Python Cheat Sheet

For Beginners & Lazy Experts (1/2)

Data Types

Text Type	`str`	`'I'm a string'`
Numeric Types	`int`	`10`
	`float`	`10.3`
	`complex`	`2 - 3j`
Boolean Type	`bool`	`True, False`
Sequence Types	`list`	`[1, 2, 'a', 'b']`
	`tuple`	`(1, 2, 3)`
	`range`	`range(4)`
Set type	`set`	`{1, 2, 3}`
Mapping type	`dict`	`{1:'a', 2:'b', 3:'c'}`

Data Type Conversions

Integer and Float Conversions

```
>>> float(2 + 3)      5.0      >>> int(2.0 + 3.0)     5
```

Real to Complex Data Type Conversion

```
>>> complex(2, 3.4)   (2+3.4j)   >>> complex(5)       (5j+0)
```

Data Type Conversion with Strings

```
>>> int('123')        123       >>> int(456.78)       456
>>> str(5)            '5'       >>> str(123.45)       '123.45'
>>> str(1+2j)         '(1+2j)'
```

Type Conversion to Tuples and Lists

```
>>> tuple([1, 2, 3])  (1, 2, 3)   >>> list((1, 2, 3))   [1, 2, 3]
>>> tuple('AB')       ('A','B')    >>> list('AB')       ['A','B']
```

Type Conversion to Dictionaries and Sets

```
>>> set(['a', 1, 1, 'b', 2])       {'a', 1, 2, 'b'}
>>> dict((('a', 1),('b', 2)))      {'a': 1, 'b': 2}
```

Convert Binary to Decimal

```
>>> bin(12)           '0b1100'     >>> int(0b11)         3
```

Convert Hexadecimal to Decimal

```
>>> hex(12)           '0xc'        >>> int(0x4f)         79
```

Convert Text to Decimal

```
>>> ord('a')          97           >>> chr(65)           'A'
```

Booleans

Booleans as Numbers

```
>>> True == 1    True          >>> False == 0    True
```

Comparison Operators

`a == b`	is *a* equal to *b*?	`a != b`	is *a* different to *b*?
`a < b`	is *a* less than *b*?	`a <= b`	is *a* less than or equal to *b*?
`a > b`	is *a* greater than *b*?	`a >= b`	is *a* greater or equal to *b*?

Membership and Identity Operators

`a in b`	is *a* in *b*?	`a is b`	are *a* and *b* the same object?
`a not in b`	is *a* not in *b*?	`a is not b`	are *a* and *b* different objects?

Boolean Operators

`not`	returns False if operand is True, True otherwise
`and`	returns True if both operands are True, False otherwise
`or`	returns False if both operands are False, True otherwise

Operator Precedence

`()`	parentheses are evaluated first
`**`	exponent
`+, -`	unary + and - signs
`*, /, //, %`	multiplication, divisions, and modulo
`+, -`	addition and substraction
`==, !=, <, <=, >=, >, is, is not, in, not in`	comparison, identity, and membership operators
`not`	logical NOT
`and`	logical AND
`or`	logical OR

Print Function

```
print('a', 'b', sep='*')                    a*b
print('c', 'd', sep='?', end='\\')          c?d\e f
print('e', 'f')
```

User Input

```
name = input("Enter your name: ")    >>> Enter your name: Promethee
print("Your name is: " + name)        Your name is: Promethee
```

Decision Structure

```
if n == 0:
    print("n is zero")
elif n > 0:
    print("n is strictly positive")
else: # n < 0
    print("n is strictly negative")
```

Repetition Structures

```
n = 0                    0          for i in range(4):     0
while n < 4              1              print(i)           1
    print(n)             2          print("i =", i)       2
    n += 1               3                                 3
print("n =", n)          n = 4                             i = 3
```

Exceptions

```
try:
    # run this code
except NameOfErrorType1:
    # handle error type 1
except NameOfErrorType2:
    # handle error type 2
except:
    # handle any other error
else:
    # run this code if no error
finally:
    # always run this code
```

Built-in Exceptions

```
FileNotFoundError
IndexError
KeyError
ModuleNotFoundError
NameError
SyntaxError
TypeError
ValueError
ZeroDivisionError
```

Modules

```
>>> import random
>>> from math import pi
>>> print(random.randint(0,9))      2
>>> print(pi)                       3.141592653589793
```

Files

`open()`	returns a file object
`close()`	close the file
`read()`	returns the file content
`readline()`	returns one line from the file
`readlines()`	returns a list of lines from the file

Access modes

`r`	read
`w`	write
`a`	append
`x`	create

Strings

String Delimiters

Single quotes	`'I am a string'`
Double quotes	`"I'm a string"`
Triple single quotes	`'''I'm a string'''`
Triple double quotes	`"""I'm a string"""`

Escape Sequences

Backslash (\)	`\\`	ASCII Linefeed (LF)	`\n`
Single quote (')	`\'`	ASCII Carriage Return (CR)	`\r`
Double quote (")	`\"`	ASCII Horizontal Tab (TAB)	`\t`

String Operations

Concatenation	`>>> 'ABC' + '3'`	`'ABC3'`
Repetition	`>>> 'ABC' * 3`	`'ABCABCABC'`

String Length

```
>>> len('0123456789')   10        >>> len('\n')          1
>>> len('')             0
```

Unicode Code/Text Conversion

```
>>> ord('a')    97          >>> chr(97)     'a'
>>> ord('A')    65          >>> chr(65)     'A'
>>> ord('\n')   10          >>> chr(9)      '\t'
```

String Membership

```
>>> 'A' in 'ABC'          True
>>> 'BC' not in 'ABC'     False
```

String Indexing

```
>>> len('ABCDEF123456')    12
>>> 'ABCDEF123456'[0]      'A'    >>> 'ABCDEF123456'[-1]     '6'
>>> 'ABCDEF123456'[11]     '6'    >>> 'ABCDEF123456'[-12]    'A'
```

String Slicing

```
>>> 'ABCD1234'[:4]          >>> 'ABCD1234'[:-4]      'ABCD'
>>> 'ABCD1234'[4:]          >>> 'ABCD1234'[-4:]      '1234'
>>> 'ABCD1234'[2:6]         >>> 'ABCD1234'[-6:-2]    'CD12'
>>> 'ABCD1234'[2:-2]        >>> 'ABCD1234'[-6:6]     'CD12'
```

String Slicing Steps

```
>>> '0123456789'[0:10:2]    >>> '0123456789'[::2]    '02468'
>>> '0123456789'[1:10:2]    >>> '0123456789'[1::2]   '13579'
>>> '0123456789'[0:10:3]    >>> '0123456789'[::3]    '0369'
>>> '0123456789'[9:0:-2]    >>> '0123456789'[::-2]   '97531'
>>> '0123456789'[-1:-10:-2] >>> '0123456789'[::-2]   '97531'
>>> '0123456789'[-1:-11:-3] >>> '0123456789'[::-3]   '9630'
```

String Methods

```
>>> "abc".upper()          'ABC'   >>> "abc".islower()          True
>>> "AAa".count('A')       2       >>> "A-".endswith('-')      True
>>> "abcb".find('d')       3       >>> "abcb".index('d')       3
>>> "abc".find('d')        -1      >>> "abc".index('d')        ValueError
>>> "a1".isalnum()         True    >>> "aA".isalpha()          True
>>> "123".isdecimal()      True    >>> "123".isdigit()         True
>>> "⅛".isnumeric()        True    >>> "3.4".isnumeric()       False
>>> "-A-".strip('-')       'A'     >>> "-A-".rstrip('-')       -A
>>> "A-B".split('-')       [A,B]   >>> "A1A".replace('A','0') '010'
```

Iterating over Strings

```
for e in 'ABC123':                      A>B>C>1>2>3>
    print(e, end='>')
for i in range(len('ABC123')):          A*B*C*1*2*3*
    print('ABC123'[i], end='*')
```

Python Cheat Sheet

For Beginners & Lazy Experts (2/2)

Lists

List Creation

```
>>> l = []; l                [] >>> l = list(); l          []
>>> l = list("abc"); l          ['a', 'b', 'c']
```

List Comprehension

```
>>> [x for x in range(10) if x % 2]          [1, 3, 5, 7, 9]
>>> [c for c in "ABCDE" if c not in "ACE"]   ['B', 'D']
```

List Indexing

```
>>> [1, 2, 4][0]                             1
>>> [1, 2, 4][-1]                            4
>>> [[1,2],[3,4],[5,6]][1]                   [3,4]
>>> [[1,2],[3,4],[5,6]][1][1]                4
```

List Slicing

```
>>> [0, 1, 2, 3, 4][1:3]                     [1, 2]
>>> [0, 1, 2, 3, 4][:2]                      [0, 1]
>>> [0, 1, 2, 3, 4][::2]                     [0, 2, 4]
>>> l = [0, 1, 2, 3, 4]; x = slice(1, 4); l[x]   [1, 2, 3]
```

List Length

```
>>> len([0,'a', 1])      3      >>> len([])          0
```

List Membership

```
>>> 0 in [0, 'a', 1]     True   >>> '0' in [0, 'a', 1]     False
```

List Comparison

```
>>> [0, 'a', 2] == [0, 'a', 2]               True
>>> [0, 'a', 2] > [0, 'b', 1]                False
>>> ['a', 1, 2] > ['a', 1]                   True
```

List Manipulation

```
>>> l = [1, 2, 3]
>>> l[0] = '1'; l                        ['1', 2, 3]
```

Concatenation and Repetition

```
>>> l = [2]
>>> l += [3]; l                          [2, 3]
>>> l *= 3; l                            [2, 3, 2, 3, 2, 3]
```

Adding Elements

```
>>> l = [2, 3]; l.append(1); l           [2, 3, 1]
>>> l = [2, 3]; l.extend('bc'); l        [2, 3, 'b', 'c']
>>> l = [2, 3]; l.insert(4, 'a'); l      [2, 3, 'a']
```

Removing Elements

```
>>> l = [1, 2, 4, 3, 4, 5]
>>> l.remove(2); l                       [1, 4, 3, 4, 5]
>>> l.pop()                              5 # l = [1, 4, 3, 4]
>>> l.pop(2)                             3 # l = [1, 4, 4]
```

Deleting Elements

```
>>> l = [1, 2, 3, 'a', 5]
>>> del l[3]; l                          [1, 2, 3, 5]
```

List Counting, Searching, and Sorting

```
>>> [3, 4, 1, 3].count(3)                2
>>> [3, 4, 1, 3].index(3)                0
>>> l = [3, 4, 1, 3]; l.reverse(); l     [3, 1, 4, 3]
>>> l = [3, 4, 1, 3]; l.sort(); l        [1, 3, 3, 4]
>>> l = [(1, 'a'), (0, 'b')]
>>> l.sort(key=lambda x:x[1], reverse = True); l
    [(0, 'b'), (1, 'a')]
```

String-to-List and Back

```
>>> "a-bb-ccc".split('-')                ['a', 'bb', 'ccc']
>>> "-".join(['0', '11', '222'])         '0-11-222'
```

List Built-in Functions

```
>>> all([True, True])     True   >>> any([True, False])    True
>>> len([0, 1, 2])        3      >>> list(("ab"))          ['a', 'b']
>>> max([0, 1, 2])        2      >>> min([0, 1, 2])        0
>>> list(reversed([1, 0, 2]))          [2, 0, 1]
>>> sorted([1, 0, 3, 2])               [0, 1, 2, 3]
>>> sum([1, 0, 2])                     3
>>> tuple([1, 0, 3, 2])                (1, 0, 3, 2))
>>> list(zip([1, 0],['b', 'a']))       [(1, 'b'), (0, 'a')]
```

Iterating over Lists

```
for e in [0, 1, 'a']:                  0>1>'a'>
    print(e, end='>')
for i in len([0, 1, 'a']):             0-1-a-
    print([0, 1, 'a'][i], end='-')
for i, a in enumerate([0, 1, 'a']):    0*0/1*1/2*a/
    print(i, a, sep='*', end='/')
```

Tuples

Tuple Creation

```
>>> t = (); t             ()     >>> t = tuple(); t        ()
>>> tuple([0, 0, 'a', 1])        (0, 0, 'a', 1)
```

Tuple Unpacking

```
>>> (x, y) = (1, 2); x    1      >>> x, y = 1, 2; y       2
```

Tuple Built-in Functions

```
all(), any(), count(), enumerate(), filter(), index(), len(),
list(), map(), max(), min(), next(), reversed(), slice(), sum(),
sorted(), tuple(), zip()
```

Tuple Operations

Indexing	>>> (1, 2, 3, 4)[1]	2
Slicing	>>> (1, 2, 3, 4)[1:3]	(2, 3)
Concatenation	>>> (1, 2) + ('a',)	(1, 2, 'a')
Repetition	>>> (1, 2) * 3	(1, 2, 1, 2, 1, 2)
Membership	>>> 1 in (1, 2)	True

Iterating over Tuples

```
for i in range(len(('A', 1, 'B', 2))):     A*1*B*2*
    print(('A', 1, 'B', 2)[i], end='*')
```

Sets

Set Creation

```
>>> s = set(); s          set()  >>> s = set('ab'); s     {'a', 'b'}
>>> s = set([0, 0, 'a', 1]); s   {0, 1, 'a'}
```

Adding Elements

```
>>> s = {1, 2, 3}; s.add(4); s              {1, 2, 3, 4}
>>> s = {'a','b'}; s.add('cd'); s           {'a','b','cd'}
>>> s = {1, 2, 3}; s.update([3, 4]); s      {1, 2, 3, 4}
>>> s = {'a','b','c'}; s.update('cd'); s    {'a','b','c','d'}
```

Removing Elements

```
>>> s = {'a', 'b', 'c'}; s.remove('a'); s   {'b', 'c'}
>>> s = {'b', 'c'}; s.discard('a'); s       {'b', 'c'}
>>> s = {'b', 'c'}; s.remove('a'); s        KeyError: 'a'
```

Set Operations

Union	{1, 2, 3} | {2, 3, 4}	{1, 2, 3, 4}
Intersection	{1, 2, 3} & {2, 3, 4}	{2, 3}
Difference	{1, 2, 3} - {2, 3, 4}	{1}
Symmetric difference	{1, 2, 3} ^ {2, 3, 4}	{1, 4}
Subset	{1, 2, 3} > {1, 2}	True
Superset	{1, 2, 3} < {1, 2}	False

Iterating over Sets

```
for e in {0, 'a', 1}:                  0>1>'a'>
    print(s, end='>')
```

Dictionaries

Dictionary Creation

```
>>> d = dict(); d                      {}
>>> d = {'x':0, 'y':1}; d              {'x': 0, 'y': 1}
>>> d = dict(x = 0, y = 1); d          {'x': 0, 'y': 1}
```

Dictionary Length

```
>>> len({0:'a', 1:'b'})                2
```

Key Membership

```
>>> d = {0:'a', 1:'b'}
>>> 0 in d                             True
```

Dictionary Manipulation

Retrieving a Value Given a Key

```
>>> d = {0:'a', 1:'b'}
>>> d.get(0)                           'a'
>>> d.get(2)
>>> d.get(2, None)                     None
```

Adding a Key/Value Pair

```
>>> d[2]='c'; d                        {0:'a', 1:'b', 2:'c'}
```

Updating a Key/Value Pair

```
>>> d[2]='e'; d                        {0:'a', 1:'b', 2:'e'}
```

Deleting a Key/Value Pair

```
>>> del d[2]; d                        {0:'a', 1:'b'}
```

Dictionary Methods

```
>>> d = {0:'a', 1:'b', 2:'c'}
>>> d.get(0)                           'a'
>>> d.get(3, None)                     None
>>> d.items()                          [(0, 'a'), (1, 'b'), (2, 'c')]
>>> d.keys()                           [0, 1, 2]
>>> d.values()                         ['a', 'b', 'c']
>>> d.pop(2)                           'c' # d = {0:'a', 1:'b'}
>>> d[3] = 'd'; d                      {0:'a', 1:'b', 3:'d'}
>>> d.popitem()                        (3, 'd') # d = {0:'a', 1:'b'}
>>> d.update({3:'e'}); d               {0:'a', 1:'b', 3:'e'}
>>> d.clear(); d                       d = {}
```

Iterating over Dictionaries

```
>>> d = {0:'a', 1:'b', 2:'c'}
for key in d:                          0>1>2>
    print(key, end='>')
for key in d.keys():                   0*1*2*
    print(key, end='*')
for item in d.items():                 (0, 'a')
    print(item)                        (1, 'b')
                                       (2, 'c')
for k, v in d.items():                 0, 'a'
    print(k, v)                        1, 'b'
                                       2, 'c'
```

Similarities & Differences

	List	Tuple	Set	Dictionary
Ordered	Yes	Yes	No	Yes (after Python 3.7)
Duplicates	Yes	Yes	No	No (duplicate keys)
Mutable	Yes	Yes	No	Yes
Comparison	Yes	Yes	Yes	No
Access elements	Index	Index	Index	Key
Slicing	Yes	Yes	No	No
Concatenation	Yes	Yes	No	No
Repetition	Yes	Yes	No	No
Iteration	Yes	Yes	Yes	Yes