Hina Arora
@hinaaroraa

Don't Miss Out these 👇

IMPORTANT PYTHON METHODS !

SWIPE >>>

List Methods

01. append() ⟶ Adds an element to the end of the list.

Example :- 🔴🟩🟩 ----> append(🔺) ----> 🔴🟩🟩🔺

02. extend() ⟶ Extends the list by appending elements from an iterable.

Example :- 🔴🟩🟩 ----> extend(🔺🔴) ----> 🔴🟩🟩🔺🔴

03. pop() ⟶ Removes and returns an element from the end of the list.

Example :- 🔴🟨🟩 ----> pop(1) ----> 🔴🟩

04. index() ⟶ Returns the index of the first occurrence of a specified element in the list.

Example :- 🔴🟩🟩 ----> index(🔴) ----> 0

05. sort() ⟶ Sorts the list in place.

Example :- 🟩🔴🔺 ----> sort( ) ----> 🔴🟩🔺

06. reverse() ⟶ Reverses the order of the elements in the list.

Example :- 🔴🟩🟩 ----> reverse( ) ----> 🟩🟩🔴

SWIPE >>>

07. insert() ⟶

Inserts an element at a specified index in the list.

Example :- ●■■ ----> insert(3, ▲) ----> ●■■▲

08. remove() ⟶

Removes the first occurrence of a specified element from the list.

Example :- ●■■ ----> remove(●) ----> ■■

09. count() ⟶

Returns the number of occurrences of a specified element in the list.

Example :- ●■■ ----> remove(■) ----> 2

10. copy() ⟶

Returns a shallow copy of the list.

Example :- ●■■ ----> copy( ) ----> ●■■

SWIPE >>>

String Methods

01. upper() ⟶ Converts all characters to uppercase

Example :- 'hello' ----> upper() ----> HELLO

02. lower() ⟶ Converts all characters to lowercase

Example :- 'HELLO' ----> lower() ----> hello

03. strip() ⟶ Returns a trimmed version of the string

Example :- '   HELLO   ' ----> strip() ----> HELLO

04. split() ⟶ Splits the string at the specified separator, and returns a list.

Example :- '   HELLO   ' ----> strip() ----> HELLO

05. join() ⟶ Converts the elements of an iterable into a string

Example :- 'myTuple = ("John", "Peter", "Vicky")
----> "#".join(myTuple) ----> John Peter Vicky

06. replace() ⟶ Returns a string where a specified value is replaced with a specified value

Example :- '31/01/2022' ----> replace() ----> '31-01-2022'

SWIPE »»

07. startswith() ⟶ Returns True if the string starts with a specified prefix, otherwise False.

Example :- 'hello' ----> startswith("h") ----> True

08. endswith() ⟶ Returns True if the string ends with a specified suffix, otherwise False.

Example :- 'HELLO' ----> endswith('O') ----> True

09. find() ⟶ Returns the index of the first occurrence of a specified substring in the string, or -1 if not found.

Example :- 'HELLO WORLD' ----> find('OR') ----> 7

10. count() ⟶ Returns the number of non-overlapping occurrences of a specified substring in the string.

Example :- 'HELLO WORLD' ----> count('L') ----> 3

SWIPE »»

Dictionary Methods

Let's Assume :- abc = {'A':1, 'B':2, 'C':3}

01. get() ⟶ Returns the value associated with a specified key, or a default value if the key does not exist.

Example :- get('C') ----> 3

02. keys() ⟶ Returns a view object that displays a list of all keys in the dictionary.

Example :- keys() ----> dict_keys(['A', 'B', 'C'])

03. values() ⟶ Returns a view object that displays a list of all values in the dictionary.

Example :- values() ----> dict_values([1, 2, 3])

04. items() ⟶ Returns a view object that displays a list of all key-value pairs in the dictionary as tuples.

Example :- items() ----> dict_items([('A', 1), ('B', 2), ('C', 3)])

05. update() ⟶ Updates the dictionary with key-value pairs from another dictionary or an iterable.

Example :- update('D':4) ----> {'A':1, 'B':2, 'C':3, 'D':4}

SWIPE >>>

Let's Assume :- abc = {'A':1, 'B':2, 'C':3}

06. pop() ⟶

Removes and returns the value associated with a specified key from the dictionary.

Example :- pop('B') ----> 2

07. popitem() ⟶

Removes and returns an arbitrary key-value pair from the dictionary.

Example :- popitem() ----> ('C', 3)

08. clear() ⟶

Removes all key-value pairs from the dictionary.

Example :- clear() ----> {}

09. setdefault() ⟶

Returns the value associated with a specified key, or adds a key-value pair with a default value if the key does not exist.

Example :- setdefault('C') ----> 3

10. copy() ⟶

Returns a shallow copy of the dictionary.

Example :- copy() ----> {'A':1, 'B':2, 'C':3}

SWIPE >>>

Set Methods

Let's Assume :- fruits = {"apple", "banana", "cherry"}

01. add() ⟶ Adds an element to the set.

Example :- fruits.add("orange") ----> {'orange', 'apple', 'banana', 'cherry'}

02. remove() ⟶ Removes an element from the set.

Example :- fruits.remove("banana") ----> {'cherry', 'apple'}

03. union() ⟶ Returns a new set containing all elements from the set and another set.

Example :- union({"google", "microsoft"}) ----> {'cherry', 'apple', 'google', 'banana', 'microsoft'}

04. intersection () ⟶ Returns a new set containing elements that are common to the set and another set.

Example :- intersection({"apple"}) ----> {'apple'}

05. difference() ⟶ Returns a new set containing elements that are in the set but not in another set.

Example :- difference({"microsoft", "apple"}) ----> {'cherry', 'banana'}

06. issubset() ⟶ Returns True if the set is a subset of another set, otherwise False.

Example :- fruits.issubset({"apple", "banana", "cherry", "orange"}) ----> True

SWIPE >>>

File Methods

01. open() →	Opens a file and returns a file object.
02. read() →	Reads the contents of a file and returns it as a string.
03. write() →	Writes a string to a file.
04. close() →	Closes a file.

Math Methods

01. abs() → Returns the absolute value of a number.

Example :- abs(-94) ----> 94

02. pow() → Returns the result of raising a number to a specified power.

Example :- Math.pow(9,3) ----> 729

03. sqrt() → Returns the square root of a number.

Example :- math.sqrt(16) ----> 4.0

04. max() → Returns the maximum value among a series of numbers.

Example :- max(5, 10) ----> 10

05. min() → Returns the minimum value among a series of numbers.

Example :- min(5, 10) ----> 5

SWIPE »»

Regular Expression Methods

01. re.match() →	Determines if the regular expression matches at the beginning of a string.
02. re.search() →	Searches a string for a match to the regular expression.
03. re.findall() →	Returns all non-overlapping occurrences of a pattern in a string as a list.
04. re.sub() →	Replaces all occurrences of a pattern in a string with a replacement string.
05. re.compile() →	Compiles a regular

Date and Time Methods

01. datetime.now() →	Returns the current date and time as a datetime object.
02. strftime() →	Returns a formatted string representing a date and time
03. strptime() →	Parses a string representing a date and time and returns a datetime object.
04. timedelta() →	Represents a duration of time.
05. date() →	Represents a date (year, month, and day).

SWIPE »>

END

Don't Forget to Follow For More !

Thank you for Reading

Checkout my previous post too

COLLABORATION

Hina Arora
Tech Manager at Jio Health Hub

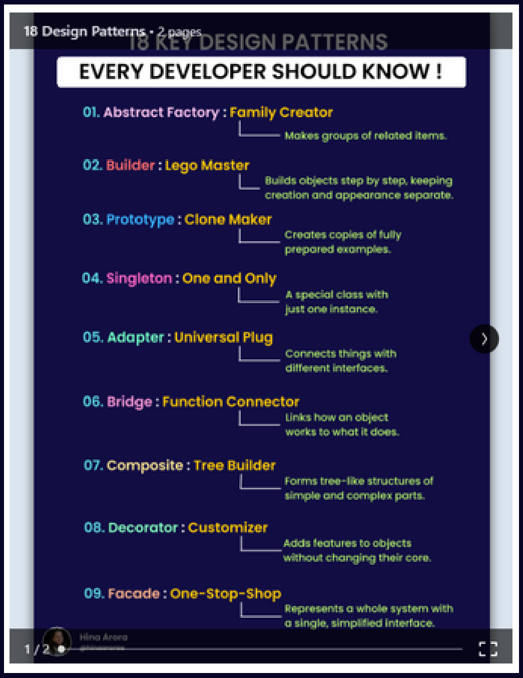
Omar Halabieh
Tech Director at Amazon Payment Service

Making the Leap from IC to Manager ? 10 Tips to Guide Your Journey !

Hina Arora @hinaaroraa

Omar Halabieh @omarhalabieh

Click Here


18 Design Patterns · 2 pages

18 KEY DESIGN PATTERNS

EVERY DEVELOPER SHOULD KNOW !

01. Abstract Factory : Family Creator
Makes groups of related items.

02. Builder : Lego Master
Builds objects step by step, keeping creation and appearance separate.

03. Prototype : Clone Maker
Creates copies of fully prepared examples.

04. Singleton : One and Only
A special class with just one instance.

05. Adapter : Universal Plug
Connects things with different interfaces.

06. Bridge : Function Connector
Links how an object works to what it does.

07. Composite : Tree Builder
Forms tree-like structures of simple and complex parts.

08. Decorator : Customizer
Adds features to objects without changing their core.

09. Facade : One-Stop-Shop
Represents a whole system with a single, simplified interface.

Hina Arora

1 / 2

Click Here