

### File operations

<code>f = open(path)</code>	open file
<code>f.read()</code>	read file
<code>f.readline()</code>	read line from file
<code>f.readlines()</code>	return list of lines of file
<code>f.write(s)</code>	write s to file
<code>f.close()</code>	close file
<code>with open(path) as name: statements</code>	open and after statements close file

### Logic and Math Operators

<code>x + y</code>	addition
<code>x - y</code>	subtraction
<code>x * y</code>	multiplication
<code>x / y</code>	division
<code>x // y</code>	floor
<code>x ** y</code>	exponent
<code>x % y</code>	modulo
<code>x == y</code>	equal
<code>x != y</code>	not equal
<code>x &gt; y</code>	greater than
<code>x &gt;= y</code>	greater than or equal
<code>x &lt; y</code>	less than
<code>x &lt;= y</code>	less than or equal

### Converting Data Types

<code>str(num)</code>	number to string
<code>str(txt, encoding)</code>	encoded bytes to string
<code>int("st r", base = 10)</code>	number string to integer
<code>hex(int)</code>	integer to hex string

### Converting Data Types (cont)

<code>bin(int)</code>	integer to binary string
<code>int(float)</code>	float to integer
<code>float(int or str)</code>	integer / string to float
<code>ord(str)</code>	string to ASCII
<code>chr(int)</code>	integer to ASCII

### Base Functions

<code>int(), float(), str(), bool() ...</code>	type casting
<code>len(data)</code>	return length of data
<code>min(value), max(value)</code>	minimum maximum
<code>pow(x, y, [z])</code>	x to the power y [mod z]
<code>range(start, stop, [step])</code>	range of
<code>input(), print()</code>	console input console output
<code>filter(function, iterable)</code>	filter iterable
<code>id(object)</code>	unique object ID
<code>map(function, iterable)</code>	map function onto iterable
<code>round(n, [x])</code>	round n to x decimal places

[\_] optional argument

### Module Import

```
import module as name

from module import submodule as name
```

C

By **monika.g**  
[cheatography.com/monika-g/](https://cheatography.com/monika-g/)

Not published yet.  
Last updated 6th February, 2023.  
Page 1 of 2.

Sponsored by **Readable.com**  
Measure your website readability!  
<https://readable.com>

### Exception Handling

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

### Dictionaries

<code>d = {}</code>	create dictionary
<code>d = { key : val }</code>	create dictionary with values
<code>d[key] = val</code>	assign a value
<code>d[key],</code> <code>d.get(key)</code>	access value at key
<code>d.keys()</code>	iterable view of keys
<code>d.values()</code>	iterable view of values
<code>d.items()</code>	iterable view of (key, value) tuples
<code>d.clear()</code>	clear dictionary
<code>key in d</code>	determine if key exists

### Lists

<code>n = []</code>	create list
<code>n[index] = val</code>	assign value at index
<code>n[index]</code>	access value at index
<code>n.append(val)</code>	add to list
<code>n.insert( position, val)</code>	insert into list
<code>n.count(val)</code>	count occurness
<code>n.remove(val)</code>	remove item
<code>n.pop(val)</code>	remove an return item
<code>n.reverse()</code>	reverse n
<code>n.sort()</code>	sort n

### Slicing and Indexing

<code>x[start:stop:step]</code>	<code>x = [4, 8, 9, 3, 0]</code>
<code>x[0]</code>	4
<code>x[-1]</code>	0
<code>x[:3]</code>	[4, 8, 9]
<code>x[3:]</code>	[3, 0]
<code>x[:-2]</code>	[4, 8, 9]
<code>x[: :2]</code>	[4, 9, 0]
<code>x[: : -1]</code>	[0, 3, 9, 8, 4]

