



# python™

## Standard Library Python 3.x

### General Functionality

#### Built-In (Accessible Directly)

<code>abs()</code>	Returns the absolute value of an integer	<code>isinstance()</code>	Returns <code>True</code> if an object is of a certain class or subclass
<code>all()</code>	Returns <code>True</code> if <code>all</code> elements of an iterable are <code>True</code>	<code>issubclass()</code>	Returns <code>True</code> if a class is a subclass of the provided class
<code>any()</code>	Returns <code>True</code> if <code>any</code> element of an iterable is <code>True</code>	<code>iter()</code>	Returns an iterator object
<code>ascii()</code>	Returns a string without the non-ASCII characters	<code>len()</code>	Returns the number of items in an object
<code>bin()</code>	Returns an integer as a binary string	<code>locals()</code>	Returns a <code>dict</code> representing the current local symbol table
<code>breakpoint()</code>	Creates a <code>breakpoint</code> and enters the debugger	<code>map()</code>	Returns an iterator which applies a function to each element of an iterable
<code>bytearray()</code>	Returns an array of bytes objects	<code>min()</code>	Returns the smallest element in an iterable
<code>bytes()</code>	Returns a bytes object	<code>next()</code>	Gets the next element in an iterator
<code>callable()</code>	Returns <code>True</code> if the object argument is callable	<code>oct()</code>	Converts an integer into an octal string
<code>chr()</code>	Returns a string character using a <code>unicode</code> argument	<code>open()</code>	Opens a file and returns a file object
<code>compile()</code>	Compiles code into an executable object	<code>ord()</code>	Returns an integer that is the <code>unicode</code> representation of a provided character
<code>delattr()</code>	Removes attributes of an object	<code>pow()</code>	Returns a base number raised to a specified power
<code>dir()</code>	Returns a list of names or attributes from the local scope	<code>print()</code>	Prints objects to a text stream
<code>divmod()</code>	Return the quotient and remainder of two numbers	<code>repr()</code>	Returns a string that is a printable representation of an object
<code>enumerate()</code>	Returns a value obtained from iterating over an iterable	<code>reversed()</code>	Returns an iterator in reverse order
<code>eval()</code>	Evaluates an expression using a set of local and global variables	<code>round()</code>	Returns a number that has been rounded to a specified number of digits
<code>exec()</code>	Executes the code using a set of local and global variables	<code>setattr()</code>	Assigns a value to a specified attribute of an object
<code>filter()</code>	Creates an iterator for a set of functions that return <code>True</code>	<code>sorted()</code>	Returns the items of an iterator in a specified order
<code>format()</code>	Creates a formatted representation of a value from a provided value	<code>sum()</code>	Returns the added value of the provided iterable
<code>getattr()</code>	Returns the value of a named attribute of an object	<code>super()</code>	Returns an object that delegates inherited methods back to the parent class
<code>globals()</code>	Returns a <code>dict</code> containing the current global symbol table	<code>vars()</code>	Returns the <code>__dict__</code> attribute for an object
<code>hasattr()</code>	Returns <code>True</code> if the provided object has the provided attribute	<code>zip()</code>	Returns an iterator that combines elements from several iterables
<code>hash()</code>	Returns the hash value of an object	<code>__import__()</code>	Imports a module, specifying the context under which it will be done
<code>help()</code>	Starts Python's built-in help system		
<code>hex()</code>	Converts a number to a hexadecimal string		
<code>id()</code>	Returns a unique identifier for an object		
<code>input()</code>	Returns a value based on user input		

### Date/Time Processing

#### `datetime` Module (Accessible as `datetime.x`)

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

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

#### Classes of the `datetime` Module

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

<code>strftime()</code> or <code>__format__()</code>	Returns a string representing the date in a <b>specified format</b>	<code>tzinfo</code>	A <code>tzinfo</code> object associated with the time
<code>min</code>	The <b>earliest</b> representable date	<code>fold</code>	A value used to disambiguate times when they change due to a time offset
<code>max</code>	The <b>latest</b> representable date	<code>timedelta</code>	An object representing the duration between two dates or times
<code>resolution</code>	The smallest possible difference between dates	<b>Properties of the timedelta Class (Accessible as <code>timedelta_Instance.x</code>)</b>	
<code>year</code>	A number between the MINYEAR and MAXYEAR properties of the module	<code>total_seconds()</code>	Returns the total number of seconds in the <code>timedelta</code>
<code>month</code>	A number between 1 and 12	<code>min</code>	The <b>smallest negative</b> <code>timedelta</code> object
<code>day</code>	A number between 1 and the number of days in the current date's month	<code>max</code>	The <b>largest positive</b> <code>timedelta</code> object
<code>time</code>	A time object without respect to any given day	<code>resolution</code>	The smallest possible difference between <code>timedelta</code> objects
<b>Properties of the time Class (Accessible as <code>time_Instance.x</code>)</b>			
<code>replace()</code>	Returns a time with altered parameters	<code>days</code>	A number between -99999999 and 99999999
<code>isoformat()</code> or <code>__str__()</code>	Returns a string representing the time in <b>ISO 8601</b> format	<code>seconds</code>	A number between 0 and 86399
<code>strftime()</code> or <code>__format__()</code>	Returns a string representing the time with a <b>specific</b> format	<code>microseconds</code>	A number between 0 and 999999
<code>utcoffset()</code>	Returns the <b>Coordinated Universal Time (UTC)</b> offset of the time, if any	<code>datetime</code>	An object holding properties of both date and time objects
<code>dst()</code>	Returns the <b>daylight savings time</b> offset of the time, if any	<b>Properties of the datetime Class (Accessible as <code>datetime_Instance.x</code>)</b>	
<code>tzname()</code>	Returns the <b>name</b> of the time offset of the time, if any	<code>astimezone()</code>	Returns a <code>datetime</code> adjusted for a new time zone
<code>min</code>	The <b>earliest</b> representable time	<code>utctimetuple()</code>	Returns a <code>time.struct_time</code> object adjusted for a new time zone
<code>max</code>	The <b>latest</b> representable time	<code>tzinfo</code>	An abstract object representing the time offset for time and <code>datetime</code> objects
<code>resolution</code>	The smallest possible difference between times	<code>timezone</code>	A subclass of the <code>tzinfo</code> object, an object representing a given offset
<code>hour</code>	A number in the range of 24	<b>Properties of the timezone Class (Accessible as <code>timezone_Instance.x</code>)</b>	
<code>minute</code>	A number in the range of 60	<code>utcoffset()</code>	Returns the offset value of the object
<code>second</code>	A number in the range of 60	<code>tzname()</code>	Returns the identifying value specified at object construction
<code>microsecond</code>	A number in the range of 1000000	<code>fromutc()</code>	Returns an aware <code>datetime</code> adjusted for offset
		<code>utc</code>	The UTC timezone

## System & Computer Controls

### os Module (Accessible as `os.x`)

<code>fsdecode()</code>	Decodes the path-like filename from the filesystem	<code>access()</code>	Tests for access to a specific path
<code>fspath()</code>	Returns the filesystem representation of the path	<code>chdir()</code>	Changes the current directory to one specified by a path
<code>getenv()</code>	Returns the value of the environmental variable key	<code>chmod()</code>	Changes the mode of a path to a specified mode
<code>get_exec_path()</code>	Returns a list of directories to be searched for a specific executable when the process is launched	<code>getcwd()</code>	Returns a <b>string</b> representing the current working directory
<code>getlogin()</code>	Returns the name of the user logged in to the controlling terminal of the process	<code>getcwdb()</code>	Returns a <b>bytestring</b> representing the current working directory
<code>getppid()</code>	Returns the process id of the parent process	<code>link()</code>	Creates and names a link pointing to a path
<code>putenv()</code>	Sets a named environment variable to a value	<code>listdir()</code>	Returns a list of entries in a directory at a specific path
<code>strerror()</code>	Returns an error message corresponding to a specific piece of code	<code>lstat()</code>	Performs a system call on a given path without following symbolic links
<code>umask()</code>	Sets the current <code>umask</code> and returns the previous one	<code>mkdir()</code>	Creates a directory
<code>fdopen()</code>	Returns the file object associated with a file descriptor	<code>makedirs()</code>	Creates a directory and all intermediate directories leading to it
<code>close()</code>	Closes a <b>specific</b> file descriptor	<code>major()</code>	Extracts the device <b>major</b> number from a raw device number
<code>closerange()</code>	Closes a <b>range</b> of file descriptors	<code>minor()</code>	Extracts the device <b>minor</b> number from a raw device number
<code>device_encoding()</code>	Returns a description of the device encoding of a file descriptor	<code>makedev()</code>	Creates a raw device number
<code>dup()</code>	<b>Returns</b> a duplicate of a file descriptor	<code>readlink()</code>	Returns a string representing a path pointed to by a symbolic link
<code>dup2()</code>	<b>Closes and returns</b> a duplicate of a file descriptor	<code>remove()</code>	Removes a file
<code>fstat()</code>	Gets the status of a file descriptor	<code>removedirs()</code>	Removes a directory and the directories leading to it
<code>fsync()</code>	Forces a disk-write of a file	<code>rename()</code>	Renames a file or directory
<code>ftruncate()</code>	Truncates a file to a specific byte size	<code>renames()</code>	Renames a file or directory, creating required intermediate directories
<code>isatty()</code>	Returns <b>True</b> if the file is connected to a <b>tty</b> device	<code>replace()</code>	Renames a file or directory, replacing an existing file if one already exists with the new name
<code>lseek()</code>	Sets the current position of a file descriptor to a specific position	<code>rmdir()</code>	Removes a directory
<code>open()</code>	Opens a file path with parameters	<code>scandir()</code>	Returns an iterator representing all entries in a given directory or file
<code>pipe()</code>	Creates a pipe	<code>stat()</code>	Returns a <code>stat_result</code> object for a path
<code>get_terminal_size()</code>	Returns the size of the terminal window	<code>symlink()</code>	Creates a symbolic link
<code>get_inheritable()</code>	<b>Gets</b> the Boolean inheritable flag of a <b>file descriptor</b>	<code>truncate()</code>	Truncates a file down to a specific size
<code>set_inheritable()</code>	<b>Sets</b> the Boolean inheritable flag of a <b>file descriptor</b>	<code>unlink()</code>	Removes a file
<code>get_handle_inheritable()</code>	<b>Gets</b> the Boolean inheritable flag of a <b>handle</b>	<code>utime()</code>	Sets the access and modified times of a file
<code>set_handle_inheritable()</code>	<b>Sets</b> the Boolean inheritable flag of a <b>handle</b>	<code>walk()</code>	Returns all filenames in a directory tree
		<code>abort()</code>	Aborts the current process

add_dll_directory()	Adds a path to the DLL search path
exec1() 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
<b>Classes of the os Module</b>	
DirEntry	An object yielded by scandir() that contains directory, file, or symbolic link information
<b>Properties of the DirEntry Class (Accessible as DirEntry_Instance.x)</b>	
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
<b>pathlib Module (Accessible as pathlib.x)</b>	
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
<b>Classes of the pathlib Module</b>	
PurePath	A generic Path object that can handle paths without accessing a filesystem
<b>Properties of the PurePath Class (Accessible as PurePath_Instance.x)</b>	
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

<b>Properties of the Path Class (Accessible as Path_Instance.x)</b>	
<b>Note:</b> 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
<b>threading Module (Accessible as threading.x)</b>	
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
<b>Classes of the threading Module</b>	
Thread	An object that represents an activity run on a separate thread of control
<b>Properties of the Thread Class (Accessible as Thread_Instance.x)</b>	
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
c1 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
expm1()	Returns the result of raising the value of $e$ to the power of a number <b>and subtracting 1</b>
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 <b>arc cosine</b> of a number in radians
asin()	Returns the <b>arc sine</b> of a number in radians
atan()	Returns the <b>arc tangent</b> of a number in radians
cos()	Returns the <b>cosine</b> 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 <b>sine</b> of a number of radians
tan()	Returns the <b>tangent</b> of a number of radians
degrees()	Converts an angle from <b>radians to degrees</b>
radians()	Converts an angle from <b>degrees to radians</b>
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 <b>numbers</b>
triangular()	Returns a random float between two <b>bounds and a set mode</b>
bivariate()	Returns a random number from a <b>beta</b> distribution
expovariate()	Returns a random number from an <b>exponential</b> distribution
gammavariate()	Returns a random number from a <b>gamma</b> distribution
gauss()	Returns a random number from a <b>Gaussian</b> distribution
lognormvariate()	Returns a random number from a <b>log-normal</b> distribution
normalvariate()	Returns a random number from a <b>normal</b> distribution
vonmisesvariate()	Returns a random number from a <b>von Mises</b> distribution
paretovariate()	Returns a random number from a <b>Pareto</b> distribution
weibullvariate()	Returns a random number from a <b>Weibull</b> 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
<b>Properties of the ChainMap Class (Accessible as ChainMap_Instance.x)</b>	
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
<b>Properties of the Counter Class (Accessible as Counter_Instance.x)</b>	
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 <b>right</b> side of the deque
appendleft()	Adds an element to the <b>left</b> 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 <b>right</b> side of the deque
extendleft()	Adds elements to the <b>left</b> 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 <b>right</b> side of the deque
popleft()	Removes and returns an element from the <b>left</b> 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
<b>Properties of the defaultdict Class (Accessible as defaultdict_Instance.x)</b>	
_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

**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

\$7.95 Author: Berajah Jayne

Disclaimer: This guide is intended for informational purposes only. Due to its condensed format, this guide cannot possibly cover every aspect of the subject. BarCharts, Inc. and its writers, editors, and design staff are not responsible or liable for the use or misuse of the information contained in this guide.

All rights reserved. No part of this publication may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without written permission from the publisher.

Made in the USA ©2020 BarCharts Publishing, Inc. 0720

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

**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
tonono()	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

ISBN-13: 978-142324438-7

ISBN-10: 142324438-9

hundreds of titles at  
**quicKstudy.com**Find us on  
Facebook