

Data Type

Integer	-100, 0, -100
Float	-100.98, 0.0001, 90.00
String	'Python', '400', '100+200', 'True'
Boolean	True, False

Operators

Numeric		Comparison	
+	Addition	==	Equal
-	Subtraction	!=	Different
*	Multiplication	>	Higher
/	Division	<	Lower
**	Exponent	>=	Higher or Equal
%	Modulus	<=	Lower or Equal
//	Floor Division		

Boolean		String	
&	Logical AND	+	Concatenation
	Logical OR		
not	Logical NOT		

Assignment Statement & Expression

Expression is a combination of values, variables, and operators

222 'text'

25+36 'App' + 'le'

Assignment Statement links a variable name on the left hand side of the operator, with an expression on the right hand side.

a = 222 b = 'text'

c = 25 + 36 d = 'App' + 'le'

a = a + 1

List Operations

Create a List	L1 = [1, 2, 3, 4, 5, 6]
Get the first element	L1[0]
Get the last element	L1[-1]
Count List elements	len()
Insert an element	insert()
Insert an element to the end	append()
Sort all elements	sort()
Remove an element	pop()
Convert object to List	list()

Slicing

Slicing Expression

List Name[start index : stop index : step size]

(Step size is optional)

Examples

2nd - 5th elements L1[1:5]

2nd - Last elements L1[1:]

1st - 3rd elements L1[0:3]

All alternate elements L1[::2]

Dictionary (Dict.) Operations

Create a Dict. D1 = {"Andrew":18, "Johnson":23, "Olivia":22}

Create a Dict. dict(zip())
from two Lists

Access Dict. D1["Andrew"]
value

Update Dict. D1["Andrew"] = 20
value

Add an element D1["Sue"] = 25

Drop an element del D1["Johnson"]

Count Dict. elements len()

Return all keys keys()

Return all values values()

Tuple Operations

Create a Tuple	T1 = (1, 2, 3, 4, 5, 6)
Convert List to Tuple	tuple()

Note: Tuple elements are immutable and cannot be changed via operations.

Set Operations

Create a Set	S1 = {1, 2, 3, 4, 5}
Insert an element	add()
Find unique elements	set()
Create a Set with all elements from 2 Sets	union()
Create a Set with common elements from 2 Sets	intersection()

List Comprehension

newlist = [expression for variable in sequence if condition]

Conditional Statements

```
if Condition 1:
    Code Block 1
elif Condition 2:
    Code Block 2
elif Condition 3:
    Code Block 3
else:
    Code Block 4
```

for Loops

```
for <variable> in <sequence>:
    Code Block

Example:
for x in range(0,5):
    print(x)
```

while Loops

```
while <expression>:  
    Code Block
```

Example:

```
i = 2  
while i <= 10:  
    print(i)  
    i = i + 3
```

Nested loop

```
for iterating_var in sequence:  
    for iterating_var in  
sequence:  
        Code Block1  
        Code Block2  
while <expression>:  
    while <expression>:  
        Code Block1  
    Code Block2
```

Statements used with Loops

break	Terminate the whole loop
continue	Stop the current iteration of the loop, and continue with the next. Loop does not terminate.
pass	Do nothing and continue the rest of the code inside a loop for the current iteration



By **ocivv**
cheatography.com/ocivv/

Published 24th February, 2023.
Last updated 12th January, 2023.
Page 2 of 2.

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