PYTHON CHEATSHEET

Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators

Operator	Example	Same as
=	c= 5	c = 5
+=	m += 3	m = m + 3
-=	m -= 3	m = m - 3
*=	m *= 3	m = m * 3
/=	m /= 3	m = m / 3
%=	m %= 3	m = m % 3
**=	m **= 3	m = m ** 3
//=	m //= 3	m = m // 3

Comparison Operators

Operator	Name	Example
==	Equal	a == b
!=	Not equal	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

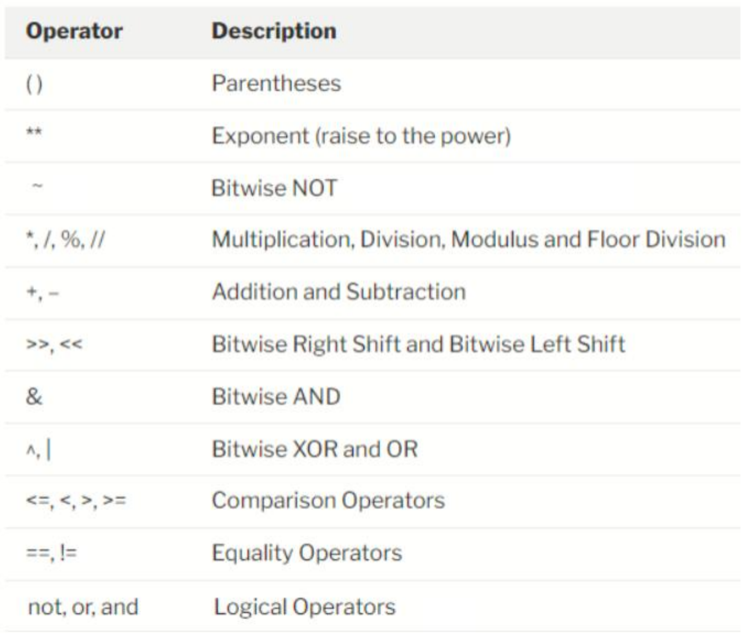
Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	a < 5 and b< 10
Or	Returns True if one of the statements is true	a < 5 or b < 4
not	Reverse the result, returns False if the result is true	not(a < 5 and b< 10)

Bitwise Operators

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	(a & b) =12 (means 0000 1100)
|	OR	Sets each bit to 1 if one of two bits is 1	(a | b) = 61 (means 0011 1101)
^	XOR	Sets each bit to 1 if odd number of bits are 1	(a^b) = 49 (means 110001)
~	NOT	Inverts all the bits	(~a) = -61 (means 1100 0011 in 2's complement)

Operators Precedence



Operator	Description
( )	Parentheses
**	Exponent (raise to the power)
~	Bitwise NOT
*, /, %, //	Multiplication, Division, Modulus and Floor Division
+, –	Addition and Subtraction
>>, <<	Bitwise Right Shift and Bitwise Left Shift
&	Bitwise AND
^, |	Bitwise XOR and OR
<=, <, >, >=	Comparison Operators
==, !=	Equality Operators
not, or, and	Logical Operators

Input() function

- name = input("Enter your name: ") ← String by default
- age = int(input("Enter your age: ")) ← data type is saved as an integer value

Control structures

- Sequence
- Selection – if, if-else, if-elif -else
- Iteration – while, for

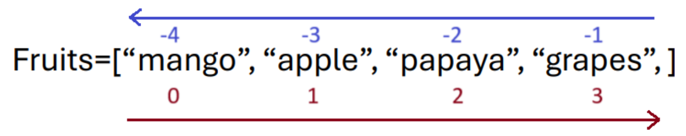
Break statement

```
for a in range(10):
    If a==6:
        break
    print(a)
```

Continue statement

```
for w in range(10,0,-1):
    if w==8:
        continue
    print(w)
```

Lists



Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Get a list as input from user

```
# For list of integers
list1 = []
list1 = [int(item) for item in input("Enter the list items : ").split()]
or
inlist1=input(“Enter the list items: ”)
list1 = [int(item for item inlist1.split())]

# For list of strings/chars
list2 = []
list2 = [item for item in input("Enter the list items : ").split()]
print(list2)
```

Tuples newtuple=(11,22,33,44,55)

methods	example	description

a.index(tuple)	>>> a=(1,2,3,4,5)
	>>> a.index(5)
	4
		Returns the index of the first matched item.

a.count(tuple)	>>>a=(1,2,3,4,5) >>> a.count(3) 1	Returns the count of the given element.
len(tuple)	>>> len(a) 5	return the length of the tuple
min(tuple)	>>> min(a) 1	return the minimum element in a tuple
max(tuple)	>>> max(a) 5	return the maximum element in a tuple
del(tuple)	>>> del(a)	Delete the entire tuple.

Sets newset = {1, 2, 3, 4}

add()	insert an element into the set
remove()	remove an element
len()	Get the number of elements in a set
clear()	Remove all elements from a set.
pop()	Remove and return a random element from the set.
copy()	Return a copy of the set.

Dictionaries Marks = {“Ann”: 45, “Bob”: 60, “Sam”: 83}

Method	Description
clear()	Removes all the elements from the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing the tuple for each key-value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Strings

Method	Example	Output

The upper() method returns the string in upper case:	a = "Hello, World!"
	print(a.upper())
		HELLO, WORLD!

The lower() method returns the string in lower case:	a = "Hello, World!"
	print(a.lower())
		hello, world!

The strip() method removes any whitespace from the beginning or the end:	a = " Hello, World! " print(a.strip()) # returns "Hello, World!"	Hello, World!

The replace() method replaces a string with another string:	a = "Hello, World!"
	print(a.replace("H", "J"))
		Jello, World!

The split() method splits the string into substrings if it finds instances of the separator:	a = "Hello, World!" print(a.split(",")) # returns ['Hello', ' World!']	['Hello', ' World!']

Slicing –

>>> my_list[1:3] – extracts a sub list from index 1 up to 3 but not 3.

>>> my_list[1:] - extracts a sub list starting from index 1 to the end.

>>> my_list[:3] – from the beginning up to 3 but not 3.

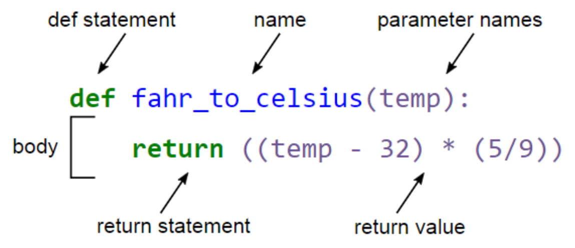
>>> my_list[:] - extracts the entire list

Subset Lists of sub lists

>>> my_list2[1][0] - Accesses the index 0 of the sub list extracted

>>> my_list2[1][:2] - Accesses up to 2 but not 2 of the sub list extracted

Functions



```
def fahr_to_celsius(temp):
    return ((temp - 32) * (5/9))
```

Scope of variables

```
x = 300 #Global

def myfunc():
  x = 200 #Local
  print(x)
```

File Handling

There are four different methods (modes) for opening a file:

"r"	- Read	- Default value. Opens a file for reading, error if the file does not exist
"a"	- Append	- Opens a file for appending, creates the file if it does not exist
"w"	- Write	- Opens a file for writing, creates the file if it does not exist
"x"	- Create	- Creates the specified file, returns an error if the file exists

Basic structure

```
f=open("demo1.txt", "r")
....
....
f.close()
```

File read

f = open("demo1.txt", "r") → opens the file in read mode
print(f.read()) → reads and prints the entire content of the file
print(f.read(5)) → prints the first 5 characters in a file
print(f.readlines()) → reads all lines in a file, to a list
print(f.readline()) → prints the next line(one line per command)

File Write

- "a" - Append - will append to the end of the file.
- "w" - Write - will overwrite any existing content.

Create a New File

✓ "x" - Create - will create a file, returns an error if the file exist
✓ "a" - Append - will create a file if the specified file doesn't exist
✓ "w" - Write - will create a file if the specified file doesn't exist

---

Delete a file

Remove the file "n1.txt":

```
import os
os.remove("n1.txt")
```

Additional notes:

Comments - Comments starts with a #, and Python will ignore them.
Variables - Variables are containers for storing data values.

Rules for Python variables:

- ✓ A variable name must start with a letter or the underscore character
- ✓ A variable name cannot start with a number
- ✓ A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- ✓ Variable names are case-sensitive (age, Age and AGE are three different variables)
- ✓ Python keywords can't be used

Python with MySQL

Create connection

```
import mysql.connector

# Establishing the connection
mydb = mysql.connector.connect(
   host='localhost',     # Your host, usually localhost
   user='yourusername',   # Your MySQL username
   password='yourpassword', # Your MySQL password
   database='yourdatabase'  # Name of the database to connect to
)

# Creating a cursor object using the cursor() method
mycursor = mydb.cursor()

# Closing the connection
mycursor.close()
mydb.close()
```

Create a database named 'school'

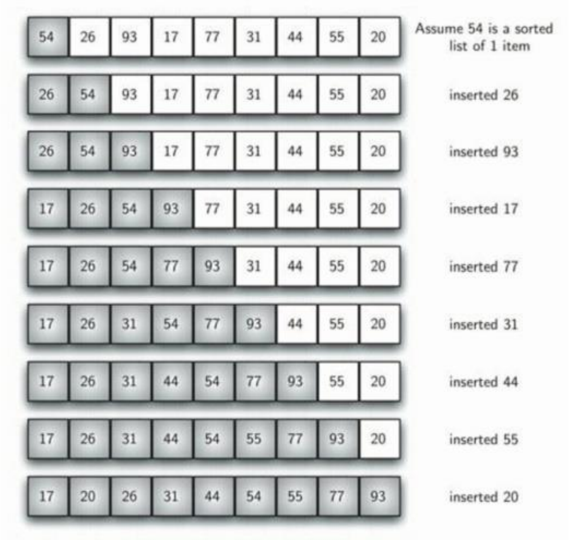
```
mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE
school ")
```

Creating a table

```
mycursor.execute(
   "CREATE TABLE student ( \
   regNo INT PRIMARY KEY, \
   name VARCHAR(100), \
   address VARCHAR(255), \
   contactNo VARCHAR(15) )"
)
```

Bubble sort method (for Sorting list in ascending order)


54 26 93 17 77 31 44 55 20 Assume 54 is a sorted list of 1 item
26 54 93 17 77 31 44 55 20 inserted 26
26 54 93 17 77 31 44 55 20 inserted 93
17 26 54 93 77 31 44 55 20 inserted 17
17 26 54 77 93 31 44 55 20 inserted 77
17 26 31 54 77 93 44 55 20 inserted 31
17 26 31 44 54 77 93 55 20 inserted 44
17 26 31 44 54 55 77 93 20 inserted 55
17 20 26 31 44 54 55 77 93 inserted 20