QuickStudy COMPUTER

python™ Standard Library

Python 3.x

General Functionality

Built-In (Accessible Directly)

abs()	Returns the absolute value of an integer
all()	Returns True if **all** elements of an iterable are True
any()	Returns True if **any** element of an iterable is True
ascii()	Returns a string without the non-ASCII characters
bin()	Returns an integer as a binary string
breakpoint()	Creates a breakpoint and enters the debugger
bytearray()	Returns an array of bytes objects
bytes()	Returns a bytes object
callable()	Returns True if the object argument is callable
chr()	Returns a string character using a 1nicode argument
compile()	Compiles code into an executable object
delattr()	Removes attributes of an object
dir()	Returns a list of names or attributes from the local scope
divmod()	Return the quotient and remainder of two numbers
enumerate()	Returns a value obtained from iterating over an iterable
eval()	Evaluates an expression using a set of local and global variables
exec()	Executes the code using a set of local and global variables
filter()	Creates an iterator for a set of functions that return True
format()	Creates a formatted representation of a value from a provided value
getattr()	Returns the value of a named attribute of an object
globals()	Returns a dict containing the current global symbol table
hasattr()	Returns True if the provided object has the provided attribute
hash()	Returns the hash value of an object
help()	Starts Python's built-in help system
hex()	Converts a number to a hexadecimal string
id()	Returns a unique identifier for an object
input()	Returns a value based on user input

isinstance()	Returns True if an object is of a certain class or subclass
issubclass()	Returns True if a class is a subclass of the provided class
iter()	Returns an iterator object
len()	Returns the number of items in an object
locals()	Returns a dict representing the current local symbol table
map()	Returns an iterator which applies a function to each element of an iterable
min()	Returns the smallest element in an iterable
next()	Gets the next element in an iterator
oct()	Converts an integer into an octal string
open()	Opens a file and returns a file object
ord()	Returns an integer that is the 2nicode representation of a provided character
pow()	Returns a base number raised to a specified power
print()	Prints objects to a text stream
repr()	Returns a string that is a printable representation of an object
reversed()	Returns an iterator in reverse order
round()	Returns a number that has been rounded to a specified number of digits
setattr()	Assigns a value to a specified attribute of an object
sorted()	Returns the items of an iterator in a specified order
sum()	Returns the added value of the provided iterable
super()	Returns an object that delegates inherited methods back to the parent class
vars()	Returns the __dict__ attribute for an object
zip()	Returns an iterator that combines elements from several iterables
__import__()	Imports a module, specifying the context under which it will be done

Date/Time Processing

datetime Module (Accessible as datetime.x)

MINYEAR	Property describing the **smallest** year number permitted in classes generated by the module
MAXYEAR	Property describing the **largest** year number permitted in classes generated by the module
date()	Returns a date object for storing months, days, and years
date.today()	Returns the current local date
date.fromtimestamp()	Returns the local date supplied by the POSIX timestamp
date.fromordinal()	Returns the date based on an ordinal of the proleptic Gregorian calendar
date.fromisoformat()	Returns a date based on a **date_string**
date.fromisocalendar()	Returns a date based on an **ISO calendar date**
time()	Returns a time object for storing hours, minutes, seconds, microseconds, and offset data
time.fromisoformat()	Returns a time corresponding to a given time_string
timedelta()	Creates a timedelta object
datetime()	Returns a datetime object that has all the properties of both date and time
datetime.now()	Returns a datetime with the current local date and time
datetime.utcnow()	Returns a naive datetime with the current UTC date and time

datetime.utcfromtimestamp()	Returns a naive datetime corresponding to the POSIX timestamp
datetime.combine()	Returns a datetime combining a given date and time object
datetime.strptime()	Returns a datetime from a string according to a given format
timezone()	Creates a timezone object specifying an offset

Classes of the datetime Module

date	**A naive date object**
Properties of the date Class (Accessible as date_Instance.x)	
replace()	Returns a date with altered parameters
timetuple()	Returns a time.struct_time object
toordinal()	Returns the ordinal of the proleptic Gregorian calendar of the date
weekday()	Returns the day of the week as an integer, starting at **0**
isoweekday()	Returns the day of the week as an integer, starting at **1**
isocalendar()	Returns the date as a tuple representing an ISO calendar date
isoformat() or __str__()	Returns a string representing the date in a **YYYY-MM-DD fashion**
ctime()	Returns a string representing the **date and time**

strftime() or __format__()	Returns a string representing the date in a **specified format**
min	The **earliest** representable date
max	The **latest** representable date
resolution	The smallest possible difference between dates
year	A number between the MINYEAR and MAXYEAR properties of the module
month	A number between 1 and 12
day	A number between 1 and the number of days in the current date's month
time	A time object without respect to any given day

Properties of the time Class (Accessible as time_Instance.x)

replace()	Returns a time with altered parameters
isoformat() or __str__()	Returns a string representing the time in **ISO 8601** format
strftime() or __format__()	Returns a string representing the time with a **specific** format
utcoffset()	Returns the **Coordinated Universal Time (UTC)** offset of the time, if any
dst()	Returns the **daylight savings time** offset of the time, if any
tzname()	Returns the **name** of the time offset of the time, if any
min	The **earliest** representable time
max	The **latest** representable time
resolution	The smallest possible difference between times
hour	A number in the range of 24
minute	A number in the range of 60
second	A number in the range of 60
microsecond	A number in the range of **1000000**

tzinfo	A tzinfo object associated with the time
fold	A value used to disambiguate times when they change due to a time offset
timedelta	An object representing the duration between two dates or times

Properties of the timedelta Class (Accessible as timedelta_Instance.x)

total_seconds()	Returns the total number of seconds in the timedelta
min	The **smallest negative** timedelta object
max	The **largest positive** timedelta object
resolution	The smallest possible difference between timedelta objects
days	A number between **–999999999** and **999999999**
seconds	A number between **0** and **86399**
microseconds	A number between **0** and **999999**
datetime	An object holding properties of both date and time objects

Properties of the datetime Class (Accessible as datetime_Instance.x)

Note: In addition to these properties, datetime objects have all the properties of both date and time objects.

astimezone()	Returns a datetime adjusted for a new time zone
utctimetuple()	Returns a time.struct_time object adjusted for a new time zone
tzinfo	An abstract object representing the time offset for time and datetime objects
timezone	A subclass of the tzinfo object, an object representing a given offset

Properties of the timezone Class (Accessible as timezone_Instance.x)

utcoffset()	Returns the offset value of the object
tzname()	Returns the identifying value specified at object construction
fromutc()	Returns an aware datetime adjusted for offset
utc	The UTC timezone

System & Computer Controls

os Module (Accessible as os.x)

fsdecode()	Decodes the path-like filename from the filesystem
fspath()	Returns the filesystem representation of the path
getenv()	Returns the value of the environmental variable key
get_exec_path()	Returns a list of directories to be searched for a specific executable when the process is launched
getlogin()	Returns the name of the user logged in to the controlling terminal of the process
getppid()	Returns the process id of the parent process
putenv()	Sets a named environment variable to a value
strerror()	Returns an error message corresponding to a specific piece of code
umask()	Sets the current umask and returns the previous one
fdopen()	Returns the file object associated with a file descriptor
close()	Closes a **specific** file descriptor
closerange()	Closes a **range** of file descriptors
device_encoding()	Returns a description of the device encoding of a file descriptor
dup()	**Returns** a duplicate of a file descriptor
dup2()	**Closes and returns** a duplicate of a file descriptor
fstat()	Gets the status of a file descriptor
fsync()	Forces a disk-write of a file
ftruncate()	Truncates a file to a specific byte size
isatty()	Returns True if the file is connected to a tty device
lseek()	Sets the current position of a file descriptor to a specific position
open()	Opens a file path with parameters
pipe()	Creates a pipe
get_terminal_size()	Returns the size of the terminal window
get_inheritable()	**Gets** the Boolean inheritable flag of a **file descriptor**
set_inheritable()	**Sets** the Boolean inheritable flag of a **file descriptor**
get_handle_inheritable()	**Gets** the Boolean inheritable flag of a **handle**
set_handle_inheritable()	**Sets** the Boolean inheritable flag of a **handle**

access()	Tests for access to a specific path
chdir()	Changes the current directory to one specified by a path
chmod()	Changes the mode of a path to a specified mode
getcwd()	Returns a **string** representing the current working directory
getcwdb()	Returns a **bytestring** representing the current working directory
link()	Creates and names a link pointing to a path
listdir()	Returns a list of entries in a directory at a specific path
lstat()	Performs a system call on a given path without following symbolic links
mkdir()	Creates a directory
makedirs()	Creates a directory and all intermediate directories leading to it
major()	Extracts the device **major** number from a raw device number
minor()	Extracts the device **minor** number from a raw device number
makedev()	Creates a raw device number
readlink()	Returns a string representing a path pointed to by a symbolic link
remove()	Removes a file
removedirs()	Removes a directory and the directories leading to it
rename()	Renames a file or directory
renames()	Renames a file or directory, creating required intermediate directories
replace()	Renames a file or directory, replacing an existing file if one already exists with the new name
rmdir()	Removes a directory
scandir()	Returns an iterator representing all entries in a given directory or file
stat()	Returns a stat_result object for a path
symlink()	Creates a symbolic link
truncate()	Truncates a file down to a specific size
unlink()	Removes a file
utime()	Sets the access and modified times of a file
walk()	Returns all filenames in a directory tree
abort()	Aborts the current process

add_dll_directory()	Adds a path to the DLL search path
execl() or execv()	Executes a new program
exit()	Exits the current process
kill()	Kills a process with a specific signal
popen()	Returns a file connected to a command through an opened pipe
startfile()	Starts a file using the associated application
system()	Executes a command in a subshell
times()	Returns an object containing user and system times
waitpid()	Waits for the completion of a process and returns its ID and status
cpu_count()	Returns the number of CPUs in the system
urandom()	Returns a random string of a specific size
name	The name of the imported operating system dependent module
environ	A mapping object representing environmental variables/directories
error	An error representing system-related errors

Classes of the os Module

DirEntry	**An object yielded by scandir() that contains directory, file, or symbolic link information**

Properties of the DirEntry Class (Accessible as DirEntry_Instance.x)

name	The base filename of the object
path	The full pathname of the object
inode()	Returns the inode number of the object
is_dir()	Returns True if the object is a **directory**
is_file()	Returns True if the object is a **file**
is_symlink()	Returns True if the object is a **symbolic link**
stat()	Returns a stat_result object for the object

pathlib Module (Accessible as pathlib.x)

PurePath()	Creates a **PurePath** object
Path()	Creates a **Path** object
Path.cwd()	Creates a Path object representing the **current directory**
Path.home()	Creates a Path object representing the **home directory** of the user

Classes of the pathlib Module

PurePath	**A generic Path object that can handle paths without accessing a filesystem**

Properties of the PurePath Class (Accessible as PurePath_Instance.x)

as_posix()	Returns a string of a path with **forward slashes**
as_uri()	Returns a string of a path as a file **URI**
is_absolute()	Returns True if the path is **absolute**
is_reserved()	Returns True if the path is **reserved**
joinpath()	Combines a path with specified elements
match()	Returns True if a path matches a specified pattern
relative_to()	Returns a version of a path relative to another path
with_name()	Returns a path with an altered **name**
with_suffix()	Returns a path with an altered **suffix**
parts	A tuple giving access to the path's components
drive	A string representing the **drive name**
root	A string representing the **path root**
anchor	The drive and root of the path concatenated
parents	An immutable sequence allowing access to a path's ancestors
parent	The immediate ancestor of a path
name	A **string representing** the final part of the path
suffix	The **file extension** of the final part of the path
suffixes	A list of all file extensions of the final element of the path
stem	The final path component, without suffixes
Path	**A subclass of Purepath with the functionality to do system calls on Path objects**

Properties of the Path Class (Accessible as Path_Instance.x)

Note: The Path class has access to all properties of a PurePath in addition to the following properties.

stat()	Returns a stat_result object containing information about the path
chmod()	Changes the file mode and permissions of the path
exists()	Returns True if the path points to an existing file or directory
expanduser()	Returns an expanded path, adding ~ and ~user constructs
glob()	Returns all matching files in a path for a given pattern
group()	Returns the name of the group that owns the file
is_dir()	Returns True if the path points to a **directory**
is_file()	Returns True if the path points to a **file**
is_symlink()	Returns True if the path points to a **symbolic link**
is_fifo()	Returns True if the path points to a **FIFO file**
is_block_device()	Returns True if the path points to a **block device**
is_char_device()	Returns True if the path points to a **character device**
iterdir()	If the path points to a directory, yields Path objects within
lchmod()	Changes the file mode and permissions of the path, changing the mode of symbolic links rather than following them
mkdir()	Creates a new directory at a path
open()	Opens the file pointed to by the path
owner()	Returns the name of the user who owns a file
read_bytes()	Returns the contents of a file as a **bytes object**
read_text()	Returns the contents of a file as a **string**
rename()	Renames a file or directory and returns a path to it
replace()	Renames a file or directory and returns a path to it, replacing existing files in the process
resolve()	Returns a new path object that is an absolute representation of a given path
rglob()	Returns files that match a pattern and the path that precedes them
rmdir()	Removes a directory
samefile()	Returns True if a path points to the same file as a specified path
symlink_to()	Makes a path into a symlink leading to a specified target
touch()	Creates a file at a path's destination
unlink()	Removes a file or symbolic link
link_to()	Creates a hard link to a specified path
write_bytes()	Opens a file in **bytes** mode, writes data to it, and closes it
write_text()	Opens a file in **text** mode, writes data to it, and closes it

threading Module (Accessible as threading.x)

active_count()	Returns the number of Thread objects currently alive
current_thread()	Returns the current Thread object
excepthook()	Handles uncaught exceptions raised by Thread.run()
get_ident()	Returns the **thread identifier** of the current thread
get_native_id()	Returns the **integral thread ID** of the current thread
enumerate()	Returns a list of all Thread objects currently alive
main_thread()	Returns the main Thread object
settrace()	Sets a **trace** function for all threads started from this module
setprofile()	Sets a **profile** function for all threads started from this module
stack_size()	Returns the thread stack size used when creating new threads
Thread()	Creates an object of the Thread class

Classes of the threading Module

Thread	**An object that represents an activity run on a separate thread of control**

Properties of the Thread Class (Accessible as Thread_Instance.x)

start()	Starts the thread's activity
run()	Represents the activity of the thread
join()	Stops the calling thread until the joined thread terminates
name	A string identifying the thread
ident	A unique identifier for a running thread
native_id	The native integral thread ID of the thread that is unique system-wide
is_alive()	Returns whether the thread is alive
daemon	A Boolean value indicating whether the thread is a daemon thread
Lock	**An object that prevents multiple threads from accessing content simultaneously that can be released from any thread**

Properties of the Lock Class (Accessible as Lock_Instance.x)	
acquire()	**Acquires** a Lock
release()	**Releases** a Lock
locked()	Returns True if the lock is acquired
RLock	**An object that prevents multiple threads from accessing content simultaneously that can be released only by the thread that holds it**

Properties of the RLock Class (Accessible as RLock_Instance.x)	
acquire()	**Acquires** a Lock or **increases** the amount of locks a thread has acquired
release()	**Releases** a Lock or **decreases** the amount of locks a thread has acquired
Condition	**A condition variable object that allows threads to queue while waiting for lock access**

Properties of the Condition Class (Accessible as Condition_Instance.x)	
acquire()	Runs the associated function in the associated lock
release()	Runs the associated function in the associated lock
wait()	Waits until a notification or timeout occurs
wait_for()	Waits until a condition evaluates as True
notify()	Wakes up a set number of threads waiting on the Condition
notify_all()	Wakes up all threads waiting on the Condition
Semaphore	**An object that prevents multiple threads from accessing content simultaneously that is managed by a counter**

Properties of the Semaphore Class (Accessible as Semaphore_Instance.x)	
acquire()	Acquires a Semaphore, **decrementing** the counter
release()	Releases a Semaphore, **incrementing** the counter
Event	**An event object that allows threads to be blocked using signals**

Properties of the Event Class (Accessible as Event_Instance.x)	
is_set()	Returns True if the internal flag is True
set()	Sets the internal flag to True
clear()	Resets the internal flag to False
wait()	Blocks until the internal flag is set to True
Timer	**A special Thread object that is run after a certain amount of time passes**

Properties of the Timer Class (Accessible as Timer_Instance.x)	
cancel()	Stops the Timer and cancels the execution of its action
Barrier	**An object that blocks threads until all threads are ready**

Properties of the Barrier Class (Accessible as Barrier_Instance.x)	
wait()	Signifies that a thread is ready to pass the barrier
reset()	Returns the barrier to an empty state
abort()	Causes all future calls of wait() to fail
parties	The number of threads that must make wait() calls to pass the barrier
n_waiting()	The number of threads currently waiting
broken()	Set to True when the barrier is aborted

Debugging Functionality

PDB Module (Accessible as PDB.x)

set_trace()	Pauses the code to enter a debugger
run()	Executes the statement in debugger mode
runeval()	Evaluates an expression in debugger mode
runcall()	Calls a function in debugger mode
post_mortem()	Starts post-mortem debugging using **specified** traceback
pm()	Starts post-mortem debugging using the **last** traceback

Debugging Commands for the PDB Module	
h or help	Prints a list of all available commands or information about one specific command entered as an argument
w or where	Prints a stack trace with the most recent time at the bottom
d or down	Moves the current frame a specified amount of levels **down** the stack trace (**newer**)
u or up	Moves the current frame a specified amount of levels **up** the stack trace (**older**)
b or break	Creates or lists breakpoints in a file
tbreak	Creates a temporary breakpoint
cl or clear	**Clears** some or all of the breakpoints in a file
disable	**Disables** some or all of the breakpoints in a file
enable	**Enables** some or all of the breakpoints in a file
ignore	Sets an ignore count for a breakpoint
condition	Sets a condition for a breakpoint to activate
commands	Adds a list of commands that will be available when the specified breakpoint is reached
s or step	Executes the current line of code and stops afterward
n or next	Executes the current line of code and stops afterward, **skipping called functions**
unt or until	Executes code until a **certain line number is reached**
r or return	Executes code until the **current function returns**
c, cont, or continue	Executes code until a **breakpoint is reached**
j or jump	Executes the **specified line** of code
l or list	Lists source code for the current **file in a specified range**
ll	Lists source code for the current **function or frame**
a or args	Prints the arguments for the current function
p	Evaluates the current expression and prints its value
pp	Evaluates the current expression and prints its value **in a formattable way**
whatis	Prints the type of an expression
source	Displays the source code for the current object
display	Displays the value of the expression if it has changed
undisplay	Stops the display of the expression in the current frame
interact	Starts an interactive interpreter containing all names in the current scope
alias	Create an alias that executes a command
unalias	Removes a specified alias
restart	Restarts the current program
q or quit	Quits the debugger and exits the program
debug	Enters a debugger inside the debugger
retval	Prints the return value that was last returned from a function

Mathematic & Numeric Operations

Math Module (Accessible as Math.x)

ceil()	Returns the smallest integer greater than or equal to a number
comb()	Returns the number of different combinations of a certain length in a certain size set of numbers
copysign()	Returns the value of one number with the sign of another
fabs()	Returns the absolute value of a number
factorial()	Returns an inter value factorial for a number
floor()	Returns the largest integer less than or equal to a number
fmod()	Returns the remainder of dividing two numbers in float form
frexp()	Returns a mantissa-exponent pair such that the mantissa multiplied by 2 to the exponent returns a specific number
fsum()	Returns a sum computed to a high precision in the case of floating-point numbers
gcd()	Returns the greatest common divisor of given numbers
isclose()	Returns True if two values are close to one another and False if they are not
isfinite()	Returns True if a number is **finite**
isinf()	Returns True if a number is **infinite**

`isnan()`	Returns `True` if a value is not a number
`isqrt()`	Returns the square root of an integer as an integer
`ldexp()`	Returns the result of multiplying a number by 2 to the power of another number
`modf()`	Returns the integer and the decimal of a number separated
`perm()`	Returns the number of permutations possible of a given size from a given number
`prod()`	Returns the product of a set of numbers
`remainder()`	Returns the remainder of dividing, such that it returns the difference expressed as $x - n \times y$, where n is the nearest integer to x/y
`trunc()`	Returns the value truncated to an integer
`exp()`	Returns the result of raising the value of *e* to the power of a number
`expml()`	Returns the result of raising the value of *e* to the power of a number **and subtracting 1**
`log()`	Returns the natural logarithm of a number to a given base
`log1p()`	Returns the natural logarithm of 1 plus a number to the base of *e*
`log2()`	Returns the base-**2** logarithm of a number
`log10()`	Returns the base-**10** logarithm of a number
`pow()`	Returns a number to the power of another number
`sqrt()`	Returns the square root of a number
`acos()`	Returns the **arc cosine** of a number in radians
`asin()`	Returns the **arc sine** of a number in radians
`atan()`	Returns the **arc tangent** of a number in radians
`cos()`	Returns the **cosine** of a number of radians
`dist()`	Returns the Euclidean distance between two points in the same dimension
`hypot()`	Returns the length of a vector from the origin point to a given point
`sin()`	Returns the **sine** of a number of radians
`tan()`	Returns the **tangent** of a number of radians
`degrees()`	Converts an angle from **radians to degrees**
`radians()`	Converts an angle from **degrees to radians**
`pi`	Equivalent to the mathematical constant **pi (3.141592...)**
`e`	Equivalent to the mathematical constant ***e* (2.718281...)**
`tau`	Equivalent to the mathematical constant **tau**, which is twice the value of pi
`inf`	Equivalent to positive infinity
`nan`	Expression conferring the value of "not a number"

`random` Module (Accessible as `random.x`)

`seed()`	Initializes the random number generator
`getstate()`	Returns an object capturing the state of the random number generator
`setstate()`	Restores the state of the random number generator to a previously captured state
`getrandbits()`	Returns an integer with a set amount of random bits
`randrange()`	Returns a random element from a range
`randint()`	Returns a random integer
`choice()`	Returns a random element from a sequence
`choices()`	Returns a set amount of random elements from a sequence
`shuffle()`	Shuffles the contents of a sequence
`sample()`	Returns a set length list of random unique elements from a set
`random()`	Returns the next random float from a range
`uniform()`	Returns a random float between two **numbers**
`triangular()`	Returns a random float between two **bounds and a set mode**
`betavariate()`	Returns a random number from a **beta** distribution
`expovariate()`	Returns a random number from an **exponential** distribution
`gammavariate()`	Returns a random number from a **gamma** distribution
`gauss()`	Returns a random number from a **Gaussian** distribution
`lognormvariate()`	Returns a random number from a **log-normal** distribution
`normalvariate()`	Returns a random number from a **normal** distribution
`vonmisesvariate()`	Returns a random number from a **von Mises** distribution
`paretovariate()`	Returns a random number from a **Pareto** distribution
`weibullvariate()`	Returns a random number from a **Weibull** distribution

Iterable & Iterator Operations

`collections` Module (Accessible as `collections.x`)

`Counter()`	Creates a `Counter` object that tallies objects in iterables
`defaultdict()`	Creates a `defaultdict`, a dictionary that can produce default values
`OrderedDict()`	Creates a `OrderedDict`, a dictionary in which the keys do not change position regardless of value
`deque()`	Creates a `deque`, a list item that specializes in adding and removing elements
`ChainMap()`	Creates a `ChainMap` that combines the values in multiple dictionaries
`namedtuple()`	Creates a `namedtuple`, a tuple that uses names instead of indexes

Classes of the `collections` Module

`ChainMap`	**An object that groups multiple mappings together**
Properties of the `ChainMap` Class (Accessible as `ChainMap_Instance.x`)	
`maps`	A list of mappings
`parents`	Returns a new `ChainMap` containing all maps in the current instance except the first one
`new_child()`	Returns a new `ChainMap` containing a new map and all maps in the current instance
`Counter`	**An object for storing tallies of objects**
Properties of the `Counter` Class (Accessible as `Counter_Instance.x`)	
`elements()`	Returns all objects in a `Counter`
`most_common()`	Returns a specified number of the most common elements of a `Counter`
`subtract()`	Subtracts object counts from a `Counter`
`update()`	Adds counted iterables to the `Counter`
`deque`	**An object supporting operations from either side**

Properties of the `deque` Class (Accessible as `deque_Instance.x`)	
`append()`	Adds an element to the **right** side of the deque
`appendleft()`	Adds an element to the **left** side of the deque
`clear()`	Removes all elements from the deque
`copy()`	Creates a shallow copy of the deque
`count()`	Counts the number of specified items in the deque
`extend()`	Adds elements to the **right** side of the deque
`extendleft()`	Adds elements to the **left** side of the deque
`index()`	Returns the position of an element in the deque
`insert()`	Inserts an element into the deque at a specified position
`pop()`	Removes and returns an element from the **right** side of the deque
`popleft()`	Removes and returns an element from the **left** side of the deque
`remove()`	Removes the first occurrence of a value in the deque
`reverse()`	Reverses the deque
`rotate()`	Appends one side of the deque by popping a specified number of elements from the other side
`maxlen`	The maximum size of the deque
`defaultdict`	**A `dict` object with the ability to set default values**
Properties of the `defaultdict` Class (Accessible as `defaultdict_Instance.x`)	
`__missing__()`	Provides a default value to a specified key
`default_factory`	An attribute holding the default value that may be applied to keys within the `dict`
`namedtuple`	**A tuple where fields are named**

Properties of the namedtuple Class (Accessible as namedtuple_Instance.x)	
_make()	Creates a new instance from an existing iterable
_asdict()	Creates a new dict mapping field names to corresponding values
_replace()	Returns a new instance of the tuple replacing specified fields with new values
_fields	A list of field names for the tuple
_field_defaults	A dict mapping the field names of the tuple to their default values
OrderedDict	**A dict with extra functionality for ordering elements**

Properties of the OrderedDict Class (Accessible as OrderedDict_Instance.x)	
popitem()	Returns and removes a key value pair from the OrderedDict
move_to_end()	Moves an existing key to an end of the OrderedDict
_make()	Creates a new instance from an existing iterable

itertools Module (Accessible as itertools.x)

accumulate()	Creates an iterator that returns accumulated sums
chain()	Creates an iterator that combines results from several iterables
from_iterable()	Returns an iterator that gets chained input from a single iterable argument
combinations()	Returns a set of subsequences from an iterable equal in length to a specific number
combinations_with_replacement()	Returns a specified length of subsequences from a given iterable, allowing repetition
compress()	Returns elements of an iterator corresponding to a provided list
count()	Creates an iterator that returns evenly spaced values
cycle()	Creates an iterator that returns elements, copying them and returning the copies when exhausted
dropwhile()	Creates an iterator that drops elements until a condition becomes True
filterfalse()	Creates an iterator that filters items from an iterable where a condition is False
groupby()	Creates an iterator that returns **keys and groups consecutively**
islice()	Creates an iterator that returns **selected elements from an iterable**
permutations()	Returns successive permutations of elements from an iterable of a specified length
product()	Returns the product of several iterables sequentially
repeat()	Creates an iterator that returns an object a set amount of times
starmap()	Creates an iterator that resolves a function using provided arguments in iterable form
takewhile()	Creates an iterator that returns elements while a condition is True
tee()	Returns a set amount of independent iterators from a single iterable
zip_longest()	Creates an iterator that aggregates several iterables, with a fill value for unequal iterables

Web & Data Transfer Operations

HTMLParser Module (Accessible as HTMLParser.x)

feed()	Feeds text to the parser
close()	Forces all buffered data to be processed
reset()	Resets the current instance, losing all unprocessed data
getpos()	Gets current line number and offset
get_starttag_text()	Returns text inside the most recently opened start tag
handle_starttag()	Handles the HTML **start** tag
handle_endtag()	Handles the HTML **end** tag
handle_startendtag()	Handles the HTML **combined start and end tag**
handle_data()	Handles arbitrary data
handle_entityref()	Handles named character references
handle_charref()	Handles decimal and hexadecimal numeric character references
handle_comment()	Handles comments
handle_decl()	Handles doctype declarations
handle_pi()	Handles processing instructions
unknown_decl()	Called when there is an unrecognized declaration

HTML Module (Accessible as HTML.x)

escape()	Converts the characters &, <, and > to HTML-safe characters
unescape()	Converts character references to Unicode characters

JSON Module (Accessible as JSON.x)

dump()	Converts a Python structure to a JSON **file**
dumps()	Converts a Python structure to a JSON **formatted string**
load()	Converts a JSON **object** to a Python object
loads()	Converts a JSON **formatted string** to a Python object

Audio Manipulation

audioop Module (Accessible as audioop.x)

add()	Returns a sound fragment that combines two samples
adpcm2lin()	Decodes an **Intel/DVI ADPCM** coded sound fragment into a linear fragment
alaw2lin()	Decodes an **a-LAW** coded sound fragment into a linear fragment
avg()	Returns the average over all samples in a sound fragment
avgpp()	Returns the average **peak-peak value** over all samples in a sound fragment
bias()	Returns a sound fragment with a **bias added to all samples**
byteswap()	Returns a sound fragment with **all samples byteswapped**
cross()	Returns the number of zero crossings in a sound fragment
findfactor()	Returns a specified factor of a sound fragment
findmax()	Returns the sample slice of a sound fragment with the highest energy
getsample()	Returns the value of a sample at a specified position in a sound fragment
lin2adpcm()	Converts a linear sound fragment to **Intel/DVI ADPCM** encoding
lin2alaw()	Converts a linear sound fragment to **a-LAW** encoding
lin2lin()	Converts a linear sound fragment to **one of a different byte format**
lin2ulaw()	Converts a linear sound fragment to **u-LAW** encoding
max()	Returns the maximum of the **absolute value of all samples** in a sound fragment
maxpp()	Returns the maximum **peak-peak value** in a sound fragment
minmax()	Returns the minimum and maximum values of all samples in a sound fragment
mul()	Returns a sound fragment with all samples multiplied by a specified value
ratecv()	Converts the frame rate of a sound fragment
reverse()	Returns a sound fragment with all samples reversed
rms()	Returns the power of the audio signal of a sound fragment
tomono()	Converts a stereo sound fragment **to** a mono sound fragment
tostereo()	Creates a stereo sound fragment **from** a mono sound fragment
ulaw2lin()	Decodes a u-Law coded sound fragment into a linear fragment

$7.95 **Author:** Berajah Jayne

ISBN-13: 978-142324423-3
ISBN-10: 142324423-0

9 781423 244233 50795

6 54614 04423 5