PYTHONIC DATA CONTAINERS PART-1

By Rishabh IO
linkedin.com/in/rishabhio

Embrace the elegance of Pythonic data structures, for within lists & dictionaries, a world of boundless creativity awaits. *--unknown*

Why do we need Lists?

ADD / REMOVE / UPDATE

IN MEMORY

Efficiently organize and manage collections of related data and perform data manipulation tasks.

APPLY OPERATIONS

e.g. Sorting

List
Syntax
001

Any valid Python Datatype can be added to list

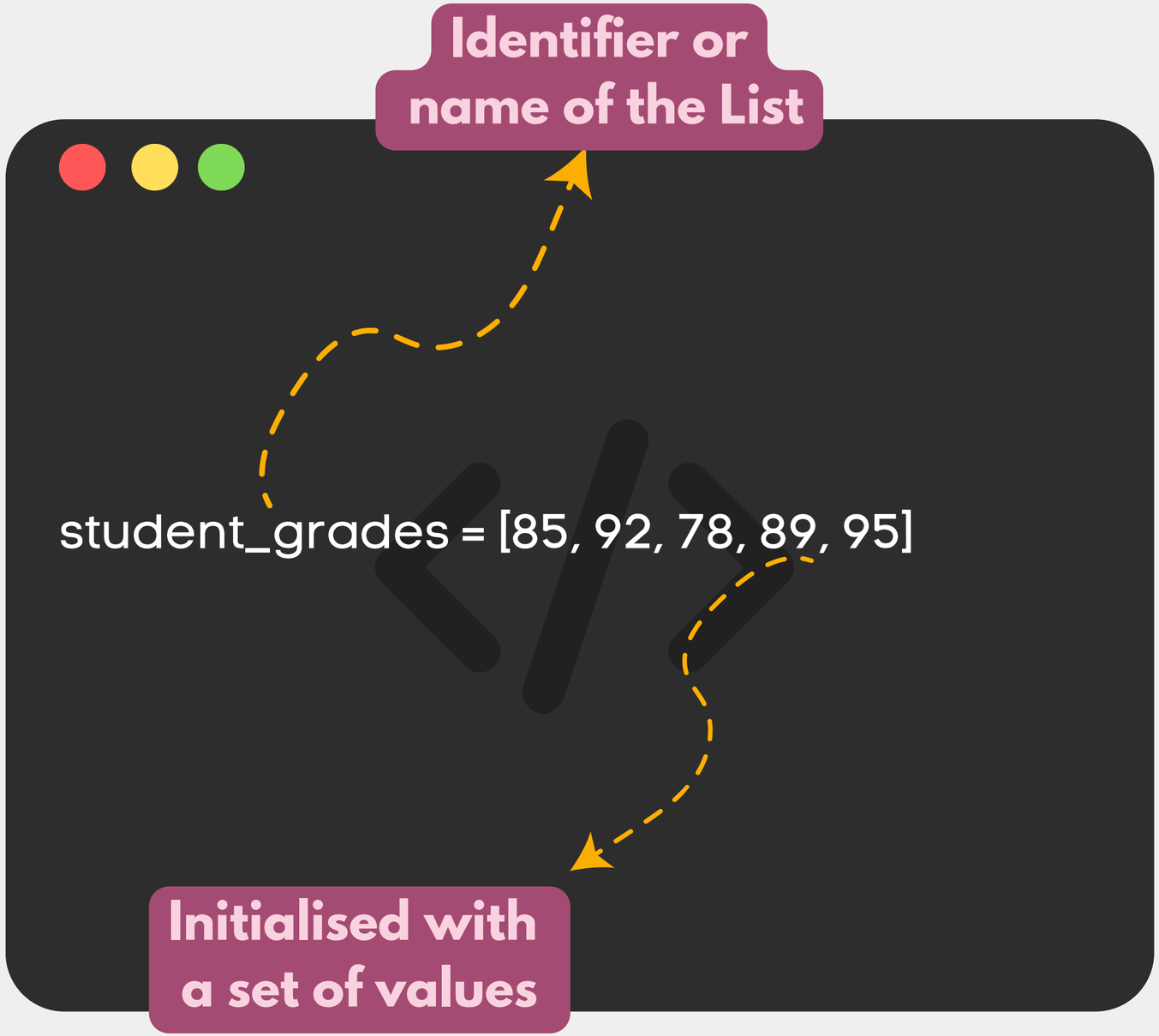
list_identifier = [ ]

square brackets initialize a list

any valid identifier in Python

Can be Heterogeneous

Code Examples 1



Code Examples 2

```
shopping_list = ["apples", "eggs", "milk", "bread"]

employee_names = ["John", "Jane", "Smith"]

playlist = ["Song 1", "Song 2", "Song 3" ]

stock_prices = [150.25, 155.50, 148.75, 160.20]
```

List
Syntax
002

Creates an
Empty List

my_list = list()

built-in function which
returns an empty list or
with elements of iterable
passed as a parameter

any valid
identifier in
Python

Also accepts
iterables as
parameters

Creating Lists from Iterables

Code Examples 3

```
>>> char_list = list("hello")
['h', 'e', 'l', 'l', 'o']

>>> my_list = list(range(1, 6))
[1, 2, 3, 4, 5]

>>> grid = [[0, 1, 0],
            [1, 1, 0],
            [0, 0, 1]]
```

Also called 2D list or matrix

List Syntax 003

Creates a list based on the expression

```
my_list = [ expression for item in iterable if condition]
```

Building lists using the list comprehension technique

any valid identifier in Python

Also accepts conditions

Creating Lists using List comprehensions

Code Examples 4

```
>>> my_list = [x for x in range(5)]
[0, 1, 2, 3, 4]

>>> squares =   = [x**2 for x range(5) ]
[0, 1, 4, 9, 16]

>>> evens = [x for x in range(10) if x % 2 == 0]
[0, 2, 4, 6, 8]
```

Replication using * operator

Code Examples 5

```
>>> lst = [0] * 5
[0, 0, 0, 0, 0]

>>> my_lst  = [ 'a', 'b' ] * 2
['a', 'b', 'a', 'b']

>>> bool_list = [ True ] * 2
[ True, True ]
```

Lists of Random Numbers

Code Examples 6

```
import random

>>> [random.randint(1, 100) for _ in
range(5)]
[50, 97, 46, 49, 37]

>>> [random.random() for _ in range(2)]
[0.2988025371630223,
0.5078970340293492]

>>>  random.sample(range(1, 100), 5)
[49, 85, 3, 62, 82]
```

Shuffling a list

Code Examples 7

```
import random

my_list = [1, 2, 3, 4, 5]
random.shuffle(my_list)
print(my_list)
```

[3, 4, 5, 2, 1]

Printing all List Methods

Code Examples 8

```
my_list = [ 1, 2, 3, 4, 5  ]

>> dir( my_list )

[ 'append', 'clear', 'copy', 'count', 'extend',
'index', 'insert', 'pop', 'remove', 'reverse',
'sort' ]
```

Traversing a List or

Visiting all elements of a List

Code Examples 9

```
my_list = [ 1, 2, 3, 4, 5  ]

# Method-1
for num in my_list:
    print(my_list)


# Method-2
n = len( my_list )
for i in range(0, n):
    print( my_list[ i ] )
```

Common List Operations

Code Examples 10

```
my_list = [ 1, 2, 3, 4, 5  ]
# Access an element on a particular index
second_element = my_list[ 1 ] # indexing starts at 0

# Overwriting an element at an index
>>> my_list[ 2 ] = 15
[ 1, 2, 15, 4, 5 ]

# Delete an element
>>> del my_list[ 0 ]
[ 2, 15, 4, 5 ]
```

List is mutable

Operations on List

Code Examples 11

```
my_list = [ 1, 2, 3, 4, 5  ]
# Access an element on a particular index
second_element = my_list[ 1 ] # indexing starts at 0

# Overwriting an element at an index
>>> my_list[ 2 ] = 15
[ 1, 2, 15, 4, 5 ]

# Delete an element
>>> del my_list[ 0 ]
[ 2, 15, 4, 5 ]
```

Negative Indexing

```
my_list = [1, 2, 3, 4, 5]
print(my_list[-1])   # Output: 5
print(my_list[-2])  # Output: 4
print(my_list[-3])  # Output: 3
```

Negative indexing in Python lists allows you to access elements from the end of the list by using negative integers as indices

access elements from the end

-ve indices possible in Lists

List Slicing

Creating a list using list slicing

```
new_list = original_list[start:stop:step]

original_list = [1, 2, 3, 4, 5]
new_list = original_list[1:4]
print(new_list)  # Output: [2, 3, 4]
```

technique used to extract a portion, or a subsequence, of elements from a list

slicing creates a new list

-ve indices possible in Lists

List Methods - 1

Code Examples 12

```
# list_name.append(element)
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # Output: [1, 2, 3, 4]


# list_name.clear()
my_list = [1, 2, 3]
my_list.clear()
print(my_list)  # Output: []
```

List Methods - 2

Code Examples 13

```
# copy() - Creates a shallow copy of the list.
my_list = [1, 2, 3]
new_list = my_list.copy()
print(new_list)  # Output: [1, 2, 3]

# count(element) - Returns the number of occurrences of an element in the list
my_list = [1, 2, 2, 3, 2]
count = my_list.count(2)
print(count)  # Output: 3
```

List Methods - 3

Code Examples 14

```
# extend(iterable) Appends elements from an iterable to the end of the list
my_list = [1, 2, 3]
my_list.extend([4, 5, 6])
print(my_list)  # Output: [1, 2, 3, 4, 5, 6]

# index(element) - Returns the index of the first occurrence of an element in the list
my_list = [1, 2, 3, 2]
index = my_list.index(2)
print(index)  # Output: 1
```

List Methods - 4

Code Examples 15

```
# insert(index, element) Inserts an element at a specific index in the list
my_list = [1, 2, 3]
my_list.insert(1, 4)
print(my_list)  # Output: [1, 4, 2, 3]

# pop(index) - Removes and returns the element at a specific index in the list
my_list = [1, 2, 3]
element = my_list.pop(1)
print(element)  # Output: 2
print(my_list)  # Output: [1, 3]
```

List Methods - 5

Code Examples 16

```
# remove(element) Removes the first occurrence of an element from the list
my_list = [1, 2, 3, 2]
my_list.remove(2)
print(my_list)  # Output: [1, 3, 2]

# reverse() - Reverses the order of elements in the list
my_list = [1, 2, 3]
my_list.reverse()
print(my_list)  # Output: [3, 2, 1]
```

List Methods - 6

Code Examples 17

```
# sort() Sorts the elements of the list in ascending order


my_list = [3, 1, 2]
my_list.sort()
print(my_list)  # Output: [1, 2, 3]
```

Code Activity

Give a List ::
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Task-1 :: REVERSE the list
Task-2 :: Print the elements on the same line

Hint: Did you use .reverse () ? / Is there any other technique

Dictionaries in Python

key-value pairs, allowing efficient retrieval and manipulation of data.

Dictionary Syntax

Curly braces used to initialise an empty dict

dict_identifier = {}

curly braces initialize a list

```
phonebook = {
    "Alice": "1234567890",
    "Bob": "9876543210",
    "Charlie": "5555555555"
}
```

any valid identifier in Python

Any immutable datatype can be a key

Dictionary Syntax

built-in function to create an empty dict

my_dict = dict()

using built-in function

```
# Adding key-values to my_dict
my_dict[1] = "Python"

# Accessing value with a key
print( my_dict[1]
```

any valid identifier in Python

Can also be used to typecast

Creating Dictionaries using List comprehensions

Code Examples Error

<count lost>

```
>>> squares = {x: x**2 for x in range(1, 6)}
>>> print(squares)
# Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
    •
>>> values = ["a", "b", " = ["a", "b", "c"]c"]
>>> my_dict = {k: v for k, v in
enumerate(keys) }
>>> print(my_dict)
# Output: {0: 'a', 1: 'b', 2: 'c'}
```

Common Dictionary Operations

Code Examples

```
# Access a value by key
my_dict = {"name": "John", "age": 30}
print(my_dict["name"])  # Output: John
print(my_dict.get("name") # John

# Check if a key exists
my_dict = {"name": "John", "age": 30}
print("name" in my_dict)  # Output: True

# Delete a key value pair
my_dict = {"name": "John", "age": 30}
del my_dict["age"]
print(my_dict)  # Output: {"name": "John"}
```

Dict is mutable

Printing all Dictionary Methods

Code Examples

```
my_dict = {"name": "John", "age": 30}

>> dir( my_dict )

[ 'clear', 'copy', 'fromkeys', 'get', 'items',
'keys', 'pop', 'popitem', 'setdefault', 'update',
'values' ]
```

Traversing a Dict or

Visiting all k-v pairs of a dict

Code Examples

```
.items()
# Returns a view object containing key-
# value pairs of the dictionary

my_dict = {"name": "David", "age": 40}

for key, value in my_dict.items():
    print(key, value)
# Output:
# name David
# age 40
```

Dict Methods - 2

Code Examples

```
# clear() - Removes all key-value pairs from the dictionary.
my_dict = {"name": "Alice", "age": 25}
my_dict.clear()
print(my_dict)  # Output: {}

# copy() - Creates a shallow copy of the dictionary
original_dict = {"name": "Bob", "age": 30}
new_dict = original_dict.copy()
print(new_dict)  # Output: {"name": "Bob", "age": 30}
```

Dict Methods - fromkeys

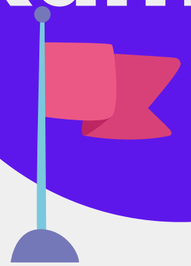
Code Examples

```
# fromkeys(keys, value) Creates a new dictionary with the specified keys and a common value

keys = ["apple", "banana", "orange"]
default_price = 0.99
fruits_dict = dict.fromkeys(keys, default_price)
print(fruits_dict)
# Output: {"apple": 0.99, "banana": 0.99, "orange": 0.99}
```

Dict Methods - Get

Code Examples



```
# get(key, default) Retrieves the value associated with a key, or a default value if the key is not found

my_dict = {"name": "Charlie", "age": 35}

print(my_dict.get("name", "Unknown"))  # Output: Charlie

print(my_dict.get("city", "Unknown"))  # Output: Unknown
```

Dict Methods - keys() and values()

Code Examples

```
# keys() returns all the keys
# values() returns all the values

my_dict = {"name": "Eve", "age": 45}
keys = my_dict.keys()
print(keys)
# Output: dict_keys(['name', 'age'])

values = my_dict.values()
print(values)
# Output: dict_values(['Eve', 45])
```

Dict Methods - pop

Code Examples

```
# pop(key, default) Removes and returns the value associated with a key, or a default value if the key is not found

my_dict = {"name": "Frank", "age": 50}
age = my_dict.pop("age")
print(age)      # Output: 50
print(my_dict)  # Output: {"name": "Frank"}
```

Dict Methods - popitem

Code Examples

```
# popitem() Removes and returns the last inserted key-value pair as a tuple

my_dict = {"name": "Grace", "age": 55}

last_item = my_dict.popitem()

print(last_item)  # Output: ('age', 55)

print(my_dict)    # Output: {"name": "Grace"}
```

Dict Methods - update

Code Examples

```
# update(other_dict) Updates the dictionary with the key-value pairs from another dictionary

my_dict = {"name": "Isaac", "age": 65}
other_dict = {"age": 66, "city": "London"}
my_dict.update(other_dict)
print(my_dict)
# Output: {"name": "Isaac", "age": 66, "city": "London"}
```