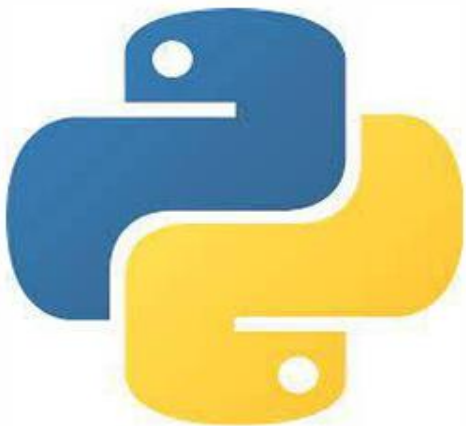
python Programming

K Vinod


Guido
van
Rossum

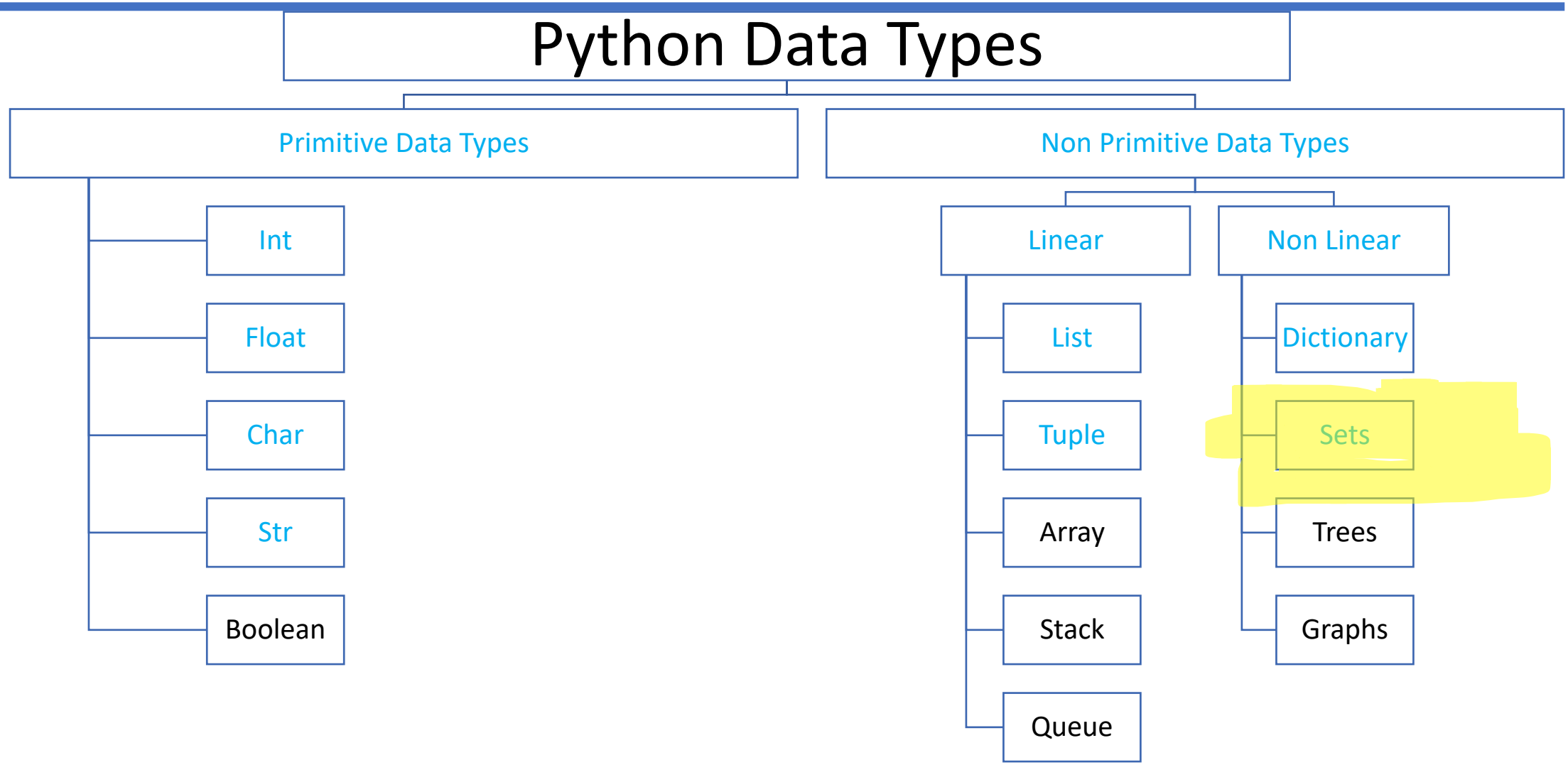

MONTY PYTHON'S
FLYING
CIRCUS



Module 3

Topic	Page No
3.1 List	
3.2 Tuple	
3.3 Dictionary	

K Vinod


Python Data Types
Primitive Data Types
Non Primitive Data Types
Int
Float
Char
Str
Boolean
Linear
Non Linear
List
Tuple
Array
Stack
Queue
Dictionary
Sets
Trees
Graphs

K Vinod

List


```
# here we use square
brackets
numbers = [1, 2, 2, 4]
print(numbers)

words = ["war", 'ds']
print(words)

mixed = [1,2,3,'dcscx',
02.2, None]
print(mixed)
print(mixed[2])
```

```
print(numbers[ : 2])

mixed[1] = 'change'
print(mixed)
mixed[1: ] = [1,2,3]
print(mixed)
```

List

how to add items to your list
most common thing you can do with your list and most important

```
fruits = ['mango','grapes']
fruits.append('apple')
print(fruits)

fruits = []
fruits.append(10)
fruits.append(145)
print(fruits)
```

```
# some more methods to add the data
# insert method
# how to join(concatenate) method
# extend method
# difference between append and extend method

fruits1 = [10,'apple']
fruits1.insert(0,'mango')
# print(fruits1)
fruits2 = [100,1000]
# fruits = fruits1 + fruits2
# print(fruits)

# fruits1.append(fruits2)



# print(fruits1)

fruits1.extend(fruits2)
print(fruits1)
```

```
# common methods to delete the data in list

a = ['grapes','grapes','mando', 26]\

# # pop method
# a.pop()
# # be default it will delete the last data
# print(a)
# a.pop(2)
# print(a)

# del operator
# del a[0]
# print(a)

# remove method
a.remove('mando')
print(a)
```

List

```
# count
# sort method
# sorted function
# reverse
# clear
# copy
```

```
fruits =['grapes','mango','apple']
if 'mango' in fruits:
    print("mango is present")
else:
    print("not present")
```

```
print(fruits1 == fruits3) # values are same
print(fruits1 is fruits3)
```

Join and Split Methods in List

```
user_info = 'udayan care 2022'.split() # split acc to spaces
user_info = 'udayan care.2022'.split('.') # specific value
print(user_info)

name,age = 'udayan care.2022'.split('.')
name,age = input("enter tha name and age : ").split('.')
print(name)
print(age)
```

Loop in List

```
fruits2 = ['orange','apple','mango','banana','orange','apple']

for fruit in fruits2:
    print(fruit)

i = 0
while i < len(fruits2):
    print(fruits2[i])
    i += 1
```

List inside List

```
matrix = [[1,2,3],[4,5,6],[7,8,9]] # 2d list
print(matrix[0])

for i in matrix:
    print(i)

for sublist in matrix:
    for i in sublist:
        print(i)

print(matrix[1][2])

s = "care"
print(type(s))
print(type(matrix))
```

More about Lists

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
def negative_list(l):
    negative = []
    for i in l:
        negative.append(-i)
    return negative
print(negative_list(numbers))
```

Practical

```
# define a function which will take list containing numbers as
input
# and return list containing sqaure of every elements
# example
# numbers = [1,2,3,4]
# square_list(numbers) ------ return -----[1,4,9,16]
```

Practical

```
def square_list(l):
    square = []
    for i in l:
        square.append(i**2)
    return square

numbers = list(range(1,11))
print(square_list(numbers))
```

```
# define a function which will take list as a argument and
this function
# will return a reversed list
# example
# [1,2,3,4]------[4,3,2,1]
# note you simply do this with reverse method or [::-1]
# but try to do with the help of append and return method
```

Practical

```
# def reverse_list(l):
#     l.reverse()
#     return l

# def reverse_list(l):
#     return l[::-1]

def reverse_list(l):
    r_list = []
    for i in range(len(l)):
        popped_item = l.pop()
        r_list.append(popped_item)
    return r_list
numbers = [1,2,3,4,5]
print(reverse_list(numbers))
```

Practical

```
# define a function that take list of words as a argument and
# return list with reverse of every element in list

# example
# ['abc','cde']----['cba','edc']
```

Practical

```
# here reverse method will not work in strings
def reverse_elements(l):
    elements = []
    for i in l:
        elements.append(i[::-1])
    return elements
words = ["abc","bgh","dsd"]
print(reverse_elements(words))
```

Practical

```
# filter odd and even
# define a function
# input
# list ------> [1,2,3,4,5,6,7,8,9]
# output ---[1,3,5,7,9] [2,4,6,8]
```

Practical

```
def filter_odd_even(l):
    odd_nums = []
    even_nums = []
    for i in l:
        if i%2 == 0:
            even_nums.append(i)
        else:
            odd_nums.append(i)
    output = [odd_nums,even_nums]
    return output

filter = [1,2,3,4,5,6,7,8,9]
print(filter_odd_even(filter))
```

Practical

```
# common elements finder function
# define a function with two input lists return a list
# which contains common elements of both the list

# example -- > [1,2,3,4,5,6] [2,5,6]
# output [2,5,6]
```

Practical

```
def common_finder(l1,l2):
    output = []
    for i in l1:
        if i in l2:
            output.append(i)
    return output

print(common_finder([1,2,3,4,5,6],[2,5,6]))
```

Practical

```
num = [6,25,30]
print(min(num))
print(max(num))

def greatest_differ(l):
    return max(l) - min(l)
print(greatest_differ(num))
```

Practical

```
# last exercise
# function
# [1,2,[1,2,3]] input

# finding the how many list in the list?

#  output 1

# [1,2,[1,2,3],[1,5,8]] input
# output 2
```

Practical

```
def sublist_counter(l):
    count = 0
    for i in l:
        if type(i) == list:
            count += 1
    return count

mixed = [1,2,[1,2,3],[1,5,8]]

print(sublist_counter(mixed))
```

TUPLE

```
# tuple is a data structure
# tuple can store any type of data
# most important
# tuples are immutable, once tuple is created u can't update

example = ('one','two','three')

# no append, no insert, no pop, no remove
```

Practical

```
# tuples used only when we know the values should not change
days =
('monday','tuesday','wednesday','thursday','friday','saturday')

# tuples are faster than lists
# methods
# count, index
# len function
# slicing
print(example[:2])
```

Practical

```
# tuple with one element
nums1 = (2,) # this is tuple
nums = (1)   # this is not a tuple
words = ('word1')   # this is not a tuple

# tuple without paranthesis
guitars = 'yamaha' , 'battan rogue' , 'taylor’

# tuple unpacking
men_in_blue = ('virat' , 'pant' , 'bumrah')
batsman , wicket_keeper , bowler = (men_in_blue)
```

Practical

```
# function returning two values
def func(int1,int2):
    add = int1 + int2
    multiply = int1*int2
    return add, multiply

print(func(2,3))

add , multiply = func(2,3)

print(add)
print(multiply)
```

Dictionary in Python

```
# how to create dictionaries
user = {'name':'india', 'age':1947}
print(user)
print(type(user))

# second method to create dictionaries
user1 = dict(name = 'Indian Army', age = 1947)
print(user1)

# how to access data from dictionaries
# NOTE - there is no indexing because of unordered collection of
data.
print(user1['name'])
```

Dictionary in Python

```
# which type of data a dictionary can store?
# anything - numbers,strings, list, dictionary
user_info = {
    'name' : 'Indian Army',
    "age" : 1947,
    "fav_operations" : ['operation polo', 'operation polo 2'],
    "fav_tunes" : ['jana gana mana', 'vandematraam']
}
print(user_info)

# how to add data to empty dictionary?
user_info2 = {}
user_info2['name'] = 'Indian Air Force'
print(user_info2)
```

Dictionary in Python

```
# check if key exist in dictionary
# if 'name' in user_info:
#     print("present")
# else:
#     print("Not Present")

# check if value exist in dictionary ---> values method
# if 'Indian Army' in user_info.values():
#     print("present")
# else:
#     print("Not Present")
```

Dictionary in Python

```
# loops in dictionary
# for i in user_info:
#     print(i)

# for i in user_info.values():
#     print(i)

# values method
# user_info_values = user_info.values()
# print(user_info_values)
# print(type(user_info_values))

# keys method
# user_info_keys = user_info.keys()
# print(user_info_keys)
# print(type(user_info_keys))
```

Dictionary in Python

```
# loops in dictionary
# for i in user_info:
#     print(user_info[i])

# items method -- > most important method ([(), (), ()])
# user_items = user_info.items()
# print(user_items)
# print(type(user_items))

for key, value in user_info.items():
    print(f"key is {key} and value is {value}")
```

Dictionary in Python

```
# how to add data
# user_info['fav_songs'] = ['song1','song2']
# print(user_info)

# pop method
# popped_item = user_info.pop("fav_tunes")
# print(f"popped item is {popped_item}")

# pop item method - randomly deleting key value pair
popped_item = user_info.popitem()
print(type(popped_item))
print(user_info)
```

Dictionary in Python

```
more_info = {'name': 'indian', 'state':'Telangana' , 'Hobbies':
['yoga','exercising']}

user_info.update(more_info)
print(user_info)
```

Dictionary in Python

```
# get method(useful)
d = {'name':'unknown','age':'unknown'}
# print(d['names']) --> to overcome this we use get method
# print(d.get('names')) -- this is better

# e = dict.fromkeys('abc', 'unknown')
# print(e)
# f = dict.fromkeys(range(1,11), 'unknown')
# print(f)
# g = dict.fromkeys(['name','age'], ['unknown','unknown'])
# print(g)
```

```
# more about get, two same keys in dictionaries
user = {'name':'india','age':1947, 'age':143}
# print(user.get('names','not found!'))
print(user)
```

```
# exercise
# define a function that takes number(n)
# return a dictionary containing cube of numbers from 1 to n

# example-->
# cube_finder(3)
# {1:1, 2:8, 3:27}
```

```
# cube finder
def cube_finder(n):
    cubes = {}
    for i in range(1, n+1):
        cubes[i] = i**3
    return cubes

print(cube_finder(10))
```

Dictionary in Python

```
from typing import Counter

# word Counter
def word_counter(s):
    count = {}
    for char in s:
        count[char] = s.count(char)
    return count

print(word_counter('ramyagaddam'))
```

Dictionary in Python

```
user = {}

name = input("waht is ur name ? : ")
age = input("waht is ur age ? : ")
fav_mov = input("ur favourite movies seperated by comma , ").split(',')
fav_song = input("ur favourite songs seperated by comma , ").split(',')

user['name'] = name
user['age'] = age
user['movies'] = fav_mov
user['songs'] = fav_song

# print(user)

for key, value in user.items():
    print(f"{key} : {value}")
```

K Vinod

Thank You