

kmu80_COSC264_assignment1

Kupakwashe Mugutso
ID: 75935958

server.py

```
import socket
import select
import time
from datetime import date
from datetime import datetime

def main():
    try:
        xVal = int(input("Enter port number 1: "))
        yVal = int(input("Enter port number 2: "))
        zVal = int(input("Enter port number 3: "))

        if xVal < 1024 or xVal > 64000:
            print("ERROR: At least port 1 number not in range")
        elif yVal < 1024 or yVal > 64000:
            print("ERROR: At least port 2 number not in range")
        elif zVal < 1024 or zVal > 64000:
            print("ERROR: At least port 3 number not in range")
        elif xVal == yVal or xVal == zVal or yVal == zVal:
            print("ERROR: Port numbers must be unique")
        else:
            port1 = xVal
            port2 = yVal
            port3 = zVal

            memo = {port1: '1', port2: '2', port3: '3'}
            sock1 = socket.socket(socket.AF_INET,
                                  socket.SOCK_DGRAM)
            sock2 = socket.socket(socket.AF_INET,
                                  socket.SOCK_DGRAM)
            sock3 = socket.socket(socket.AF_INET,
                                  socket.SOCK_DGRAM)
            sock1.bind(("", port1))
            sock2.bind(("", port2))
            sock3.bind(("", port3))
            rlist = [sock1, sock2, sock3]
            wlist = []
            elist = []
            while True:
                rlist, wlist, elist = select.select(rlist, wlist, elist)
                for sock in rlist:
                    port_num = sock.getsockname()[1]
                    packet, address = sock.recvfrom(13)

                    packet = bytearray(packet)
                    receive(packet)
                    language = int(memo[port_num])
```

```

date_or_time = int(packet[5])
today = date.today()
current_date = today.strftime("%Y : %m : %d")
year = int(current_date[:4])
month = int(current_date[7:9])
day = int(current_date[12:])
now = datetime.now()
current_time = now.strftime("%H : %M")
hour = current_time[:2]
minute = current_time[5:]

text = text_representation(language, day, month, year, date_or_time, hour, minute)

hour = int(hour)
minute = int(minute)

text = text.encode('utf-8')
text_size = len(text)
if (text_size>255):
    print("Your text representation is more than 255")
else:
    packet_size = text_size + 13
    packet_size_bits = packet_size*8
    text = int.from_bytes(text, 'big')
    dt_response_packet = construct_response(language, packet_size_bits, year, month, day, hour, minute, text_size, text)
    text = text.to_bytes(text_size, 'big')

    sock.sendto(dt_response_packet, address)

except ValueError:
    return "Not number"

def receive(packet):
    """Will receive the dt-request packet from the client and check its correct"""
    if len(packet) != 6:
        print("ERROR: Packet is not 6 bytes of data as required")
        return
    elif packet[0] != 73 and packet[1] != 126:
        print("ERROR: Packet MagicNo field does not equal '0x497E'")
        return
    elif packet[2] != 0 and packet[3] != 1:
        print("ERROR: Packet type is incorrect as it doesn't equal '0x0001'")
        return
    elif packet[4] != 0 and (packet[5] != 1 or packet[5] != 2):
        print("ERROR: Packet Request type is incorrect as it doesn't equal '0x0001' or '0x0002'")
        return

def text_representation(language, day, month, year, date_or_time, hour, minute):
    """Will return the textual representation based on the language"""

```

```

text = ""
if language == 1:
    if date_or_time == 1:
        if month == 1:
            text = f"Today's date is January {day}, {year}"
        elif month == 2:
            text = f"Today's date is February {day}, {year}"
        elif month == 3:
            text = f"Today's date is March {day}, {year}"
        elif month == 4:
            text = f"Today's date is April {day}, {year}"
        elif month == 5:
            text = f"Today's date is May {day}, {year}"
        elif month == 6:
            text = f"Today's date is June {day}, {year}"
        elif month == 7:
            text = f"Today's date is July {day}, {year}"
        elif month == 8:
            text = f"Today's date is August {day}, {year}"
        elif month == 9:
            text = f"Today's date is September {day}, {year}"
        elif month == 10:
            text = f"Today's date is October {day}, {year}"
        elif month == 11:
            text = f"Today's date is November {day}, {year}"
        else:
            text = f"Today's date is December {day}, {year}"
    else:
        text = f"The current time is {hour}:{minute}"
elif language == 2:
    if date_or_time == 1:
        if month == 1:
            text = f"Ko te ra o tenei ra ko Kohitateā {day}, {year}"
        elif month == 2:
            text = f"Ko te ra o tenei ra ko Hui-tanguru {day}, {year}"
        elif month == 3:
            text = f"Ko te ra o tenei ra ko Poutu-te-rangi {day}, {year}"
        elif month == 4:
            text = f"Ko te ra o tenei ra ko Paenga-whawha {day}, {year}"
        elif month == 5:
            text = f"Ko te ra o tenei ra ko Haratua {day}, {year}"
        elif month == 6:
            text = f"Ko te ra o tenei ra ko Pipiri {day}, {year}"
        elif month == 7:
            text = f"Ko te ra o tenei ra ko Hongongoi {day}, {year}"
        elif month == 8:
            text = f"Ko te ra o tenei ra ko Here-turi-koka {day}, {year}"
        elif month == 9:
            text = f"Ko te ra o tenei ra ko Mahuru {day}, {year}"

```

```

        elif month == 10:
            text = f"Ko te ra o tenei ra ko Whiringa-a-nuku {day}, {year}"
        elif month == 11:
            text = f"Ko te ra o tenei ra ko Whiringa-a-rangi {day}, {year}"
        else:
            text = f"Ko te ra o tenei ra ko Hakihea {day}, {year}"
    else:
        text = f"Ko te wa o tenei wa {hour}:{minute}"
else:
    if date_or_time == 1:
        if month == 1:
            text = f"Heute ist der {day}. Januar {year}"
        elif month == 2:
            text = f"Heute ist der {day}. Februar {year}"
        elif month == 3:
            text = f"Heute ist der {day}. Marz {year}"
        elif month == 4:
            text = f"Heute ist der {day}. April {year}"
        elif month == 5:
            text = f"Heute ist der {day}. Mai {year}"
        elif month == 6:
            text = f"Heute ist der {day}. Juni {year}"
        elif month == 7:
            text = f"Heute ist der {day}. Juli {year}"
        elif month == 8:
            text = f"Heute ist der {day}. August {year}"
        elif month == 9:
            text = f"Heute ist der {day}. September {year}"
        elif month == 10:
            text = f"Heute ist der {day}. Oktober {year}"
        elif month == 11:
            text = f"Heute ist der {day}. November {year}"
        else:
            text = f"Heute ist der {day}. Dezember {year}"
    else:
        text = f"Die Uhrzeit ist {hour}:{minute}"
return text

def construct_response(language, packet_size_bits, year, month, day, hour, minute, text_size, text):
    """will construct the response packet"""
    if language == 1:
        language = 0x0001
    elif language == 2:
        language = 0x0002
    else:
        language = 0x0003
    dt_response = 0
    dt_response = (dt_response)|(0x497E << (packet_size_bits-16))
    dt_response = (dt_response)|(0x0002 << (packet_size_bits-32))

```

```
dt_response = (dt_response) | (language << (packet_size_bits-48))
dt_response = (dt_response) | (year << (packet_size_bits-64))
dt_response = (dt_response) | (month << (packet_size_bits-72))
dt_response = (dt_response) | (day << (packet_size_bits-80))
dt_response = (dt_response) | (hour << (packet_size_bits-88))
dt_response = (dt_response) | (minute << (packet_size_bits-96))
dt_response = (dt_response) | (text_size << (packet_size_bits-104))
dt_response = (dt_response) | (text)
```

```
dt_response = dt_response.to_bytes(packet_size_bits//8, 'big')
return dt_response
```

```
main()
```



```

        print(f"Language Code: {language_code}")
        print(f"Year: {year}")
        print(f"Month: {month}")
        print(f"Day: {day}")
        print(f"Hour: {hour}")
        print(f"Minute: {minute}")
        print(f"Length: {length}")
        print(f"Message: {text}")

except ValueError:
    print("ERROR: Port number is not an integer")
except gaierror:
    print("ERROR: hostname does not exist or IP address is not well formed")
except ConnectionResetError:
    print("ERROR: Port does not exist")

def createPacket(date_time):
    """Will create the dt-request packet"""
    packet = 0
    packet = (packet)|(0x497E << 32)
    packet = (packet)|(0x0001 << 16)
    if date_time == 'date':
        packet = (packet)|(0x0001)
    else:
        packet = (packet)|(0x0002)
    return packet

def receive(packet):
    """Will receive the dt-response packet from the client and check its correctness"""
    if len(packet) < 13:
        print("ERROR: Packet is not at least 13 bytes of data as required")
        return
    elif packet[0] != 73 and packet[1] != 126:
        print("ERROR: Packet MagicNo field does not equal '0x497E'")
        return
    elif packet[2] != 0 and packet[3] != 2:
        print("ERROR: Packet type is incorrect as it doesn't equal '0x0002'")
        return
    elif packet[4] != 0 and (packet[5] != 1 or packet[5] != 2 or packet[5] != 3):
        print("ERROR: Packet Language type is incorrect as it doesn't equal '0x0001' or '0x0002' or '0x0003'")
        return
    elif (packet[6]+packet[7]) > 2100 or (packet[6]+packet[7]) < 0:
        print("ERROR: Packet year is either larger than 2100 or negative")
        return
    elif packet[8] < 1 or packet[8] > 12:
        print("ERROR: Packet month is out of range (1 to 12)")
        return

```



```
elif packet[9] < 1 or packet[9] > 31:
    print("ERROR: Packet day is out of range (1 to 31)")
    return
elif packet[10] < 0 or packet[10] > 23:
    print("ERROR: Packet hour is out of range (0 to 23)")
    return
elif packet[11] < 0 or packet[11] > 59:
    print("ERROR: Packet minute is out of range (0 to 59)")
    return
elif len(packet) != (13 + packet[12]):
    print("ERROR: Packet length inconsistent")
    return
```

```
main()
```

Plagiarism Declaration

This form needs to accompany your COSC264 assignment submission.

I understand that plagiarism means taking someone else's work (text, program code, ideas, concepts) and presenting them as my own, without proper attribution. Taking someone else's work can include verbatim copying of text, figures/images, or program code, or it can refer to the extensive use of someone else's original ideas, algorithms or concepts.

I hereby declare that:

- My assignment is my own original work. I have not reproduced or modified code, figures/images, or writings of others without proper attribution. I have not used original ideas and concepts of others and presented them as my own.
- I have not allowed others to copy or modify my own code, figures/images, or writings. I have not allowed others to use original ideas and concepts of mine and present them as their own.
- I accept that plagiarism can lead to consequences, which can include partial or total loss of marks, no grade being awarded and other serious consequences, including notification of the University Proctor.

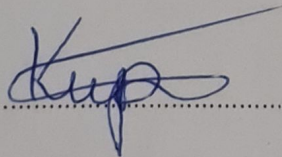
Name:

KUPAKWASHE MUGUTSO

Student ID:

75935958

Signature:



Date:

24/08/22