client.py

```
##################################
# COSC264 Sockets Assignment 2018 - client.py
# Author: Ambrose Ledbrook
# ID: 79172462
##################################


##################################
# Usage:
# python3 client.py request host port
##################################


# Importing used modules
import socket as soc
import sys
from select import select


# Defining constants
MAGIC_NUMBER = 0x497E
REQUEST_PACKET = 0x0001
RESPONSE_PACKET = 0x0002
DATE_REQUEST = 0x0001
TIME_REQUEST = 0x0002
ENGLISH_CODE = 0x0001
MAORI_CODE = 0x0002
GERMAN_CODE = 0x0003


def validate_packet(pkt):
    """
    Checking that the received response packet is valid
    :param pkt: The received response packet
    """
    text = None
    # Checking all fields of the response packet
    if len(pkt) < 13: text = "Invalid packet header length"
    elif ((pkt[0] << 8) + pkt[1]) != MAGIC_NUMBER: text = "Invalid magic number"
    elif ((pkt[2] << 8) + pkt[3]) != RESPONSE_PACKET: text = "Invalid packet type"
    elif ((pkt[4] << 8) + pkt[5]) not in [ENGLISH_CODE, MAORI_CODE, GERMAN_CODE]: text = "Invalid language code"
    elif ((pkt[6] << 8) + pkt[7]) >= 2100 or ((pkt[6] << 8) + pkt[7]) <= 0: text = "Invalid year"
    elif pkt[8] < 1 or pkt[8] > 12: text = "Invalid month"
    elif pkt[9] < 1 or pkt[9] > 31: text = "Invalid day"
    elif pkt[10] < 0 or pkt[10] > 23: text = "Invalid hour"
    elif pkt[11] < 0 or pkt[11] > 59: text = "Invalid minute"
    elif len(pkt) != (13 + pkt[12]): text = "Invalid packet length"
    if text:
        # Outputting an error message as an error was found in the packet
        print("******************************")
        print("{0}, program will terminate".format(text))
        print("******************************")
        sys.exit()


def handle_packet(pkt):
    """
    Handles the response packet once it has been received by first validating
    and then printing the contents
    :param pkt: The received response packet
    """
    # Checking the packet is valid
    validate_packet(pkt)
    # Printing the information from the packet
    if ((pkt[4] << 8) + pkt[5]) == ENGLISH_CODE:
        lang = "English"
    elif ((pkt[4] << 8) + pkt[5]) == MAORI_CODE:
        lang = "Maori"
    else:
        lang = "German"
    # Printing the contents of the response packet
    print("Response from server:")
    print("-----")
    print("Magic number: {0}\nPacket type: {1}".format(hex((pkt[0] << 8) + pkt[1]), RESPONSE_PACKET))
    print("Language code: {0}".format(((pkt[4] << 8) + pkt[5])))
    print("Year: {0}\nMonth: {1}\nDay: {2}".format(((pkt[6] << 8) + pkt[7]), pkt[8], pkt[9]))
    print("Hour: {0}\nMinute: {1}".format(pkt[10], pkt[11]))
    print("Length field: {0}".format(pkt[12]))
    print("-----")
    # Getting time and date to output
    text = pkt[13:]
    date = "{0}:{1}:{2}".format(((pkt[6] << 8) + pkt[7]), pkt[8], pkt[9])
```

```
    if pkt[11] < 10:
        time = "{0}:0{1}".format(pkt[10], pkt[11])
    else:
        time = "{0}:{1}".format(pkt[10], pkt[11])
    print("The date is: {0}".format(date))
    print("The time is: {0}".format(time))
    print("Textual representation in {0}: {1}".format(lang, text.decode()))
    # Exiting the program
    print("----------------------------")
    print("----------------------------")
    print("Program will now exit")
    print("----------------------------")
    print("----------------------------")
    sys.exit()


def process_inputs(args):
    """
    Processing the command line arguments
    :param args: The command line arguments
    :return: the request type, host address and port number
    """
    text = None
    # Getting inputs from command line arguments
    request_type = args[0]
    host = args[1]
    port = args[2]
    # Checking if there is the correct number of arguments passed
    if len(args) != 3:
        text = "Invalid number of inputs"
    else:
        # Checking if the request field is correct
        if request_type != "date" and request_type != "time":
            text = "Invalid request type, request must be either 'date' or 'time'"
        else:
            if request_type == "date":
                request = DATE_REQUEST
            else:
                request = TIME_REQUEST
        # Checking if the port field is correct
        try:
            port = int(port)
            if port < 1024 or port > 64000:
                text = "Invalid port, port must be in range 1024 to 64000"
        except ValueError:
            text = "Invalid port type, port must be an integer"
        # Checking if the host field is correct
        try:
            host = soc.gethostbyname(host)
        except soc.gaierror:
            text = "Invalid hostname or IP address"
    if text:
        # Outputting an error message as the arguments are invalid
        print("******************************")
        print(text)
        print("******************************")
        # Outputting usage instructions
        print("Usage: python3 client.py request host port")
        print("Program will now exit")
        sys.exit()
    else:
        # Retuning the processed arguments
        return request, port, host


def wait(socket, pkt, server):
    """
    Sending the request packet to the server and then waiting
    one second for a response
    :param socket: The clients socket
    :param pkt: The packet to be sent
    :param server: The address of the server to send the packet to
    """
    # Sending request packet to the server
    socket.sendto(pkt, server)
    print("----------------------------")
    print("----------------------------")
    print("Request packet sent to {0}: ".format(server))
    print(pkt)
    print("----------------------------")
    # Waiting for 1 second for response from the server
    reads, writes, exceps = select([socket], [], [], 1.0)
    if reads == writes == exceps == []:
        # Server did not respond fast enough
        print("******************************")
        print("Response too slow, program will terminate")
```

```
        print("*****************************")
        # Closing the socket
        socket.close()
        sys.exit()
    elif len(reads) != 0:
        # Receiving response from the server
        pkt, address = socket.recvfrom(1024)
        # Packet has been received from the server
        print("Response packet received from {0}: ".format(address))
        print(pkt)
        print("-----------------------------")
        print("-----------------------------")
        # Closing the socket
        socket.close()
        handle_packet(pkt)


def main():
    """
    Running the client program
    """
    # Getting the inputs passed from the user
    args = sys.argv[1:]
    request, port, host = process_inputs(args)
    server = (host, port)
    # Opening socket to communicate with the server
    socket = soc.socket(soc.AF_INET, soc.SOCK_DGRAM)
    # Creating a request packet
    request_packet = bytearray(6)
    request_packet[0:2] = MAGIC_NUMBER.to_bytes(2, "big", signed=False)
    request_packet[2:4] = REQUEST_PACKET.to_bytes(2, "big", signed=False)
    request_packet[4:6] = request.to_bytes(2, "big", signed=False)
    # Passing the pkt to be sent and then waiting for 1 second for a response from the server
    wait(socket, request_packet, server)


if __name__ == "__main__":
    main()


##################################
# End of client.py file
##################################
```