

Comparison Operators

| | | | |
|-----------------------|---------|------------------------|---------------|
| <code>x < y</code> | Less | <code>x <= y</code> | Less or eq |
| <code>x > y</code> | Greater | <code>x >= y</code> | Greater or eq |
| <code>x == y</code> | Equal | <code>x != y</code> | Not equal |

Boolean Operators

| | | |
|--------------------|----------------------|---------------------|
| <code>not x</code> | <code>x and y</code> | <code>x or y</code> |
|--------------------|----------------------|---------------------|

Arithmetic Operators

| | | | |
|--------------------|----------|---------------------|----------|
| <code>x + y</code> | add | <code>x - y</code> | subtract |
| <code>x * y</code> | multiply | <code>x / y</code> | divide |
| <code>x % y</code> | modulus | <code>x ** y</code> | x^y |

Assignment shortcuts: `x op= y`
 Example: `x += 1` increments `x`

Exception Handling

```
try:
    statements
except [ exception type [ asvar
] ]:
    statements
finally:
    statements
```

String / List / Tuple Operations

| | |
|-----------------------------|---|
| <code>len(s)</code> | length of <code>s</code> |
| <code>s[i]</code> | <code>i</code> th item in <code>s</code> (0-based) |
| <code>s[start : end]</code> | slice of <code>s</code> from <code>start</code> (included) to <code>end</code> (excluded) |
| <code>x in s</code> | True if <code>x</code> is contained in <code>s</code> |
| <code>x not in s</code> | True if <code>x</code> is not contained in <code>s</code> |
| <code>s + t</code> | the concatenation of <code>s</code> with <code>t</code> |
| <code>s * n</code> | <code>n</code> copies of <code>s</code> concatenated |
| <code>sorted(s)</code> | a sorted copy of <code>s</code> |
| <code>s.index(item)</code> | position in <code>s</code> of <code>item</code> |
| <code>s.lower()</code> | lowercase copy of <code>s</code> |

String / List / Tuple Operations (cont)

| | |
|----------------------------------|---|
| <code>s.replace(old, new)</code> | copy of <code>s</code> with <code>old</code> replaced with <code>new</code> |
| <code>s.split(delim)</code> | list of substrings delimited by <code>delim</code> |
| <code>s.strip()</code> | copy of <code>s</code> with whitespace trimmed |
| <code>s.upper()</code> | uppercase copy of <code>s</code> |
| See also | http://docs.python.org/library/stdtypes.html#stringmethods |

Environment

| | |
|--|--|
| <code>sys.argv</code> | List of command line arguments (<code>argv[0]</code> is executable) |
| <code>os.environ</code> | Dictionary of environment variables |
| <code>os.getcwd()</code> | String with path of current directory |
| <code>import sys; print(sys.argv)</code> or <code>from sys import argv; print(argv)</code> | |

Python List Methods

| | |
|-------------------------------------|----------------------------|
| <code>append(item)</code> | <code>pop(position)</code> |
| <code>count(item)</code> | <code>remove(item)</code> |
| <code>extend(list)</code> | <code>reverse()</code> |
| <code>index(item)</code> | <code>sort()</code> |
| <code>insert(position, item)</code> | |

Python sys.argv

| | |
|--|---------------------|
| <code>sys.argv[0]</code> | <code>foo.py</code> |
| <code>sys.argv[1]</code> | <code>bar</code> |
| <code>sys.argv[2]</code> | <code>-c</code> |
| <code>sys.argv[3]</code> | <code>qux</code> |
| <code>sys.argv[4]</code> | <code>--h</code> |
| <code>sys.argv</code> for the command: | |
| <code>\$ python foo.py bar -c qux --h</code> | |

Function Definitions

```
def name(arg1, arg2, ...):
    statements
    return expr
```

Conversion Functions

| | |
|--------------------------|---------------------------------------|
| <code>int(expr)</code> | Converts <code>expr</code> to integer |
| <code>float(expr)</code> | Converts <code>expr</code> to float |
| <code>str(expr)</code> | Converts <code>expr</code> to string |
| <code>chr(num)</code> | ASCII char <code>num</code> |

Data Types

| | |
|------------|------------------------------------|
| Integer | -256, 15 |
| Float | -253.23, 1.253e-10 |
| String | "Hello", 'Goodbye', ""Multiline"" |
| Boolean | True, False |
| List | [value, ...] |
| Tuple | (value, ...) ¹ |
| Dictionary | { key: value, ... } |
| Set | { value, value, ... } ² |

1. Parentheses usually optional
2. Create an empty set with `set()`

Mutating List Operations

| | |
|-------------------------------|--|
| <code>del lst[i]</code> | Deletes <code>i</code> th item from <code>lst</code> |
| <code>lst.append(e)</code> | Appends <code>e</code> to <code>lst</code> |
| <code>lst.insert(i, e)</code> | Inserts <code>e</code> before <code>i</code> th item in <code>lst</code> |
| <code>lst.sort()</code> | Sorts <code>lst</code> |

See also

<https://docs.python.org/3/library/stdtypes.html#typesseq-mutable>

Python File Methods

| | |
|-----------------------|------------------------------|
| <code>close()</code> | <code>readlines(size)</code> |
| <code>flush()</code> | <code>seek(offset)</code> |
| <code>fileno()</code> | <code>tell()</code> |
| <code>isatty()</code> | <code>truncate(size)</code> |



Python File Methods (cont)

| | |
|-------------------------|---------------------------|
| next() | write(<i>string</i>) |
| read(<i>size</i>) | writelines(<i>list</i>) |
| readline(<i>size</i>) | |

Python Indexes and Slices

| | |
|---------|-------------------|
| len(a) | 6 |
| a[0] | h |
| a[5] | n |
| a[-1] | n |
| a[-2] | m |
| a[1:] | [i,j,k,m,n] |
| a[:5] | [h,i,j,k,m] |
| a[:-2] | [h,i,j,k] |
| a[1:3] | [i,j] |
| a[1:-1] | [i,j,k,m] |
| b=a[:] | Shallow copy of a |

Indexes and Slices of a=[h,i,j,k,m,n]

Statements

If Statement

```
if expression:
    statements
```

```
elif expression:
    statements
```

```
else:
    statements
```

While Loop

```
while expression:
    statements
```

For Loop

```
for var in collection:
    statements
```

Counting For Loop

```
for i in range(start, end [,
step]):
    statements
(start is included; end is not)
```

Python Date Formatting

| | |
|----|-------------------------------------|
| %a | Abbreviated weekday (Sun) |
| %A | Weekday (Sunday) |
| %b | Abbreviated month name (Jan) |
| %B | Month name (January) |
| %c | Date and time |
| %d | Day (leading zeros) (01 to 31) |
| %H | 24 hour (leading zeros) (00 to 23) |
| %I | 12 hour (leading zeros) (01 to 12) |
| %j | Day of year (001 to 366) |
| %m | Month (01 to 12) |
| %M | Minute (00 to 59) |
| %p | AM or PM |
| %S | Second (00 to 61 ⁴) |
| %U | Week number ¹ (00 to 53) |
| %w | Weekday ² (0 to 6) |
| %W | Week number ³ (00 to 53) |
| %x | Date |
| %X | Time |
| %y | Year without century (00 to 99) |
| %Y | Year (2008) |
| %Z | Time zone (GMT) |
| %% | A literal "%" character (%) |

¹ Sunday as start of week. All days in a new year preceding the first Sunday are considered to be in week 0.

² 0 is Sunday, 6 is Saturday.

³ Monday as start of week. All days in a new year preceding the first Monday are considered to be in week 0.

⁴ This is not a mistake. Range takes account of leap and double-leap seconds.

Python Datetime Methods

| | |
|--------------------------------------|---|
| today() | fromordinal(<i>ordinal</i>) |
| now(<i>timezoneinfo</i>) | combine(<i>date</i> , <i>time</i>) |
| utcnow() | strptime(<i>date</i> , <i>format</i>) |
| fromtimestamp(<i>timestamp</i>) | |
| utcfromtimestamp(<i>timestamp</i>) | |

Python Class Special Methods

| | |
|-------------------------------|---------------------|
| __new__(cls) | __lt__(self, other) |
| __init__(self, args) | __le__(self, other) |
| __del__(self) | __gt__(self, other) |
| __repr__(self) | __ge__(self, other) |
| __str__(self) | __eq__(self, other) |
| __cmp__(self, other) | __ne__(self, other) |
| __index__(self) | __nonzero__(self) |
| __hash__(self) | |
| __getattr__(self, name) | |
| __setattr__(self, name, attr) | |
| __delattr__(self, name) | |
| __getattr__(self, name) | |
| __call__(self, args, kwargs) | |

Python Time Methods

| | |
|-------------|-------------|
| replace() | utcoffset() |
| isoformat() | dst() |
| __str__() | tzname() |

strftime(*format*)