Fundamentals of Python Programming

YT- YouTube

Date.....

Coolboymannu YT

Index

Sno	Topics	Page. no
1.	Python Programming language intro	1
2.	Input and Output, Operators and Data type.	2-3
3.	Conditional statements	4-5
4.	functions and Recursion function	5-6
5.	4 collection type of data	
	• List	7
	• Tuple	8
	• Dictionary	9
	• Sets	10
6.	Exceptions Handling	12
7.	All functions of List, Tuple, sets, Dictionary	13
8.	file handling	14
9.	Anonymous or Lambda functions	15
10.	F-String and *args and **kwargs	16
11.	Enumerate function and if__name__=='__main__':	17
12.	Decorators	18
13.	String	19-20
14.	Oops with Python	21-24

Spiral

organization [Python software foundation

Python – dynamicaly (Runtime)
C++ – statically (compile time)

Date.....

Python Programming language Intro

1. Python was developed Guido van Rossum in 1991.

(i) simple & easy
(ii) It's syntax is very simple
(iii) Python is an open source software
freely available on the internet you can freely download it & use it
(iv) Python is a case sensitive language.
(v) Python is an interpreted pl
it's run program line by line
(vi) Python is an extensible pl
[other pl No re write to a program]
(vii) object oriented pl.
(viii) dynamicaly typed pl
(ix) indentation is done using spaces.
(x) Block of codes → group of statements which executes simaltaniously.
(xi) indentation is used to mape program clears and easy to understand
(xii) Modular pl → cohesion is high & compling is low
↳ [collections of similar types of f^ns]
(xiii) supports the operator overloding only in ~~one~~ two case. [+ and *]

+	*
Add	Multiplication
Number string	Repratition
	a = 3 , b = 7
	y = a * 7
	[3 3 3 3 3 3 3]

(xiv) No concept of pointer
it's use concept of referances.

statically typed → initially declared

Date.....

1. **Print ( )** - this is a function it'uses for Print something on computer or output.

Print ("Manish") $\xrightarrow{\text{output}}$ Manish

a=1. Print ("Manish", a) → Manish 1

type ( ) Display the data type on computer

id ( ) identifiers, identi like a address

Data types

- Numeric
  - integer
  - complex Num
  - float
- Dictionary
- Boolean
- set
- sequence
  - Type
  - → strings
  - → List
  - → Tuple

complex number → $x + yi$

$2.6 + 3i$

Make any variable this is need to Don't take a spacial character or first number intiger.

<—> > ये भी सकता है नहीं भी

Date.....

end=" ", print output in same line

(3)

```
print ("""        # multiple line
           # output
        """)
```

⇒ input ( )

variable = <type> (input( ))

⇒ int ( ) , float ( )

Operators

+	Addition	Adds two num
—	Subtraction	one num from another
*	Multiplication	
/	Division	output is floating
//	Integer division	integer, non decimal num
%	Remainder	2)3(1 — remainder
**	Exponent	Raise a num to a power

```
variable_name = int(input('Enter'))
```

```
print ('one','Two','Three', sep='*')
  output ⇒  one* Two* Three
print ( one\n two \n Three)
  output ⇒  one
            Two
            Three
```

⇒ If you want to add special charectars in your string so you have to add '\' frent your speial charector like \' ⇒ '

Spiral

Date.....

Rounded to n decimal place

Print (format (floating_num, '.nf'))
Print (format (123.67891, '.2f'))
output ⇒ 123.67

Conditional execution

⇒ if( ):
Data

Chained conditional

⇒ if ( ):
statements
elif ( ):
else:

looping

While

while ( ):	while (condition is true):
[Increment / Decrement]	(do something)

for for (variable) in (sequence):
statements

sequence
list or string

for 7 in range(10):
⇒ 7, 8, 9

sequence (1:7)
1, 2, 3, 4, 5, 6

Par → Parameters
arg → Arguments

Date.....

Break it will take out the loop.
exit out

Continue it will restart the loop
Jumping

Pass it will give nothing output

Functions

⇒ if you want to access any functions in library as well as Struct's member you have to use (.) ~~dought~~ dot opperator. like math.sum()
sum is a function of ~~that~~ math's library

function syntex

```
def name(Parameters):
    ...
    print
    return ...
```

Calling a function | function-name (Par and Arg) |

Date.....

Recursion

⇒ Recursion is one of the most powerful tool in a program language. Recursion is a way of programming where function calls itself again and again. Recursion is defined as defining something in terms of itself.

- There must be terminating condition for the problem.

```
def power(x,n):
    if (n==0):
        return 1
    else:
        return x* power(x, n-1)

a=2
print(power(a,4))
```

Modular programming

It's a software design technique to split your code into separate parts. These parts are called module.

Delete a lement- del [0]
list.pop(0)

Date.....

Lists data type

- Lists are ordered collection of data items.
- They store multiple items in a single-variable.
- Lists items are separated by commas and enclosed within square brackets [ ].
- Lists are changeable meaning we can alter them after creation.
- List elements can be accessed by its indexes.
- List is a mutable objects ordered.
- List may have duplicate values.

Syntax ⇒ list_name = [value 1, value2, value3 ... valuen]
Syntax to access python list
(i) list_name[index]
list slicing ⇒ (ii) list_name[start : stop : Jump(step)]

(i) adding lists	list 1 = list a + list b
(ii) Delete elements	del (list [index])
(iii) minimum value	min (list)
(iv) max largest value	max (list)
(v) return number of elements	len (list)
(vi) count object in list	list. count (object)
(vii) add obj in end	list. append (obj)
(viii) add obj & list in end	list. extend (obj, list)
(ix) Know the index number of obj	list. index (obj)
(x) add index's num objects to	list. insert (index, objects)
(xi) remove obj in list	list. remove (obj)
(xii) reverse the list	list. reverse ()
(xiii) if same data type arrange the list	list. sort ()

index need [ ] square brackets

Date.....

C++ Used defined data type / Abstract data type

(i) structure data type
(ii) union "
(iii) class "
(iv) Enumerated "

Python supports 4 collection types of data

(i) List
(ii) Tuple
(iii) set
(iv) Dictinory

→ Mutable, Higher memory, elements accessing is slow

→ Unmutable, less memory, faster.

Tuple →
- Immutable (unchangeable)
- orderd (They may be accessed through it's index)
- Duplicates are allowed
- elements accessing is faster
- Less memory
- Parentheses ( )
- Where all used comma separated is become tuple
- Tuples are used as dictionary Keys
- Tuples can contain list and tuples also.

Two way to make a tuple

(i) a = (1, 2, 3) / a = 'Mahisn', comma is needed

(ii) t = tuple(a, b, c) tuple( ) function can make a tuple

Date.....

operations on Tuples.

(i) Adding with '+'
(ii) Replicating with '*'

(i) len (tuple) length
(ii) max (tuple) max value
(iii) min (tuple) min value
(iv) tuple (seq) make any list or string into tuple.

Advantages of Tuple

- Processing of tuple are faster then Lists.
- It makes the data safe as tuples are immutable and hence can't be changed.
- Tuples are used for string formating.

Dictionary

- unordered set of key and value pair.
- enclosed data within curly braces { }
- Key and value is known as items.
- Key should be unique.
- Key value separated by a colon(:) and item (key + value) separated by comma(,)

Properties.

- it is mutable value can be updated.
- Key must be unique and immutable.
- Values accessed by key
- the ~~word~~ key works as an index and key are decided by the user.

fn ⇒ dict_name.keys() return keys
dict_keys(['1', '2'])

Date.....

Accessing Dictionary values.

- Dictionary values can be accessed by their keys only.

Updating Dictionary elements

- add a new key-value pair
- modifying an existing entry.
- delete an existing entry.

functions

dic.update({:})
Delete - dic.pop(key)
Return key dic.values()
dic.popitem()
len(dic)

Sets →

- It is an unordered collection of data of different data types.
- set does not contain duplicate data.
- A set is created using curly brackets '{ }'

set = {"A", "B"}

- We can't access items of sets using index value because it is unordered collection of data.
- we can't change the value of set.

functions →

1 len() - used to get lenght of set
2 add() - add new items in a set. [Only 1 item]
3 update() - add 1+ items in set.
4 remove() - remove specified item from a set.
5 (discard()) -
6 union() - we can join two set using union() function.
7 Clean() - used to clear or empty the set.

Date.....

8 del del - used to delete set completely.

1 len (set)
2 set.add("A")
3 set.update("A","B")
4 set.remove("A")
5 set.discard("B")
6 set.union(set1)
7 set.clear("A")
8 del set

some common operation (set)

(i) & (ii) | (iii) -

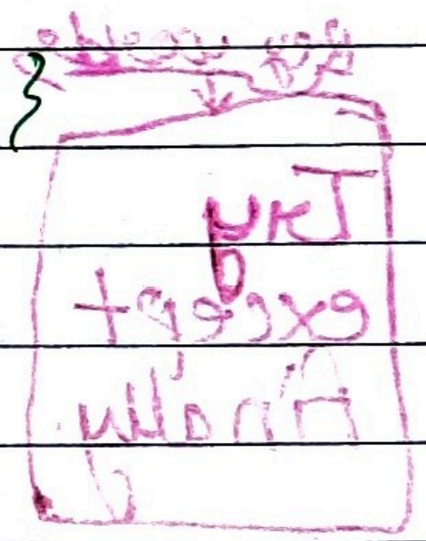
intersection (common element)

difference

union (All element not repeating)

setA = {4,5,6,9}
set B = {1,2,5,6,8}

print (setA & setB) # {5,6}
print (setA | setB) # {1,2,4,5,6,8,9}
print (setA - setB) # {9,4}



Date.....

Exceptions Handling

Programming errors → Bug

Debugging : The process of removing the errors in a program.

Types of errors:

1. Syntactic
2. Semantic
3. Run time or Exception

Exception handling : The process of removing run time errors or know as Exception handling.

Key words

Try
except
finally.

```
try :
    <body>
except <ExceptionType> :
    <handler>
```

All function of

Date.....

List - [,] (m)

len()
list.append(element)
list.insert(index, element)
list.remove(element)
list.pop(index)
list.clear()
del list[index]
del list
list2 = list1.copy()
list1 = extend(list2)
max(list)
min(list)
list.count(element)
list.index(element)
list.reverse()
list.sort()

delete element

Tuple - (,) (im)

tuple
len(tuple)
max(tuple)
min(tuple)

String " "

str.
find(" ")
str.islower()
.isupper()
.isspace()
.upper() all upper letter
.lower() " lower "

Sets - { } (im)

len(set)
set.add(element)
set.update([multiple elements])
set.remove(element)
set.discard(element)
set.union(set2)
set.clear()
del set
set1 & set2 — intersection
set1 | set2 — union
set1 - set2 — difference

Dictionary { : , } (m)

for i in list:
print(i, '=', list[i])

Dic.get(Key) = value
Dic[Key] = new value
Dic.pop
Dic.pop(Key)
len(Dic)
Dic[Key] = value
del Dic
Dic.clear()
Dic.keys() / Dic.values()
Dic.update(Key : value)

truncates → pointer will be on 0th position.

Date.....

File handling

File handling is a mechanism to store the data on the disk permanently.

Operation on file

Opening of file	fileptr = open("easy.txt", 'mode')
Writing into a file	
Appending data into "	
Reading from a file	
closing of file	fileptr.close()

Modes
- w - writing + create + truncates
- a - append + create
- r - reading
- t - text mode
- b - binary mode
- w+ - writing + reading
- r+ - writing + reading + truncates

filepointer.function()

file.write("content")	writing content in file
file.read(count)	reading " from "
file.readline()	read a whole line
file.readlines()	read all lines

Date.....

function , Recursion Advance level

Anonymous / Lambda functions

- Lambda functions are one liner function
- They are just convinence of coder

Normal function

```
def add(a,b):
    return a+b
print(add(10,20)      calling
```

lambda function

```
add = lambda x,y : x+y
print(add(10,20))     calling
```

fun-name = lambda x, y : x + y return

arguments

```
a=[[4,14],[5,6],[8,23]]
a.sort(key = lambda x : x[1])
print(a)
#output
[[5,6],[1,14],[8,23]]
```

Date.....

f- String

⇒ me = "Harry"
a = 10

① st = "This is %s %s" % (me, a)

② st = "This is {}{}" # This is Harry 10
a.format (me, a)

③ f-string

st = "This is {me} {a}" # This is Harry 10

*args and ** Kwargs Arguments

- *args arguments takes tuple and list, and argument will go in tuple form
- **kwargs arguments takes only Dictionary.

order of arguments.

(Normal arguments, *args, **kwargs)

```
def arg(*arg-t):
    print(arg-t)

list1 = [1, 2, 3, 4]
arg(*list1)
output - (1, 2, 3, 4)
```

Date.....

Enumerate function

Enumerate function gives the index and item.

```
list = [1, 2, 3, 'A']
for index, item in enumerate(list):
    print(f"index={index} and item={item}")
```

output :-

index = 0 and item = 1
index = 1 and item = 2
index = 2 and item = 3
index = 3 and item = 'A'

```
if __name__ == '__main__':
```

if __name__ == '__main__': in this section code will run if we run the program original file. other wise we are imparting the file in other file this section will not run.

name == mane run original code same file
name != main when file import other file.

Note → When ever we import the file interpreter runs the whole file

file.py

```
a = 'Hi'
print(a)
```

output :-
Hi

file2.py

```
import file
print("Hello")
```

output :-
Hi
Hello

Date.....

Decorators

When a function it pass itself into another function and another function's all logics came into main function

- We can pass a function in to another function
- We can re declaration a function with the other function's ~~defination~~ declaration.

```
def show(function):
    def give():
        print("this is first line")
        function()
        print("finished")

@show
def greed():
    print("HELLO WORLD")

#greed = show(greed)

greed()
```

Both means declaring a Decorator use only one

```
#output
this is first line
HELLO WORLD
finished
```

Date.....

String (unmutable)

- Strin is a collection of characters.
- It is created using single quotes or double and for multiline triple quotes.

str.capitalize() — first letter of string become uppercase

str.casefold() — whole str convorts into lowercase

str.center(10)
.center(10,'*') — #|****str****

str.find("element") — find in str ang gives 'e' index
.index("same") — " " 's'

str.isalnum() — string contain only alphabate (true) and number ~~false~~
.isalpha() — return true when all characters are alphabates
.isdigit()
.isidentifier() — Naming a variable
.islower() — all character are lowercase
.isnumeric()
.isupper() — all charaster are uppercase
.isspace()
.startswith('T')
.endswith('T')

Date.....

str.len() — it is used to get length

str.lower() — convert all into lowercase

str.upper() — " " " uppercase

str.swapcase() — lower to upper and upper to lower

str.strip() — " Hi " → " Hi " only 2 space start and end
123456 → 1234

- lstrip() → " Hi " → " Hi "
- rstrip() 12345 → 12345

str.replace("old","new")

str.split() — string is splited with space (default character) and it return all part into list

- split("*") — string is splited with '*'

Date.....

Oops with Python

- Python is an object-oriented programming language

Class, Object, Data member, Class variable, function overloading, Instance variable, Inheritance, Polymorphism, Encapsulation Data Abstraction, Instance, Method, Operator Overloading.

Class and Object

Class- A class is a blueprint

```
class classname:
    #class body
```

- collection of data member and functions.
- It is also called user defined data type.

⇒ Creating a object of a class

```
objectname = classname()
```

- We can access the variables of the class by using dot (.) operator with object.

Date.....

Instance and class variable

Instance variable → it is any objects property if defines outside the class

[objectname.variablename = value]

Class variable → these variables are property of class.

- objects shares the class variable
- Class can only change the value of class variable

[classname.variable_name = value]

Constructor

It is a special member function of class that executes when we create the instance (object) of that class. in other word we can say that there is no need to call a constructor.

__init__(self,...,...) method is used as a constructor in Python.

- object (self) is passed into class method so it is essential to add 'self' parameter.

```
def __del__(self):  ← Destructor
```

Date.....

```
def __init__(self):
    print("This is non parameterized"
          "constructor")
```

Inheritance

The process of getting property of one class into another class is called inheritance.

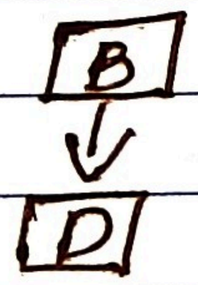
- Parent / super / Base class
- Child / sub / Derived class

```
class classname(Base_class):
```

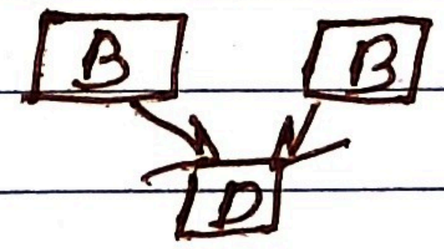
- All inherited class can access Derived class's protected and public variable.

Type of inheritance

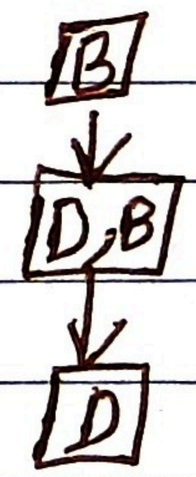
Single



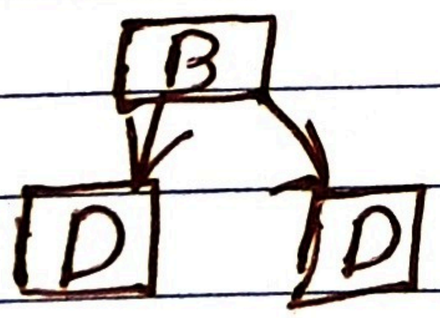
Multiple



Multilevel

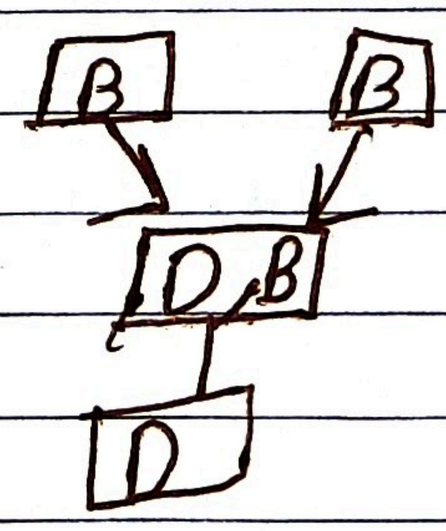


Hierarchical



Hybrid

combination of 2 or more the 2 inheritance



Date.....

Data Abstraction

Abstraction means Data hiding

—variable name Protected → class and inherit class
——variable name Private → only class access

Function Overriding → function with same name and same parameters is called function overriding.