1. BASIC OPERATORS IN PYTHON

ARITHMETIC OPERATORS	DESCRIPTION	SYNTAX
+	Adds two operands	x+y
-	Subtracts two operands	x-y
*	Multiplies two operands	x*y
/	Division(float): divides the first operand by second	x/y
//	Division(floor): divides the first operand by second	x//y
**	Exponential of first operand raised to second	x**y
%	Modulus: returns the remainder when the first operand is divided by the second	x%y

RELATIONAL OPERATORS	DESCRIPTION	SYNTAX
>	Greater than: True if left operand is greater than the right	x > y
<	Less than: True if left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

LOGICAL OPERATORS	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

BITWISE OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift (prints bitwise right shift operation)	x>>
<<	Bitwise left shift (prints bitwise right shift operation)	x<<

ASSIGNMENT OPERATORS	DESCRIPTION	SYNTAX
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand	a%=b a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b a=a&b
|=	Performs Bitwise OR on operands and assign value to left operand	a|=b a=a|b
^=	Performs Bitwise XOR on operands and assign value to left operand	a^=b a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b a=a>>b
<<=	Performs Bitwise left shift on operands and assign value to left operand	a <<= b a= a << b

2. DATA TYPES

EXAMPLE	DATA TYPE
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray

3. STRING OPERATIONS

FUNCTION	DESCRIPTION
mystring[:N]	Extract N number of characters from start of string.
mystring[-N:]	Extract N number of characters from end of string
mystring[X:Y]	Extract characters from middle of string, starting from X position and ends with Y
str.split(sep=' ')	Split Strings
str.replace(old_substring, new_substring)	Replace a part of text with different substring
str.lower()	Convert characters to lowercase
str.upper()	Convert characters to uppercase
str.contains('pattern', case=False)	Check if pattern matches (Pandas Function)
str.extract(regular_expression)	Return matched values (Pandas Function)
str.count('sub_string')	Count occurence of pattern in string
str.find( )	Return position of sub-string or pattern
str.isalnum()	Check whether string consists of only alphanumeric characters
str.islower()	Check whether characters are all lower case
str.isupper()	Check whether characters are all upper case
str.isnumeric()	Check whether string consists of only numeric characters
str.isspace()	Check whether string consists of only whitespace characters
len( )	Calculate length of string
cat( )	Concatenate Strings (Pandas Function)
separator.join(str)	Concatenate Strings

4. BUILT-IN FUNCTIONS

FUNCTION	DESCRIPTION
abs(x)	Returns absolute value of a number
ascii()	Returns ASCII value of a string
bin()	Returns binary representation of an integer
bool()	Converts value to boolean (True / False)
complex()	Used to convert numbers or string into a complex number
dict()	A constructor which creates a dictionary
dir()	Returns a list of names in the current local space
float()	Returns a floating point number from a number or string
format()	Returns a formatted representation of a given value
frozenset()	Returns an immutable frozenset object
globals()	Returns a dictionary containing the variables defined in the global namespace
help()	Used to get help related to the object passed during the call
input()	Used to get input from the user
int()	Returns the integer value of a number
iter()	Returns an iterator object
list()	Creates a list
locals()	Returns a dictionary containing the variables defined in the local namespace
map()	Returns a list of results after applying a given function to each item of an iterable
max()	Returns a maximum value in the sequence
min()	Returns a minimum value in the sequence
next()	Used to fetch the next item in the sequence
open()	Opens a file and returns a corresponding file object
pow()	Returns power of a number
print()	Prints the given object
range()	Returns an immutable sequence of numbers
round()	Round-off the digits of a number
reversed()	Returns the reversed iterator of the given sequence
set()	Returns unique values in a given sequence
slice()	To get slice of elements from a collection of elements
sorted()	To sort the elements
str()	Returns a string
sum()	To get the sum of numbers in an iterable
tuple()	Creates a tuple
type()	Returns the type of a specified object
zip()	Returns a zip object, which maps a similar index of multiple containers

5. LIST OPERATIONS

FUNCTION	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

6. TUPLE OPERATIONS

FUNCTION	DESCRIPTION
all()	Returns true if all elements are true or if tuple is empty
any()	return true if any element of the tuple is true. if tuple is empty, return false
len()	Returns length of the tuple or size of the tuple
enumerate()	Returns enumerate object of tuple
max()	return maximum element of given tuple
min()	return minimum element of given tuple
sum()	Sums up the numbers in the tuple
sorted()	input elements in the tuple and return a new sorted list
tuple()	Convert an iterable to a tuple.

7. SET OPERATIONS

FUNCTION	DESCRIPTION
add()	Adds an element to a set
remove()	Removes an element from a set. If the element is not present in the set, raise a KeyError
clear()	Removes all elements form a set
copy()	Returns a shallow copy of a set
pop()	Removes and returns an arbitrary set element. Raise KeyError if the set is empty
update()	Updates a set with the union of itself and others
union()	Returns the union of sets in a new set
difference()	Returns the difference of two or more sets as a new set
difference_update()	Removes all elements of another set from this set
discard()	Removes an element from a set if it is a member. (Do nothing if the element is not in the set)
intersection()	Returns the intersection of two sets as a new set
intersection_update()	Updates the set with the intersection of itself and another
isdisjoint()	Returns True if two sets have a null intersection
issubset()	Returns True if another set contains this set
issuperset()	Returns True if this set contains another set
symmetric_difference()	Returns the symmetric difference of two sets as a new set
symmetric_difference_update()	Updates a set with the symmetric difference of itself and another

8. DICTIONARY OPERATIONS

FUNCTION	DESCRIPTION
copy()	They copy() method returns a shallow copy of the dictionary.
clear()	The clear() method removes all items from the dictionary.
pop()	Removes and returns an element from a dictionary having the given key.
popitem()	Removes the arbitrary key-value pair from the dictionary and returns it as tuple.
get()	It is a conventional method to access a value for a key.
dictionary_name.values()	returns a list of all the values available in a given dictionary.
str()	Produces a printable string representation of a dictionary.
update()	Adds dictionary dict2's key-values pairs to dict
setdefault()	Set dict[key]=default if key is not already in dict
keys()	Returns list of dictionary dict's keys
items()	Returns a list of dictionary's (key, value) tuple pairs
has_key()	Returns true if key in dictionary dict, false otherwise
fromkeys()	Create a new dictionary with keys from seq and values set to value.
type()	Returns the type of the passed variable.
cmp()	Compares elements of both dict.

9. CONDITIONAL STATEMENTS

STATEMENT	SYNTAX	EXAMPLE
if statement	if <expression>: <body; body is executed if expression is true>	if x < y: print('yes')

if.. else statement	if <expression>:
	<body; body is executed if expression is true>
	else:
	<body; body is executed if expression in 'if' is false>
		if x < y:
		print('yes')
		else:
		print('no')

Nested if-else statement	if <expression>:
	<body; body is executed if expression is true>
	elif <expression>:
	<body; body is executed if 'elif' expression is true>
	elif <expression>:
	<body; body is executed if 'elif' expression is true>
	....
	else:
	<body; body is executed if expression in 'if' is false>
		if time == 'morning':
		print('Good morning')
		elif name == 'afternoon':
		print('Good afternoon')
		elif name == 'evening':
		print('Good evening')
		...
		else:
		print("Good day")

For loop	for i in sequence: <statement(s) of for >	for val in range(11): print(val)
While loop	while <expression>: <statement(s) of while; is executed till expression is true >	while value < 10: print(value)
List Comprehension	[expression for item in list if conditional]	[x : x**2 for i in range(100) if x%2 ==0]