

Variable Types

10	Int
10.0	Float
10 + 2j	Complex
"string" or 'string'	String
(1, 2, 3)	Tuple
[1, 2, 3]	List
{'a': 1, 'b': 2, 'c': 3}	Dictionary
{1, 4, 5}	Set
True and False	Booleans

Basic Functions

print(s)	Prints the string <i>s</i> to the screen
input(msg)	Prints the <i>msg</i> to the screen, waits for user keyboard input and returns the input as string
len(x)	Returns the number of elements of the sequence <i>x</i>
help(x)	Returns the help (docstring) of a module or function
eval(x)	Evaluates a string and executes it as a line of code
exec(x)	Evaluates a block of code in a string or a code object
max(x)	Returns the greatest value in a sequence
min(x)	Returns the smallest value in a sequence
type(x)	Returns the variable type of <i>x</i>

Variable Assignment and Operations

a = 10	Assigns 10 to a
b = 5	Assigns 5 to b
c = a + b	Evaluates a + b and assigns it to c
print(c)	Prints the value of c (15)

If Statements

```
a = 5
n = 6
if a > n**3:
    print("Is much greater")
elif a > n:
    print("Is greater")
else:
    print("Is lower")
```

Is lower

For Loop

```
a = 0
for i in range(1, 101):
    a += i
print(a)
```

5050

While Loop

```
a = 0
i = 1
while i < 101:
    a += i
    i += 1
print(a)
```

5050

Function

```
def double(x):
    return 2*x
print(double(3))
```

6

Function with Default Arguments

```
def multiply(a, b=1):
    return a * b
x1 = multiply(3, 4)
x2 = multiply(5)
print(f"x1 = {x1}")
print(f"x2 = {x2}")
```

x1 = 12
x2 = 5

Function - Keyword Arguments

```
def kword_func(a, b, c):
    print(f"a = {a}")
    print(f"b = {b}")
    print(f"c = {c}")
kword_func(c=10, a=5, b=2)
```

a = 5
b = 2
c = 10

Lambda Function

```
f = lambda x, y: x2 + x*y + y2
print(f(2, 3))
```

19

List Comprehension

```
lc = [i**2 for i in range(6)]
print(lc)
```

[0, 1, 4, 9, 16, 25]

Dictionary Comprehension

```
from math import factorial
dc = {i:factorial(i) for i in range(7)}
print(dc)
```

{0: 1, 1: 1, 2: 2, 3: 6, 4: 24, 5: 120, 6: 720}

Python Keywords

if	Starts an if statement definition
elif	Tests another condition if the previous isn't <i>True</i>
else	Executes the block of code if all previous tests are <i>False</i>
and	Returns <i>True</i> only if both statements are <i>True</i>
or	Returns <i>True</i> if at least one statement is <i>True</i>
not	Returns the opposite Boolean
in (on if statements)	Checks if an element is in a sequence
is	Checks if two elements are equal to each other
assert	Checks a logic test and raises an error message if the test fails
while	Starts a while loop definition
for	Starts a for loop definition
in (on for loop statements)	Takes each element of a sequence, one per iteration
continue	Skips to the next loop iteration
break	Exits loop statement
def	Starts a function definition
lambda	Starts a lambda function definition

Python Keywords (cont)

return	Exits function and returns a value
yield	Used to create generators. Returns a value but allows to resume the function
class	Starts a class definition
pass	Does nothing. Used where a block of text is need syntactically
from	From a module or library...
import	Imports a module or function...
as	With this name...
try	Starts an exception handling statement
except	Catches an exception
finally	Executes at the end of the exception handling statement
raise	Throws an error
global	Make a variable global
local	Make a variable local
with	Wraps code block with methods defined by a context manager
del	Delete a variable from memory

Write File

```
a = [1, 3, 2, 5, 6, 2, 4]
b = [3, 5, 1, 2, 7, 7, 8]
with open('data.csv', 'w') as file:
    for i, j in zip(a, b):
        file.write(f"{i}, {j}\n")
```

This code writes a csv file based on the data in *a* and *b*

Read File

```
with open('data.csv', 'r') as file:
    for line in file:
        print(line)
```

This code reads the file and prints it line by line.