**Python**

Contents

[Specialty of Python: 3](#_Toc527694088)

[Type inference 3](#_Toc527694089)

[Variable declaration 3](#_Toc527694090)

[Including Library or external file 3](#_Toc527694091)

[Sets, Lists, Tuples and Dictionary 3](#_Toc527694092)

[List Functions 4](#_Toc527694093)

[append 5](#_Toc527694094)

[string exist check 5](#_Toc527694095)

[Number of Elements in a List: 5](#_Toc527694096)

[Sorting forward & backward: 5](#_Toc527694097)

[Slicing (subset of a List): 5](#_Toc527694098)

[Dictionary 5](#_Toc527694099)

[When to use Dictionary or List or Set? 6](#_Toc527694100)

[How to preserve the order of pairs in a dictionary? 6](#_Toc527694101)

[Lists & Strings 7](#_Toc527694102)

[while loop 7](#_Toc527694103)

[for loop 7](#_Toc527694104)

[Functions in Python 7](#_Toc527694105)

[2D List & Nested for loop 8](#_Toc527694106)

[Exception handling 8](#_Toc527694107)

[Print function in Python: 9](#_Toc527694108)

[Differences b/w Python2 and Python3 9](#_Toc527694109)

[zip() function 9](#_Toc527694110)

[List Comprehensions: 10](#_Toc527694111)

[for loop and list comprehensions 10](#_Toc527694112)

[PEP8 10](#_Toc527694113)

[Shortest way to open a text file and display contents? 10](#_Toc527694114)

[Translator Exercise 10](#_Toc527694115)

[String Functions 11](#_Toc527694116)

[Split 11](#_Toc527694117)

[format 11](#_Toc527694118)

[Ternary if Statement 12](#_Toc527694119)

[DocString 12](#_Toc527694120)

## Specialty of Python:

Type inference

Python is a [type inferred](https://en.wikipedia.org/wiki/Type_inference) language, it can automatically infer (know) the variable type based on the assignment/initialization value.

Variable declaration

In Python, variables do not need declaration to reserve memory space. The "variable declaration" or "variable initialization" happens automatically when we assign a value to a variable.

## Including Library or external file

# from math import \*

## Sets, Lists, Tuples and Dictionary

Both Lists and Tuples allow heterogeneous data to be stored. The elements are Ordered in both these means, the elements are stored in the order it is entered. The differences are,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation** | **Set** | **Lists** | **Tuples** | **Dictionary** |
| Syntax | { } | [ ] | ( ) | { } |
| Editable | Mutable | Mutable | Immutable | Mutable |
| Order | Un-ordered | Ordered | Ordered | Un-ordered |
| Data | Heterogeneous | Heterogeneous | Heterogeneous | Key, value pair |
| Duplicates | Not allowed | Allowed | Allowed | Not allowed |
|  |  |  |  |  |

#### **Set: (Set doesn’t allow duplicates and orders the elements)**

findType = {3, 2, 1, 4, 3, 5}

print(findType)

results in, set([1, 2, 3, 4, 5])

#### [Set](https://www.programiz.com/python-programming/set) is an unordered collection of unique items. Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [] does not work.

#### **List: (duplicates allowed in List, maintains the element sequence and mutable)**

findType = [3, 2, 1, 4, 3, 5]

print(findType)

results in, [3, 2, 1, 4, 3, 5]

#### **tuple: (duplicates allowed in tuple, maintains the element sequence but unmutable)**

findType = (3, 2, 1, 4, 3, 5)

print(findType)

results in (3, 2, 1, 4, 3, 5)

Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.

fruits\_list = ["apple", "mango", "grapes", "banana"]

fruits\_tuple = ("apple", "mango", "grapes", "banana")

fruits\_set = {"apple", "mango", "grapes", "banana"}

fruits\_dict = {'a': "apple", 'm': "mango", 'g': "grapes", 'b': "banana"}

print("Fruits List is : " +str(fruits\_list))

print("Fruits Tuple is : "+str(fruits\_tuple))

print("Fruits set is : "+str(fruits\_set))

print("Fruits dictionary is: "+str(fruits\_dict))

**results in**,

Fruits List is : ['apple', 'mango', 'grapes', 'banana']

Fruits Tuple is : ('apple', 'mango', 'grapes', 'banana')

Fruits set is : set(['mango', 'banana', 'grapes', 'apple'])

Fruits dictionary is: {'a': 'apple', 'b': 'banana', 'm': 'mango', 'g': 'grapes'}

Value of two variables can be swapped using Tuples without using a third variable like,

a = 10

b = 15

print(a, b)

a, b = b, a

print(a, b)

this results in

10 15

15 10

**Note**: here the usage “ , (comma) “ represents the usage of tuple.

## List Functions

List is an ordered sequence of items. All the items in a list do not need to be of the same type. Lists are mutable.

Code:

coach\_names = ["Laks", "Uma", "Karthik", "Santio"]

print("Coaches in Agile team are: \n")

for name in coach\_names:

print(name)

results in,

Coaches in Agile team are:

Laks

Uma

Karthik

Santio

### append

coach\_names.append("Ramya")

appends “Ramya” to the list of coaches.

### string exist check

if ("Ramya" in coach\_names):

print("coach already exist")

else:

coach\_names.append("Ramya")

### Number of Elements in a List:

print("# of coaches in Agile team is: " + str(len(coach\_names)))

results in, # of coaches in Agile team is: 5

### Sorting forward & backward:

coach\_names.sort() # Ascending order

coach\_names.reverse() # reverses the list; now print will give in descending order

### Slicing (subset of a List):

List\_name[start\_index: end\_index+1]

coach\_names[2:5]

## Dictionary

# **dictionaries** - key value pairs

monthCodes = {

1 : "January",

"2" : "February",

"Mar" : "March",

"Apr" : "April",

"12" : "December",

}

print(monthCodes.get("12","Not a valid key"))

Example #2:

Capitals = {

"India" : "New Delhi",

"USA" : "Washington DC",

"UK" : "London",

"France": "Paris",

"Australia" : "Canberra"

}

for city in Capitals.keys():

print(str(city) + "=" + str(Capitals[city]))

this results in India=New Delhi

UK=London

France=Paris

USA=Washington DC

Australia=Canberra

Example #3 – Count of characters in a sentence:

str = "hi, this is Mariya"

letterCount = {}

for ch in str:

ch = ch.lower()

if ch not in letterCount:

letterCount[ch] = 1

else:

letterCount[ch] += 1

print(letterCount)

this results in, {'r': 1, 'a': 2, 'y': 1, ' ': 3, 't': 1, 'm': 1, 'i': 4, 's': 2, ',': 1, 'h': 2}

## When to use Dictionary or List or Set?

Lists – are used to store the values and maintains the insertion order

Set – are used to store the values; doesn’t maintain the insertion order

Dictionary – is used to store “Key, Value” pairs.

## How to preserve the order of pairs in a dictionary?

Using ‘OrderedDict’ subclass. It remembers the order in which the contents are added.

import collections

d = {‘A’: 1, ‘B’: 2}

d = collections.OrderedDict()

## Lists & Strings

How to convert a String into letter? – use List.

name = "Mariya Preeths"

print(list(name))

results in ['M', 'a', 'r', 'i', 'y', 'a', ' ', 'P', 'r', 'e', 'e', 't', 'h', 's']

How to combine individual letters into a String? – use join.

name = ['M', 'a', 'r', 'i', 'y', 'a', ' ', 'P', 'r', 'e', 'e', 't', 'h', 's']

print("".join(name))

results in Mariya Preeths

## while loop

i = 1;

while i <=10:

print(i)

i=i+2

## for loop

friends = ["Uma", "Sathya", "Sudharsan", "Surya"]

for name in friends:

print(name)

for index in range(1, 10):

print(index)

Key difference of ‘for loop’ syntax b/w Python and Java (or any other language):

In Java or other languages, the syntax of ‘for loop’ is:

for(initialize; condition check; iterate)

{

………..

………..

}

whereas in Python, for loop iterates over any sequences (List or String or Range, etc.,)

eg:

for i in range(5):

print(i)

## Functions in Python

def <func\_name> :

<stmt 1>

<stmt 2>

.

.

.

<stmt n>

### Function Arguments

You can call a function by using the following types of formal arguments −

* Required arguments
* Keyword arguments
  + def myFunc(name, age):

myFunc(age=46, name=”Mariya”)

* Default arguments
  + def myFunc(name, age=46):

myFunc(“Mariya”)

* Variable-length arguments
  + def myFunc(arg1, \*vararg):

myFunc(10)

myFunc(30, 50, 70)

### The Anonymous Functions

These functions are called anonymous because they are not declared in the standard manner by using the def keyword. You can use the lambda keyword to create small anonymous functions.

sum = lambda arg1, arg2: arg1 + arg2

print(“Sum of two values is: “, sum(10, 20))

2D List & Nested for loop

number\_grid = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9],

[0]

]

for row in number\_grid:

for col in row:

print(col)

Exception handling

try:

number = int(input("Enter a number\n"))

print("entered num is: " + str(number))

except ValueError as err:

print(err)

Eg: Sum the digits of a given number:

try:

num1=input("Enter a number")

except ValueError as err1:

print(err1)

sum=0

print("digits of given number is: ")

while(num1>0):

num2=int(num1)%10

sum=sum+num2

num1=int(num1)//10

print(num2)

print("Sum of digits of given number is: " + str(sum))

Eg: last\_name is missing or adding int and str types error handling:

agile\_coaches = {

"name": "Laks",

"skill": "trainer",

"qual": "B.E"

}

agile\_coaches["last\_name"] = "Ram"

try:

last\_name = agile\_coaches["last\_name"]

print(last\_name)

numbered\_last\_name = 3 + last\_name

except KeyError:

print("last\_name is not defined")

except TypeError:

print("You can't add an int and str types")

except Exception as error:

print("Exception is: ", error)

print("Exception code works!")

results in Ram

You can't add an int and str types

Exception code works!

Print function in Python:

The actual syntax of the print() function is

print(\*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

Thus, print function adds a space between the resulting output values. This can be changed to any character like,

print(1, 2, 3, 4, sep=’\*’)

results in, 1\*2\*3\*4

Differences b/w Python2 and Python3

|  |  |
| --- | --- |
| **Python2** | **Python3** |
| Division Operator  43/2 = 21 | Division Operator  43/2 = 21.5  43//2 = 21 |
| Print is a Statement  print ‘Hello!’ | Print is a function  Print(“Hello!”) |
| range() returns lists | range() returns iterators or generators |

zip() function

It combines two lists and produces a list of tuples.

eg:

list 1 = [1, 3, 5, 7]

list 2 = [5, 7, 2, 6]

zip(list1, list2) gives = [1, 5, 3, 7, 5, 2, 7, 6]

List Comprehensions:

p = [1, 2, 3, 4, 5]

q = [9, 8, 7, 6]

r = [5, 6, 7]

print([x+y+z for x,y,z in zip(p,q,r)])

results in [15, 16, 17]

for loop and list comprehensions

for loop equivalent can be simplified in list comprehensions as,

nums = [1, 2, 3, 4, 5]

my\_list = []

for n in nums:

my\_list.append(n)

print (my\_list)

the same above code can be simplified in ‘list comprehension’ as,

nums = [1, 2, 3, 4, 5]

my\_list = [n for n in nums]

print(my\_list)

PEP8

It’s a style guide for Python code. Standard guideline for Python coding.

Shortest way to open a text file and display contents? – using ‘with’ operator.

with open(“file-name”, “r”) as fp

fileData = fp.read()

print(fileData)

what ‘with’ operator does here is, it takes care of closing the file as well automatically.

The other option is,

print (open (“filename.txt”).read())

Translator Exercise (change any vowels into ‘g’):

def translate(phrase):

translation = ""

for letter in phrase:

if letter in "AEIOUaeiou":

translation = translation + "g"

else:

translation = translation + letter

return translation

print(translate("Mariya Preeths"))

String Functions

### Split

Splits a sentence into words

Code:

sentence = "this is a six words sentence"

print(sentence.split( ))

results in, ['this', 'is', 'a', 'six', 'words', 'sentence']

### format

format substitutes the given arguments in a sentence

Code:

name1 = "Mariya"

name2 = "Preeths"

print("Nice to meet you {0}, this is {1}.".format(name1, name2))

print(f"Nice to meet you too {name2}, {name1} is happy now!")

results in,

Nice to meet you Mariya, this is Preeths.

Nice to meet you too Preeths, Mariya is happy now!

a=10; b=20

print(“Value of a is {} and value of b is {}” .format(a,b)

results in, value of a is 10 and value of b is 20

name = "Mariya"

greeting = "Good Morning"

print("Hello {}, {}" .format(name, greeting))

results in, Hello Mariya, Good Morning

print("Hello {name}, {greeting}" .format(greeting="How are you?", name="Mariya"))

results in, Hello Mariya, How are you?

### Ternary if Statement

Code:

num1 = input("Enter first num")

num2 = input("Enter second num")

print("Bigger" if num1 > num2 else "smaller")

if the input is, 10 and 20

results in, smaller

DocString

DocString is short for documentation string

It is a string that occurs as the first statement in a module, function, class or method definition. It describes what a function/class does in that string. Triple quotes are used while writing docstrings. Docstring is available to us as the attribute \_\_doc\_\_ of the function.

Code:

def double(num):

"""Function to double the value"""

print 2\*num

print(double.\_\_doc\_\_)

results in, Function to double the value

**Variable** – a variable is a named location used to store data in memory.

Find the largest of 3 numbers:

a=10; b=15; c=25

if (a>b) and (a>c):

print("a is the largest")

elif(b>a) and (b>c):

print("b is the largest")

elif(a==b) and (b==c) and (a==c):

print("all 3 numbers are equal")

elif(a==b):

print("a and b are equal and largest")

elif(b==c):

print("b and c are equal and largest")

elif(a==c):

print("a and c are equal and largest")

else:

print("c is the largest")

Check if a given name exists in the given list:

*bird\_name = input("Enter a bird name\n")*

*birds = ["Crow", "Pigeon", "Sparrow", "Eagle", "hen"]*

*def check\_bird(name):*

*x = [bird.lower() for bird in birds]*

*if name.lower() in x:*

*print(name, " exists in birds list")*

*else:*

*print(name, " doesn't exist in birds list")*

*check\_bird(bird\_name)*

Functions returning values & printing them

*def get\_name():*

*name\_entry = input("enter a name\n")*

*return name\_entry*

*def get\_greeting():*

*greeting\_entry = input("enter a greeting\n")*

*return greeting\_entry*

*def make\_greeting(name, greeting = "Hello "):*

*return (greeting + " " + name + "!")*

*print(make\_greeting(get\_name(), get\_greeting()))*

Returning multiple values from a function:

<https://www.geeksforgeeks.org/g-fact-41-multiple-return-values-in-python/>

Python Links:

In Tamil:

<https://www.youtube.com/watch?v=Y68_rD2VfFY>

very good tutorial for Python:

<https://youtu.be/rfscVS0vtbw>

List & Tuples:

<https://www.youtube.com/watch?v=3dt4OGnU5sM>

CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans are Apart.

<https://www.youtube.com/watch?v=JdOHv-LF-nA&index=4&list=PL5mHZvl6c0Wk3i6ciBVti7dkZfnwyjeAG>

Learn Python:

<https://www.programiz.com/python-programming/first-program>

NPTEL Course:

<https://www.youtube.com/watch?v=JdOHv-LF-nA>

<https://www.youtube.com/watch?v=jCzT9XFZ5bw>

adding GUI to your Python:

<https://opensource.com/article/18/8/pysimplegui>

the joy of programming in Python:

<https://nptel.ac.in/courses/106106182/>

<https://www.programiz.com/python-programming/operators>

Learn Python:

[https://www.tutorialspoint.com/python3/](https://www.youtube.com/redirect?redir_token=dShQ7WtI_mmAFQ8Z2-bEGLqbbBV8MTUzODI4OTIzMkAxNTM4MjAyODMy&q=https%3A%2F%2Fwww.tutorialspoint.com%2Fpython3%2F&event=comments)

Hitesh Courses:

<https://courses.learncodeonline.in/>

ML & Data Science Bootcamp Course:

<https://courses.learncodeonline.in/learn/Machine-Learning-Bootcamp>?

Good Video for Machine Learning (SimplyLearn):

<https://www.youtube.com/watch?v=7JhjINPwfYQ>

<https://www.youtube.com/watch?v=IpGxLWOIZy4>

Skills required to learn ML:

<https://www.youtube.com/watch?v=DZ7xuZ1-uh8>

<https://www.youtube.com/watch?v=GHPMgEC5Q30&rel=1&color1=0xcbba9f&color2=0xcbba9f&border=0&fs=1>

<https://nptel.ac.in/courses/106106182/50>

Good source to learn Python Programming

<https://www.edx.org/course/introduction-to-python-absolute-beginner-1>

<https://courses.edx.org/courses/course-v1:Microsoft+DEV236x+3T2018/courseware/372fb285386c4bfe85611cd45a96b325/0384b34b15a445b3ba5a499794cd0575/?child=first>

<https://swayam.gov.in/courses/4178-spoken-tutorial-python-english/viewcourseware?cinfo=MP7Z4Xew36%252fT7XrKt%252bvHQxQK2iykhw5aHBaAHt%252fGABZCP%252f7yejIRMvmGJ2xo2t%252b9n8Zw9KrC236oOlERWXlf72NcReyBvoIh8IuOQe5qOFuMd4Q3QJO7GqmfsDiwrj7wgLjeLr2DImlF%252bWPy0QcdBr0QLKJNY955imJPYpFDwys%253d>

<https://www.tutorialspoint.com/python3/python_modules.htm>