CHRIST THE KING BOYS MATRIC HR. SEC. SCHOOL, KUMBAKONAM – 612 001

LIST OF PYTHON FUNCTIONS
I. BUILT-IN FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
range()	**range(start,stop,[step])**	start – refers to the initial value stop – refers to the final value step – refers to increment value, this is optional part.	range(1,10,1) – will start the range values from 1 to 9
abs()	**abs(n)**	Returns absolute value of a number	a= -4 print(abs(a)) **Output:** 4
ord()	**ord(c)**	Returns the ASCII Value for the Unicode character	a= 'a' print(ord(a)) **Output:** 97
chr()	**chr(c)**	Returns the Unicode Character for the ASCII value	a= 98 print(chr(a)) **Output:** B
bin()	**bin(n)**	Returns the binary string prefixed with "0b" for the given integer.	a= 15 print(bin(a)) **Output:** 0b1111
type()	**type(object)**	Returns Type of an Object	a= 4 print(type(a)) **Output:** <class 'int'>
id()	**id(object)**	Returns Identify of an Object	a= 4 print(id(a)) **Output:** 1354785672
min()	**min(list)**	Return the Minimum value in a list	a= [1,2,3,4] print(min(a)) **Output:** 1

max()	**max(list)**	Return the Maximum value in a list	a= [1,2,3,4] print(max(a)) **Output:** 4
sum()	**sum(list)**	Return the Sum of values in a list	a= [1,2,3,4] print(sum(a)) **Output:** 10
format()	**format(n,[f])**	Returns the output based on the given format (i) Binary, (ii) Octal, (iii) Fixed – Point Notation	a= 22 print(format(a,'b') print(format(a,'o') print(format(a,'f') **Output:** 10110 26 22.000000
round()	**round(n,[n digits])**	Returns the Nearest integer to its input	a= 4.9 print(round(a)) **Output:** 5
pow()	**pow(a,b)**	Returns the computation of ab i.e., (a**b) a is raised to the power of b.	a=2 b=3 print(pow(a,b)) **Output:** 8
eval()	**eval()**	This function is used to evaluate the value passed to the function	>>>a=eval(input("Enter the expression:")) Enter the expression: 3*4/2 >>>a 6.0
next()	**next()**	The next() function returns the next item from the iterator. It can also be used to skip a row of the csv file.	import csv f=open("c:\pyprg\sample.csv",'r') reader = csv.reader(f) next(reader)
open()	**open(fileobject,mode)**	This function returns a file object also called a handle. It is used to	f=open("c:\pyprg\sample.csv",'r')

		read or modify the file accordingly.	
close()	**f.close()**	Closing a file will free up the resources that were tied with the file	f=open("c:\pyprg\sample.csv",'r') f.close()
print()	**print("String to be displayed")** **print(variable)**	This function is used to display the result on the screen.	print("Welcome") **Output:** Welcome
input()	**variable=input("Prompt String")**	This function is used to accept data as input at runtime.	>>>city=input("Enter your City:") Enter your City: Madurai
writerow()	**writerow()**	This function writes one row at a time.	writer.writerow(csvData)
writerows()	**writerows()**	This function writes all the data at once.	writer.writerows(csvData)
dictreader()	**dictreader()**	This function works similar to reader() but creates an object which maps data to a dictionary. DictReader works by reading the first line of the CSV and using each comma separated value in this line as a dictionary key.	import csv filename = 'c:\pyprg\sample8.csv' file=csv.DictReader(open(filename,'r'))
dictwriter()	**dictwriter()**	This function works with dictionary. It takes additional argument field names that are used as dictionary keys.	w = csv.DictWriter(CF, fieldnames=fields)
fetchone()	**cursor.fetchone()**	This function returns the next row of a query result set or None in case there is no row left.	res=cursor.fetchone()
fetchall()	**cursor.fetchall()**	This function is to fetch all the rows from the database table	res=cursor.fetchall()
fetchmany()	**cursor.fetchmany()**	This function that returns the next number of rows(n) of the result set.	res=cursor.fetchmany(3)
reader()	**csv.reader(fileobject,delimiter,fmtparams)**	The reader function is designed to take each line of the file and make a list of all columns.	reader=csv.reader(F)
writer()	**csv.writer(fileobject,delimiter,fmtparams)**	The writer function returns a writer object which converts the user's data into delimited strings on the given file-like object.	writer=csv.writer(F)
register_dialect()	**csv.register_dialect()**	A dialect describes the format of the csv file that is to be read. A dialect is a class of csv module which helps to define parameters for reading and writing CSV. It allows you to create, store, and re-use various formatting parameters for your data.	csv.register_dialect('myDialect',delimiter = ',',skipinitialspace=True)

II. MATHEMATICAL FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
floor	**math.floor(x)**	Returns the largest integer less than or equal to x	import math x=26.7 y= -26.7

			print(math.floor(x)) print(math.floor(y) **Output:** 26 -27
ceil()	**math.ceil(x)**	Returns the smallest integer greater than or equal to x	import math x=26.7 y= -26.7 print(math.floor(x)) print(math.floor(y) **Output:** 27 -26
sqrt()	**math.sqrt(x)**	Returns the square root of x. x must be greater than 0(zero).	import math x=16 print(math.sqrt(x)) **Output:** 4

III. BUILT- IN STRING FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
len()	**len(str)**	Returns the length of the string	a=”Python” print(len(a)) **Output:** 6
capitalize()	**capitalize()**	Used to capitalize the first character of the string	a=”python” print(a.capitalize()) **Output:** Python
center()	**center(width,fillchar)**	Returns a string with the original string centered to a total of width columns and filled with fill char in columns that do not have characters	a=”Python” print(a.center(10,’*’)) **Output:** **Python**
find()	**find(sub,[start,[end]])**	It is used to search the first occurrence of the sub string in the given	a=”Python”

		string. It returns the index at which the substring starts. It returns -1 if the substring does not occur in the string.	print(a.find('P')) **Output:** 0 a="Python" print(a.find('t',0,4)) **Output:** 2 a="Python" print(a.find('a')) **Output:** -1
isalnum()	**isalnum()**	Returns True if the string contains only letter and digit, otherwise returns False.	a="Python" print(a.isalnum()) **Output:** False a="12Python" print(a.isalnum()) **Output:** True
isalpha()	**isalpha()**	Returns True if the string contains only letters, otherwise returns False.	a="Python" print(a.isalpha()) **Output:** True a="12Python" print(a.isalpha()) **Output:** False
isdigit()	**isdigit()**	Returns True if the string contains only digits, otherwise returns False.	a="Python" print(a.isdigit()) **Output:** False a="123" print(a.isdigit()) **Output:**

			True
lower()	**lower()**	Returns the exact copy of the string with all the letters in lowercase.	a=”PYTHON” print(a.lower()) **Output:** python
upper()	**upper()**	Returns the exact copy of the string with all the letters in uppercase.	a=”python” print(a.upper()) **Output:** PYTHON
isupper()	**isupper()**	Returns True if the string is in uppercase, otherwise False	a=”PYTHON” print(a.isupper()) **Output:** True
islower()	**islower()**	Returns True if the string is in lowercase, otherwise False	a=”python” print(a.islower()) **Output:** True
title()	**title()**	Returns a String in Title Case	a=”python programming” print(a.title()) **Output:** Python Programming
swapcase()	**swapcase()**	It will change the case of every character to its opposite case.	a=”PyThOn” print(a.swapcase()) **Output:** pYtHoN
count()	**count(str,beg,end)**	Returns the number of substrings occurs within the given range. Remember that substring may be a single character. beg and end arguments are optional. If it is not given, python searched in whole string. Search is case sensitive.	a=”Raja Raja” print(a.count(‘Raja’)) print(a.count(‘r’)) print(a.count(‘a’,0,5)) **Output:** 2 0 2
ord()	**ord(char)**	Returns the ASCII code of the character	a= ‘a’

			print(ord(a)) **Output:** 97
chr()	**chr(ascii)**	Returns the character represented by a ASCII	a= 98 print(chr(a)) **Output:** B

IV. LIST FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
len()	**len(list)**	To find the length of the list.	a=[1,2,3,4] print(len(a)) **Output:** 4
append()	**list.append(element to be added)**	Add single element to the List	a=[1,2,3] a.append(4) print(a) **Output:** [1,2,3,4]
extend()	**list.extend([elements to be added])**	Add more than one element to an existing list	a=[1,2,3] a.extend([4,5,6]) print(a) **Output:** [1,2,3,4,5,6]
insert()	**list.insert(index,element to be inserted)**	Insert an element at any position of a List	a=[1,3,4] a.insert(1,2) print(a) **Output:** [1,2,3,4]
remove()	**list.remove(element)**	Used to delete one or more elements if the index value is unknown	a=[1,2,3,4] a.remove(2) print(a) **Output:** [1,3,4]

pop()	**list.pop(index of an element)**	Deletes and returns the last element of a list if the index is not given	a=[1,2,3,4] a.pop(0) **Output:** 1
clear()	**list.clear()**	Deletes all the elements in the list	a=[1,3,4] a.clear() print(a) **Output:** [ ]
list()	**list()**	Creates list in Python	a=(1,2,3,4) b=list(a) print(b) **Output:** [1,2,3,4]
copy()	**list.copy()**	Returns a Copy of the List	a=[1,2,3,4] b=a.copy() print(b) **Output:** [1,2,3,4]
count()	**list.count(value)**	Returns the number of similar elements present in the list.	a=[1,2,2,3,4] b=print(a.count(2)) print(b) **Output:** 2
index()	**list.index(element)**	Returns the index value of the first recurring element	a=[11,12,13,14] b=print(a.index(12)) print(b) **Output:** 1
reverse()	**list.reverse()**	Reverses the order of elements in the list	a=[1,2,3,4] a.reverse() print(a) **Output:** [4,3,2,1]

sort()	**sort(reverse=True|False, key=myFunc)**	Sorts the elements in list Ascending order is by default If reverse is True, the list will be in descending order, otherwise ascending order. key=myFunc myFunc is the user defined function that specifies the sorting criteria	a=['Orange','Apple','Banana'] a.sort() print(a) a.sort(reverse=True) print(a) **Output:** ['Apple','Banana','Orange'] ['Orange','Banana','Apple']
max()	**max(list)**	Returns the maximum value in a list	a=[1,2,3,4,5] print(max(a)) **Output:** 5
min()	**min(list)**	Returns the minimum value in a list	a=[1,2,3,4,5] print(min(a)) **Output:** 1
sum()	**sum(list)**	Returns the sum of values in a list	a=[1,2,3,4,5] print(sum(a)) **Output:** 15

V. TUPLE FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
tuple()	**tuple()**	Creates Tuples from the list	a=[1,2,3,4] b=tuple(a) print(b) **Output:** (1,2,3,4)
type()	**type(object)**	To know the data type of the python object	a=(1,2,3,4) print(type(a)) **Output:** <class 'tuple'>

VI. SET FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
set()	**set()**	A list or tuple can be converted as set.	a=[1,2,3,4] b=set(a) print(b) **Output:** {1,2,3,4}
union()	**union()**	It is used to joins two sets in python	a={1,2,3} b={4,5,6} print(a.union(b)) **Output:** {1,2,3,4,5,6}
intersection()	**intersection()**	It is used to intersect two sets in python	a={1,2,3} b={3,4,5} print(a.intersection(b)) **Output:** {3}
difference()	**difference()**	It is used to do the difference of two sets in python	a={1,2,3} b={3,4,5} print(a.difference(b)) **Output:** {1,2}
symmetricdifference()	**symmetricdifference()**	It is used to do the symmetric difference of two sets in python	a={1,2,3} b={3,4,5} print(a.symmetric_difference(b)) **Output:** {1,2,4,5}

VII. DICTIONARY FUNCTIONS

FUNCTION	SYNTAX	DESCRIPTION	EXAMPLE
clear()	**clear()**	Used to delete all the elements in the dictionary	a= {'Roll No': 12001,'SName':'Meena','Mark1': 98,'Mark2': 86} a.clear() **Output:** { }