```python
# Packet Structure (response.py)
# Bach Vu
# 01/08/2020

from packet import *
from datetime import datetime
from language import DT_Language

class DT_Response(DT_Packet):
    ErrorMessage = [
        "Expecting received packet have MagicNum 0x497E.",
        "Expecting received packet have packetType 0x0002.",
        "Undefined language outupt Type [Eng/Maori/Ger]?",
        "Year is over 2100. Data received must be invalid.",
        "Month is not between 1 and 12. Data received must be invalid.",
        "Day is not between 1 and 31. Data received must be invalid.",
        "Hour is not between 0 and 23. Data received must be invalid.",
        "Minute is not between 0 and 59. Data received must be invalid.",
        "Some data of displaying message is missing",
        "Packet header is shorter than expected"
    ]

    def __init__(self, language, mode, head_info=None):
        self.language = language

        if head_info is None:
            super().__init__(0x0002)
            now = datetime.now() # Time when obj created
            self.time = [now.year, now.month, now.day, now.hour, now.minute]
            dt = DT_Language(language, mode, self.time)
            self.message = dt.DTtoString().encode('utf8')
            self.m_len = len(self.message)
        else:
            self.MagicNum   = head_info[0]
            self.packetType = head_info[1]
            self.time       = head_info[2]
            self.message    = head_info[3]
            self.m_len      = head_info[4]

    def __repr__(self):
        out = "{}\n<Magic: {}> <packetType: {}> <lang: {}>\n<Time: {}> <MessLen: {}>"
        mess = type(self).__name__ + ": " + str(self.message, 'utf-8')
        return out.format(mess, hex(self.MagicNum),
                DT_Packet.DT_hex(self.packetType),
                DT_Packet.DT_hex(self.language),
                self.time, self.m_len)

    def header_errorCode(self):
        error_code = 0
        if self.MagicNum != 0x497E:
            error_code = 1
        elif self.packetType != 0x0002:
            error_code = 2
        elif self.language < 0x0001 or self.language > 0x0003:
            error_code = 3
        elif self.time[0] < 0 or self.time[0] > 2100:
            error_code = 4
        elif self.time[1] < 1 or self.time[1] > 12:
            error_code = 5
        elif self.time[2] < 1 or self.time[2] > 31:
            error_code = 6
        elif self.time[3] < 0 or self.time[3] > 23:
            error_code = 7
        elif self.time[4] < 0 or self.time[4] > 59:
            error_code = 8
        elif self.m_len != len(self.message):
            error_code = 9
        return error_code

    def encodePacket(self):
        """ Get the actual bytearray store data of this packet """
        # Error check
        check = self.isValid()
        if check != 0:
            return check

        # Header
        header = ""
        header += DT_Packet.intToBinStr(self.MagicNum,16)
        header += DT_Packet.intToBinStr(self.packetType,16)
        header += DT_Packet.intToBinStr(self.language,16)
        header += DT_Packet.intToBinStr(self.time[0],16)
        header += DT_Packet.intToBinStr(self.time[1],8)
        header += DT_Packet.intToBinStr(self.time[2],8)
        header += DT_Packet.intToBinStr(self.time[3],8)
        header += DT_Packet.intToBinStr(self.time[4],8)
        header += DT_Packet.intToBinStr(self.m_len,8)
        header  = int(header, 2).to_bytes(13, byteorder='big')

        # Pack
        packet = bytearray()
        packet += header
        packet += self.message
        return packet

    @staticmethod
    def decodePacket(packet, mode):
        if len(packet) < 13:
            return 10

        """ Turn bytearray to object """
        magic    = DT_Packet.byteArrToInt(packet[0:2])
        packType = DT_Packet.byteArrToInt(packet[2:4])
        language = DT_Packet.byteArrToInt(packet[4:6])
        year     = DT_Packet.byteArrToInt(packet[6:8])
        month    = DT_Packet.byteArrToInt(packet[8:9])
        day      = DT_Packet.byteArrToInt(packet[9:10])
        hour     = DT_Packet.byteArrToInt(packet[10:11])
        minute   = DT_Packet.byteArrToInt(packet[11:12])
        length   = DT_Packet.byteArrToInt(packet[12:13])

        time = [year, month, day, hour, minute]
        mess = packet[13:]

        # Error check
        param = [magic, packType, time, mess, length]
        responsePack = DT_Response(language, mode, tuple(param))
        check = responsePack.isValid()
        if check != 0:
            return check
        return responsePack
```