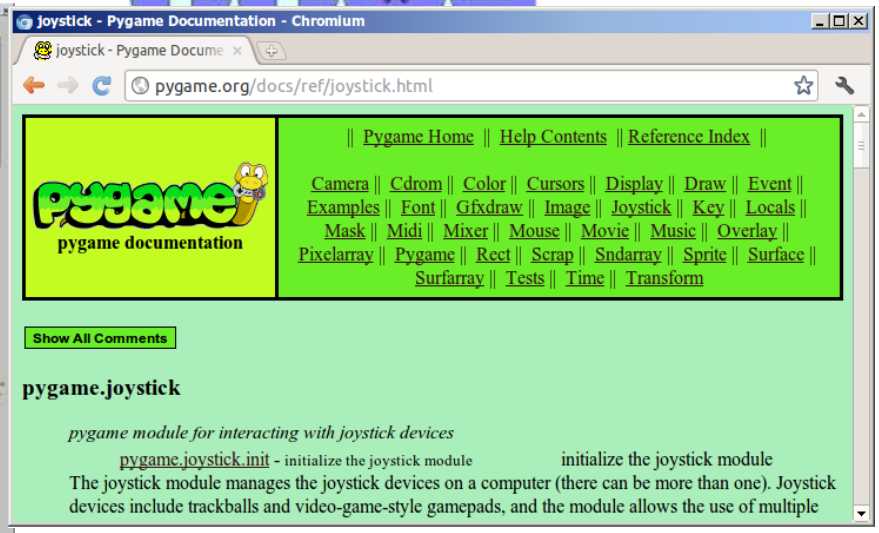
- pygame.org has lots of docs and examples
- can download tons of existing games
  - run them
  - look at their code for ideas
- if you can imagine it,
  you can create it!











Vẽ đồ thị, hiển thị dữ liệu 2D, 3D

https://matplotlib.org/stable/gallery/mplot3d/index.html

matplotlib

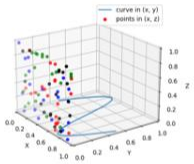
Plot types | Examples | Tutorials | Reference | User guide | Develop | Release notes

stable

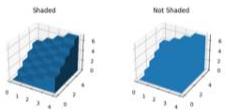
Section Navigation

- Lines, bars and markers
- Images, contours and fields
- Subplots, axes and figures
- Statistics
- Pie and polar charts
- Text, labels and annotations
- pyplot
- Color
- Shapes and collections
- Style sheets
- axes_grid1

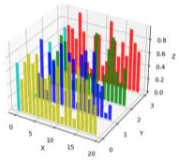
3D plotting



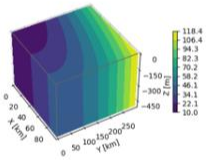
Plot 2D data on 3D plot



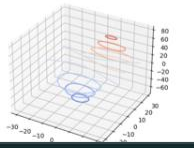
Demo of 3D bar charts

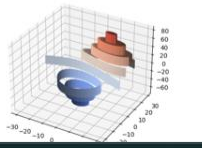


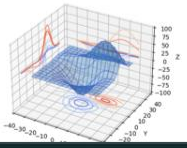
Create 2D bar graphs in different planes

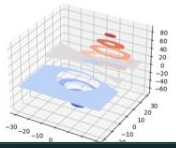


3D box surface plot









10°C Il pleut | Search | ENG FR | 8:29 AM 11/25/2022

https://matplotlib.org/stable/gallery/

Tkinter

Importing Tkinter modules

↓

Creating the main window for GUI app

↓

Adding widgets to the app

↓

Enter the Main event Loop

Q & A

Thank you!