```
"""
CP1404/CP5632 - Practical - Suggested Solution
ASCII table and converter
"""
MAX_COLUMNS = 10
MIN_COLUMNS = 2

LOWER = 33
UPPER = 127

char = input("Enter a character: ")
print("The ASCII code for {} is {}".format(char, ord(char)))
number = int(input("Enter a number between {} and {}: ".format(LOWER, UPPER)))
while number < LOWER or number > UPPER:
    number = int(
        input("Enter a number between {} and {}: ".format(LOWER, UPPER)))
print("The character for {} is {}".format(number, chr(number)))

# ASCII table (single column)
for value in range(LOWER, UPPER + 1):
    print("{:3} {:>4}".format(value, chr(value)))

# ASCII tables with columns (two versions)
columns = int(input("Enter number of columns: "))
while columns < MIN_COLUMNS or columns > MAX_COLUMNS:
    print("Please use a value between {} and {}".format(MIN_COLUMNS, MAX_COLUMNS))
    columns = int(input("Enter number of columns: "))
# calculate the range of values and the number of full rows
number_of_values = UPPER - LOWER + 1
rows = number_of_values // columns

print("Version 1: Horizontal then vertical ordering")
# iterate through the full rows first, incrementing by 1
value = LOWER
for row in range(rows):
    for column in range(columns):
        print("{:6} {:>2}".format(value, chr(value)), end="")
        value += 1
    print()

# last row is special as it may not have all columns so handle separately
# start where we left off and only print up to UPPER
starting_value = value
for value in range(starting_value, UPPER + 1):
    print("{:6} {:>2}".format(value, chr(value)), end="")
print("\n")

print("Version 2: Vertical then horizontal ordering")
# iterate through rows
for row in range(rows + 1):
    starting_value = LOWER + row
    value = starting_value
    # print all column values not including the last one (-1)
    for column in range(columns - 1):
        value_to_print = value + (column * rows)
        print("{:6} {:>2}".format(value_to_print, chr(value_to_print)), end="")
        value += 1
```

```
# last column may not exist so handle separately
# having the if statement outside the for loop means we don't do it every column
# so it is more efficient (we can't avoid doing it every row AFAIK)
value_to_print = value + ((column + 1) * rows)
if value_to_print <= UPPER:
    print("{:6} {:>2}".format(value_to_print, chr(value_to_print)), end="")
print()
```