Understanding ASCII Values For Characters Using Python

ASCII Values For Capital Letters

A = 65
B = 66
C = 67
D = 68
E = 69
F = 70
G = 71
H = 72
I = 73
J = 74
K = 75
L = 76
M = 77
N = 78
O = 79
P = 80
Q = 81
R = 82
S = 83
T = 84
U = 85
V = 86
W = 87
X = 88
Y = 89
Z = 90

ASCII Values For Small Letters

a = 97
b = 98
c = 99
d = 100
e = 101
f = 102
g = 103
h = 104
i = 105
j = 106
k = 107
l = 108
m = 109
n = 110
o = 111
p = 112
q = 113
r = 114
s = 115
t = 116
u = 117
v = 118
w = 119
x = 120
y = 121
z = 122

ASCII

```
>>> ord('A')
65
>>> ord('Z')
90
>>> ord('a')
97
>>> ord('z')
122
Captital A-Z: 65 to 90

>>> l = ['r', 'u', 'd', 'r', 'a', 'n', 's', 'h']
>>> max(l)
'u'
>>> ord('u')
117
>>> min(l)
'a'
>>> ord('a')
97

>>> bin(65)
'0b1000001'
>>> bin(66)
'0b1000010'
>>>
```

Bit #	7	6	5	4	3	2	1	0
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0

One usecase of ASCII:
For storing in a memory of an 8 bit computer, we need to convert Alphabet to Decimal then Decimal to Binary.

A -> 65 -> 01000001
B -> 66 -> 01000010

ord() and chr()

Here, ord() and chr() are built-ins.

Ord(): gives you ASCII for the alphabet

Chr(): gives you the alphabet for the ASCII

```
>>> ord('A')
65

>>> chr(65)
'A'
```

Unicode

Unicode, formally The Unicode Standard, is an information technology standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.

Character sets for different bases:

Binary	Base 2	0, 1
Decimal	Base 10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	Base 16	0-9 then A-F