# PYTHON – STRINGS

# Common string quotes and literals

```
>>> S = ''                      #  Empty string
>>> S = "spam's"                # Same as single quote
>>> S = 's\np\ta\x00m'          # Escape sequences
>>> S = """..."""               # Triple-quoted block strings
>>> S = r'\temp\spam'           # Raw strings
>>> S = b'spam'                 # Byte strings in 3.0
>>> S1 + S2                     # Concatenate
>>> S * 3                       # Re-iterate
```

# Common string quotes and literals

- S[i] # indexing
- S[i:j] # slicing
- len(S) # function
- S.find('pa') # method
- S.rstrip() # method
- S.replace('pa', 'xx') # method

# Common string quotes and literals

- S.split(',') # split on delimiter
- S.isdigit() # content test
- S.lower() # case conversion
- S.endswith('spam') # end test
- 'spam'.join(strlist) # delimiter join

# Common string quotes and literals

```
>>> for x in S: print(x)
>>> 'spam' in S
>>> [c * 2 for c in S]          # iteration
>>> map(ord, S)                 # membership
```

# Indexing Tricks

```
>>> S[0]      # fetches the first item.
>>> S[−2]     # same as S[len(S)−2]
>>> S[i:j]    # contiguous sequences
>>> S[1:3]    # does what?
>>> S[1:]     # ???
```

# `ord()`

Using the `ord()` function to convert a binary string to a decimal number.

Explain how this works:

```
>>> B = '1101'                          # Convert binary digits to integer with ord
>>> I = 0
>>> while B != '':
...     I = I * 2 + (ord(B[0]) - ord('0'))
...     B = B[1:]
...
>>> I
13
```

# More examples

```
>>> print('------- ...more... ---')
>>> print('-' * 80)


>>> myjob = "hacker"
>>> for c in myjob: print(c, end=' ')
...

>>> S = 'spam'
>>> S[0], S[-2]
('s', 'a')
>>> S[1:3], S[1:], S[:-1]
('pa', 'pam', 'spa')
```

# String Backslash Characters

- \\ # delimits \
- \' # delimits '
- \" # delimits "
- \b # Backspace
- \f # Formfeed

# String Backslash Characters

- \n # New Line
- \r # Return Carriage
- \t # Horizontal tab
- \v # Vertical tab
- \x*hh* *# Char hex val hh*
- *\000* *# Char with octal val ooo*
- *\other* # Not an escape (keeps both \ and other)

# Raw Strings Suppress Escapes!

Escape characters are automatically read:

```
myfile = open('C:\new\text.dat', 'w‘)
```

Same as:

```
myfile = open(‘C:(newline)ew(tab)ext.dat, ‘w’)
```

So what do we do???

# Raw Strings Suppress Escapes!

```
>>> myfile = open(r'C:\new\text.dat', 'w')
```

```
>>> myfile = open('C:\\new\\text.dat', 'w')
```

# Exercise

Write a program which prompts the user to input two IP addresses and a mask. Determine if the two addresses are on the same subnet and print yes or no accordingly.

Hints:

1. First line is always.. #!/usr/bin/python3
2. Use the “input” function to prompt the user for the info
3. Use the “split” method and “int()” to convert the input strings into 4 element integer lists
4. Use a for loop (for I in range(4): ) to logically “and” each address octet with the respective mask octet and compare them one octet at a time…

```
ip1=input('Enter an ip address  ')
ip2=input('Enter a second ip address  ')

m=input('Enter a mask value   ')

ip1=ip1.split('.')
ip2=ip2.split('.')
m=m.split('.')

for i in range(4):
    ip1[i]=int(ip1[i])
    ip2[i]=int(ip2[i])
    m[i]= int(m[i])
    if (ip1[i] & m[i] != ip2[i] & m[i]):
        print("not the same subnet on the ",i+1,"
 octet")
```

# Changing String

```
>>> S = 'spam'
>>> S[0] = 'x'                          # Raises an error!
TypeError: 'str' object does not support item assignment


>>> S = S + 'SPAM!'
>>> S
'spamSPAM!'
>>> S = S[:4] + 'Burger' + S[-1]
>>> S
'spamBurger!'
```

# Changing String

`replace()`:

```
>>> S = 'splot'
>>> S = S.replace('pl', 'pamal')
>>> S
'spamalot'
```

`format()`:

```
>>> 'That is %d %s bird!' % (1, 'dead')
That is 1 dead bird!
>>> 'That is {0} {1} bird!'.format(1, 'dead')
'That is 1 dead bird!'
```

# Changing String

`replace()`:

```
>>> 'aa$bb$cc$dd'.replace('$', 'SPAM')
'aaSPAMbbSPAMccSPAMdd'
```

`find()`:

```
>>> S = 'xxxxSPAMxxxxSPAMxxxx'
>>> where = S.find('SPAM')              # Search for position
>>> where                               # Occurs at offset 4
4
>>> S = S[:where] + 'EGGS' + S[(where+4):]
>>> S
'xxxxEGGSxxxxSPAMxxxx'
```

# Changing String

`replace()`:

```
>>> S = 'xxxxSPAMxxxxSPAMxxxx'
>>> S.replace('SPAM', 'EGGS')          # Replace all
'xxxxEGGSxxxxEGGSxxxx'

>>> S.replace('SPAM', 'EGGS', 1)       # Replace one
'xxxxEGGSxxxxSPAMxxxx'
```

# Changing String

Convert to list:

```
>>> S = 'spammy'
>>> L = list(S)
>>> L
['s', 'p', 'a', 'm', 'm', 'y']

>>> L[3] = 'x'
>>> L[4] = 'x'

>>> L
['s', 'p', 'a', 'x', 'x', 'y']
```

# Changing String

`join()`:

```
>>> S = ''.join(L)
>>> S
'spaxxy'

>>> 'SPAM'.join(['eggs', 'sausage', 'ham', 'toast'])
'eggsSPAMsausageSPAMhamSPAMtoast'
```

# Parsing String

```
>>> line = 'aaa bbb ccc'
>>> col1 = line[0:3]
>>> col3 = line[8:]
>>> col1
'aaa'
>>> col3
'ccc'


>>> line = 'aaa bbb  ccc'
>>> cols = line.split()
>>> cols
['aaa', 'bbb', 'ccc']
```

# String Methods

- S.endswith(suffix [, start [, end]])
- S.replace(old, new [, count])
- S.find(sub [, start [, end]])
- S.rfind(sub [,start [,end]])

# String Methods

- S.isalnum()
- S.isalpha()

- S.rsplit([sep[, maxsplit]])
- S.isdecimal()

- S.rstrip([chars])
- S.isdigit()

- S.split([sep [,maxsplit]])
- S.splitlines([keepends])

- S.startswith(prefix [, start [, end]])
- S.isnumeric()
- S.strip([chars])

# Operations In General

- Operations work the same for all the types in the same category, so we'll only need to define most of these ideas once.

- There are three major type (and operation) categories in Python:

1. Numbers (integer, floating-point, decimal, fraction, others) support addition, multiplication, etc.

2. Sequences (strings, lists, tuples) support indexing, slicing, concatenation, etc.

3. Mappings (dictionaries) support indexing by key, etc.