

Introduction to Python

Feature Summary, Session 3

FILES

- **open()** function: **construct a file object**
file object has methods for reading or writing the file
argument: string filename or file path,
and optional "mode" argument ('r', 'w', 'a')
return value: file object

```
fh = open('myfile.txt')  
fh = open('myfile.txt', 'w')  
fh = open('myfile.txt', 'a')  
str -> open file -> file object
```
- file **.read()** method: **read a file as a single string**
the entire file text is returned in one string
argument: N/A (method works on file object itself)
return value: string

```
text = fh.read()  
file read file -> str
```
- file **.readlines()** method: **read a file as a list of strings**
*the entire file text is divided into lines (by newline)
and returned as a list of strings, each string a line*
argument: N/A (method works on file object itself)
return value: list of strings

```
lines = fh.readlines()  
file read file -> list of strings
```
- **for** looping on a file object: **iterate over each string line in a file**
*as Python iterates over each line in the file, it assigns the line to the
loop variable (at right, **myline**) as a string, then executes the block.
The assignment and block execution happen once for each line in the file*
argument: N/A (iterates over file object itself)
return value: with each iteration, a string line from the file assigned to loop variable

```
for myline in fh:  
    print(myline)  
for str in file:  
    print str
```

STRINGS

- string **.split()** method: **divide a string into a list of strings**
*divides a string into a list of strings by splitting on the
delimiter argument*
argument: string delimiter (**None** or no delimiter to
split on whitespace)
return value: a list of strings from the string divided
on the delimiter (with delimiter removed)

```
line = 'this:that:other'  
items = line.split(':')  
# ['this', 'that', 'other']  
str -> str split -> list of str  
aline = 'this that other'  
items = line.split() #=or=  
str split -> list of str  
items = line.split(None)  
# ['this', 'that', 'other']  
None -> str split -> list of str
```
- **.join()** method: **join a list of strings into a string**
*method is called on a string delimiter and returns a string
consisting of the items in the list, joined with the delimiter*
argument: a list of strings
return value: string

```
items = ['a', 'hi', 'd']  
line = ':'.join(items)  
print(line) # 'a:hi:d'  
list of str -> str join -> str
```

- **slice** a string: ***slice a string by index (character position)***

extract a portion of a string by index position

subscript: the integer position of the start and end of string (end index is *non-inclusive*)

return value: string

```
mystr = '20180917'
month = mystr[4:6]      # 09
str slice -> str
```

- **.rstrip()** method: ***remove "whitespace" characters from right side of string***

returns a string with all spaces, tabs and/or newlines removed from the original string

argument: N/A (method works on string object itself)

return value: string

```
li = 'line. \n'
sli = li.rstrip()      #'line.'
str strip newline from end -> str
```

- **.splitlines()** method: ***split a "multiline" string on newline***

split a string containing newlines into a list of string lines, with the newlines removed

argument: N/A (method works on string object itself)

return value: list of strings

```
text = 'line1\nline2\nline3'
sp = text.splitlines()
      # ['line1','line2','line3']
str split into lines -> list of str
```

LISTS

- **subscript**: ***select an item from a list***

index: int position of item counting from 0

return value: whatever object is referenced in the selected item of the list

```
items = ['th', 'tha', 'oth']
second = items[1]      #'tha'
list of strings subscript -> str
```

- **len()** function: ***return integer length of a list***

argument: list

return value: integer

```
items = ['th', 'tha', 'oth']
print(len(items))      # 3
list -> get length -> int
```

COMMAND LINE ARGUMENTS

- **sys.argv** list of strings: ***access command line arguments***

This list of strings "springs into being" at program

start. It contains the program name in the first

item, and any text args entered at the command line when the program was executed.

```
import sys
args = sys.argv
print(args)
      # ['prog.py', 'arg1']
read commandline args->list of str
```