Python Cheat Sheet

List Methods

Method	Purpose	
l.append(x)	Adds x to the end of the list.	
l.extend(iterable)	Adds all items from iterable to the list.	
l.insert(i, x)	Inserts x at index i.	
l.index(x, start, end)	Finds the index of x between start and end.	
l.remove(x)	Removes the first occurrence of x.	
l.sort(key, reverse)	Sorts the list in place. key is a function to	
	transform items before sorting.	
l.pop(i)	Removes and returns the item at index i.	
l.reverse()	Reverses the list in place.	

List Comprehensions

Syntax	Description
[expression for x in iterable]	Creates a list from an iterable.
[x for x in iterable if condition]	Adds a filter condition.

Functional List Operations

Function	Purpose	
filter(func, list)	Filters elements using func.	
map(func, list)	Applies func to each element.	

Strings

Method	Purpose
s.split(sep)	Splits string into a list using sep. Defaults to whitespace.
s.join(iterable)	Joins elements of iterable with s as a separator.
s.strip(chars)	Removes chars from both ends of the string.
s.lower()	Converts string to lowercase.
s.upper()	Converts string to uppercase.

File Handling

• Open: open(filename, mode)

o Modes: r (read), w (write), a (append).

• Read: file.read(), file.readlines(), file.readline()

Write: file.write(data)

• Close: file.close()

• Use with open() for automatic cleanup.

FILE *fopen (const char *pszFilename, const char *szMode)

Mode	Description	Reading binary data
R	Read (text mode)	fread(&struct_var, sizeof(struct_var), 1, file_ptr);
W	Write (overwrite, text mode)	
a	Append (text mode)	Writing binary data
rb	Read (binary mode)	fwrite(&struct_var, sizeof(struct_var), 1, file_ptr);
wb	Write (overwrite, binary mode)	
ab	Append (binary mode)	Direct Positioning in a File
r+	Read/Write (text mode)	int fseek (FILE *pFile, long lRelativeByteAddress
rb+	Read/Write (binary mode)	, int iSeekMode)
wb+	Read/Write (binary mode), create file if needed	
ab+	Read/append (binary mode), create file if needed	

Low level I/O

Mode	Description	<pre>int read(int fd, void *psbBuf, long lCount)</pre>
open	Opens a file and returns a file descriptor	Beginning with the current file position, read
read	Reads data from a file	reads 1Count bytes into the buffer. The data
write	Writes data to a file	read can be binary.
leek	Sets the position in a file	int write(int fd, void *psbBuf, long lCount)
stat	Gets file metadata	Beginning with the current file position, write
opendir	Opens a directory	writes 1Count bytes from the buffer to the file.
readdir	Reads directory entries	

Inode

- size of the file in bytes	Number of data blocks needed (N) = file size / block size
- device ID of the device containing the file	
- user ID of the owner	Number of entries per block (E) = block size / length of address
- group ID	
- file mode	Blocks of direct pointers needed (D) = (N – 12) / E
- timestamps for when the inode was last	(round up to integer value)
changed, content last modified, and last	
access	Blocks of indirect pointers needed (I) = $(D - 1) / E$
- link count	(round up to integer value)
- 12 direct pointers to data blocks	
- one indirect pointer	Total blocks needed = N + D + I
 one double indirect pointer 	(not counting inode)
- one triple indirect pointer	

 - pid_t fork(): Creates a child process. Returns: - 0: To the child process. - Child PID: To the parent process. 1: If failed. 	- getpid(): Get current process ID.- getppid(): Get parent process ID.- waitpid(): Wait for a child process to finish.(avoids creating orphans)
 - Pipe: one-way communication between parent and child. - Example: pipe(fd) creates fd[0] for reading and fd[1] for writing. 	Named Pipes (FIFOs): Persistent after processes terminate, allowing unrelated processes to communicate Example: mkfifo("mypipe", permissions); open("mypipe", mode)`.
Shared Memory: - Fastest IPC method Requires synchronization (e.g., locks) Steps: 1. ftok(filename, proj_id): Generate key. 2. shmget(key, size, flags): Create/get shared memory. 3. shmat(shmid, NULL, 0): Attach shared memory. 4. Access and modify shared memory. 5. shmdt(ptr): Detach memory. 6. shmctl(shmid, IPC_RMID, NULL): Deallocate memory.	Process States - **Ready**: Process is waiting to be assigned to CPU **Running**: Process is executing **Waiting**: Waiting for resources (I/O, etc.) **Terminated**: Completed execution.

Attempt	Conditions	Result
read	Empty pipe, writer attached	Read blocked
write	Full pipe, reader attached	Write blocked
read	Empty pipe, no writer attached	EOF returned
write	No reader	SIGPIPE