**EXPERIMENT NO : 2D**

**Python Programs To Implement Functions (Built-in, User Defined ,Anonymous).**

**Aim :- python programs to implement Functions (Built-in, User Defined, Anonymous).**

# THEORY:

##### OUTPUT:

*Python 3.11.0a4 (main, Jan 17 2022, 12:57:32) [MSC v.1929 32 bit (Intel)] on win32*

*Type "help", "copyright", "credits" or "license()" for more information. #AKASH YADAV ID.NO:VU4F2122016 EXP:1C DATE:31/1/2023*

Built in function

# 1#Python abs() Function

##### #Definition and Usage

**The abs() function returns the absolute value of the specified number.**

***Syntax abs(n)***

***#CREATING VARIABLE***

***x=abs(3+5j) print(x)***

***5.830951894845301***

**

**#Python all() Function**

**#Definition and Usage**

**The all() function returns True if all items in an iterable are true, otherwise it returns False. If the iterable object is empty, the all() function also returns True.**

**Syntax all(iterable) #list a1=[1,1,1]**

**x=all(a1) print(x) True #sets**

**a1={1,0,1,0}**

**x=all(a1) print(x)**

**False**

**# Returns False because both the second and the forth items are False #tuple**

**a1=(0,True,False) x=all(a1)**

**print(x)**

**False**

**# Returns False because both the first and the third items are False #dict**

**a1={0:"akash",b:"yadav"}**

**a1={0:'akash',1:'yadav'}**

**x=all(a1) print(x)**

**False**

**## Returns False because the first key is false.**

**# For dictionaries the all() function checks the keys, not the values.**

# #Python any() Function

#### #Definition and Usage

***The any() function returns True if any item in an iterable are true, otherwise it returns False. If the iterable object is empty, the any() function will return False.***

***Syntax***

***any(iterable)***

###### #CREATING VARIABLE

***a1=(1,0,1,False) x=any(a1) print(x)***

***True***

**#Python complex() Function**

###### #Definition and Usage

***The complex() function returns a complex number by specifying a real number and an imaginary number.***

###### Syntax

***complex(real, imaginary)***

###### #CREATING VARIABLE

***a1=complex(3,7) print(a1)***

***(3+7j)***

***#Python dict() Function***

###### #Definition and Usage

***The dict() function creates a dictionary.***

***A dictionary is a collection which is unordered, changeable and indexed. Read more about dictionaries in the chapter: Python Dictionaries.***

###### Syntax

***dict(keyword arguments)***

###### #CREATING VARIABLE

***a1=dict(NAME="AKASH YADAV",AGE=21,COUNTRY="INDIA")***

***print(a1)***

***{'NAME': 'AKASH YADAV', 'AGE': 21, 'COUNTRY': 'INDIA'}***

## #Python dir() Function

###### EXAMPLE:

***class person:***

***name="akash yadav" age=21 country="india"***

***print(dir(person))***

***[' class ', ' delattr ', ' dict ', ' dir ', ' doc ', ' eq ', ' format ', ' ge ',***

***' getattribute ', ' getstate ', ' gt ', ' hash ', ' init ', ' init\_subclass ', ' le ', ' lt ', ' module ', ' ne ', ' new ', ' reduce ', ' reduce\_ex ', ' repr ',***

***' setattr ', ' sizeof ', ' str ', ' subclasshook ', ' weakref ', 'age', 'country', 'name']***

***#Python divmod() Function***

###### #Definition and Usage

***The divmod() function returns a tuple containing the quotient and the remainder when argument1 (dividend) is divided by argument2 (divisor).***

###### Syntax

***divmod(dividend, divisor)***

###### #CREATING VARIABLE

***a1=divmod(7,2) print(a1)***

***(3, 1)***

### #Python enumerate() Function

###### #Definition and Usage

***The enumerate() function takes a collection (e.g. a tuple) and returns it as an enumerate object. The enumerate() function adds a counter as the key of the enumerate object.***

***Syntax enumerate(iterable, start)] #CREATING VARIABLE***

***x=("akash","suraj","viram") a1=enumerate(x)***

***print(a1)***

***<enumerate object at 0x000001A6E38BA250>***

### #Python eval() Function

###### #Definition and Usage

***The eval() function evaluates the specified expression, if the expression is a legal Python statement, it will be executed.***

###### Syntax

***eval(expression, globals, locals)***

###### #CREATING VARIABLE

***a1="print(57)" eval(a1)***

***57***

### #Python exec() Function

###### #Definition and Usage

***The exec() function executes the specified Python code.***

***The exec() function accepts large blocks of code, unlike the eval() function which only accepts a single expression***

###### Syntax

***exec(object, globals, locals)***

###### #CREATING VARIABLE

***a1 = 'name = "akash yadav"\nprint(name)' exec(a1)***

***akash yadav***

### #Python float() Function

###### #Definition and Usage

***The float() function converts the specified value into a floating point number.***

***Syntax float(value) print(float(7)) 7.0***

### #Python format() Function

###### #Definition and Usage

***The format() function formats a specified value into a specified format.***

###### Syntax

***format(value, format)***

###### #CREATING VARIABLE

***x=format(0.5,"%") print(x) 50.000000%***

### #Python getattr() Function

###### #Definition and Usage

*The getattr() function returns the value of the specified attribute from the specified object.*

*Syntax*

*getattr(object, attribute, default)*

###### #CREATING VARIABLE

*name="AKASH" age=21 country="india"*

*x=getattr(akash,'age') print(x)*

*21*

*print(getattr(akash,'name')) AKASH*

### #Python globals() Function

###### #Definition and Usage

*The globals() function returns the global symbol table as a dictionary.*

*A symbol table contains necessary information about the current program*

###### Syntax

*globals()*

###### #CREATING VARIABLE

*a1=globals() print(a1)*

*{' name ': ' main ', ' doc ': None, ' package ': None, ' loader ':*

*<class '\_frozen\_importlib.BuiltinImporter'>, ' spec ': None,*

*' annotations ': {}, ' builtins ': <module 'builtins' (built-in)>, 'akash':*

*<class ' main .akash'>, 'x': 21, 'a1': {...}}*

### #Python hasattr() Function

###### #Definition and Usage

*The hasattr() function returns True if the specified object has the specified attribute, otherwise False.*

*Syntax*

*hasattr(object, attribute)*

###### #CREATING VARIABLE

*class akash*

*SyntaxError: incomplete input class akash:*

*name="AKASH YADAV"*

*age=21 occupation="HR"*

*print(hasattr(akash,'name')) True print(hasattr(akash,'age'))*

*True print(hasattr(akash,'ocupation')) False print(hasattr(akash,'occupation'))*

*True*

### #Help function in Python

***#The Python help function is used to display the documentation of modules, functions, classes, keywords, etc.***

*The help function has the following syntax: help([object])*

***EXAMPLE:***

*help(print)*

*Help on built-in function print in module builtins:*

*print(\*args, sep=' ', end='\n', file=None, flush=False) Prints the values to a stream, or to sys.stdout by default.*

*sep*

*string inserted between values, default a space. end*

*string appended after the last value, default a newline. file*

*a file-like object (stream); defaults to the current sys.stdout. flush*

*whether to forcibly flush the stream.*

### #Python hex() Function

###### #Definition and Usage

*The hex() function converts the specified number into a hexadecimal value. The returned string always starts with the prefix 0x.*

###### Syntax

*hex(number)*

###### #CREATING VARIABLE

*x=hex(101) print(x) 0x65*

*print(hex(5)) 0x5 print(hex(16)) 0x10*

### #Python id() Function

###### #Definition and Usage

*The id() function returns a unique id for the specified object. All objects in Python has its own unique id.*

*The id is assigned to the object when it is created.*

*The id is the object's memory address, and will be different for each time you run the program. (except for some object that has a constant unique id, like integers from -5 to 256)*

***Syntax***

*id(object)*

###### #CREATING VARIABLE

*a1=("akash","suraj","viram") print(id(a1)) 1697882073024*

### #Python input() Function

###### #Definition and Usage

*The input() function allows user input.*

*Syntax input(prompt)*

###### #CREATING VARIABLE

*print("enter your name") enter your name x=input()*

*Akash Yadav print('Hello,'+x)*

*Hello,Akash Yadav*

### #Python int() Function

###### #Definition and Usage

*The int() function converts the specified value into an integer number.*

***Syntax***

*int(value, base)*

###### #CREATING VARIABLE

*a1=int(5.7) print(a1)*

*5*

*print(int(8.9)) 8*

### #Python isinstance() Function

###### #Definition and Usage

*The isinstance() function returns True if the specified object is of the specified type, otherwise False.*

*If the type parameter is a tuple, this function will return True if the object is one of the types in the tuple.*

***Syntax***

*isinstance(object, type)*

###### #CREATING VARIABLE

*a1=isinstance(8,int) print(a1)*

*True*

### #Python iter() Function

###### #Definition and Usage

*The iter() function returns an iterator object.*

*Syntax*

*iter(object, sentinel)*

###### #CREATING VARIABLE

*x=iter(["akash","viram","yadav"]) print(next(x))*

*akash print(next(x)) viram print(next(x)) yadav*

### #Python len() Function

###### #Definition and Usage

*The len() function returns the number of items in an object.*

*When the object is a string, the len() function returns the number of characters in the string.*

***Syntax***

*len(object)*

###### #CREATING VARIABLE

*a1=["akash","viram","yadav"] print(len(a1))*

*3*

### #Python list() Function

###### #Definition and Usage

*The list() function creates a list object.*

*A list object is a collection which is ordered and changeable. Read more about list in the chapter: Python Lists.*

***Syntax***

*list(iterable)*

###### #CREATING VARIABLE

*a1=(("akash","viram","yadav")) print(list(a1))*

*['akash', 'viram', 'yadav']*

### #Python max() Function

###### #Definition and Usage

*The max() function returns the item with the highest value, or the item with the highest value in an iterable.*

*If the values are strings, an alphabetically comparison is done.*

***Syntax***

*max(n1, n2, n3, ...)*

###### #CREATING VARIABLE

*print(max(5,6,8,12,23)) 23*

*print(max(78,95,343,5,6,78,565,65,654,6445,345,5454,343,45,67,76,86)) 6445*

### #Python min() Function

###### #Definition and Usage

*The min() function returns the item with the lowest value, or the item with the lowest value in an iterable.*

*If the values are strings, an alphabetically comparison is done.*

***Syntax***

*min(n1, n2, n3, ...)*

***#CREATING VARIABLE*** *print(min(121,233,434,344,555,543,34434,566,65654,54651,31,2,544,45564)) 2*

### #Python next() Function

###### #Definition and Usage

*The next() function returns the next item in an iterator.*

*You can add a default return value, to return if the iterable has reached to its end.*

***Syntax***

*next(iterable, default)*

###### #CREATING VARIABLE

*mylist = iter(["akash", "suraj", "viram"])*

*print(next(mylist)) akash print(next(mylist))*

*suraj print(next(mylist))*

*viram*

### #Python oct() Function

###### #Definition and Usage

*The oct() function converts an integer into an octal string. Octal strings in Python are prefixed with 0o.*

***Syntax***

*oct(int)*

###### #CREATING VARIABLE

*print(oct(16)) 0o20*

*print(oct(1)) 0o1*

### #Python ord() Function

###### #Definition and Usage

*The ord() function returns the number representing the unicode code of a specified character.*

***Syntax***

*ord(character)*

###### #CREATING VARIABLE

*print(ord("A")) 65*

*print(ord("a")) 97*

### #Python pow() Function

###### #Definition and Usage

The pow() function returns the value of x to the power of y (xy).

If a third parameter is present, it returns x to the power of y, modulus z.

**Syntax**

pow(x, y, z)

###### #CREATING VARIABLE

print(pow(2,2)) 4

print(pow(3,2)) 9

### #Python range() Function

###### #Definition and Usage

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

**Syntax**

range(start, stop, step)

###### #CREATING VARIABLE

a1=range(7) for n in a1:

print(n)

0

1

2

3

4

5

6

#User defined functions

*All the functions that are written by any us comes under the category of user defined functions. Below are the steps for writing user defined functions in Python.*

*In Python****, def keyword*** *is used to declare user defined functions.*

*An indented block of statements follows the function name and arguments which contains the body of the function.*

***Syntax:***

*def function\_name(): statements*

*.*

*.*

###### #EXAMPLE

def AKASH():

print("user define function with no argument!") AKASH()

user define function with no argument!

### #Parameterized Function

*The function may take arguments(s) also called parameters as input within the opening and closing parentheses, just after the function name followed by a colon.*

*Syntax:*

*def function\_name(argument1, argument2, ...): statements*

*.*

*.*

###### #EXAMPLE

*def EvenOdd(a): if(a % 2 == 0):*

*print("ENTERED NUMBER IS EVEN:")*

*def EvenOdd(a): if(a % 2 == 0):*

*print("ENTERED NUMBER IS EVEN:")*

*else:*

*print("ENTERED NUMBER IS ODD:")*

*EvenOdd(2)*

*ENTERED NUMBER IS EVEN:*

*EvenOdd(15)*

*ENTERED NUMBER IS ODD:*

*EvenOdd(100)*

*ENTERED NUMBER IS EVEN:*

*EvenOdd(55.5)*

*ENTERED NUMBER IS ODD:*

*EvenOdd(59.9)*

*ENTERED NUMBER IS ODD:*

*EvenOdd(60.9)*

*ENTERED NUMBER IS ODD:*

*EvenOdd(60.0)*

*ENTERED NUMBER IS EVEN:*

### #Default arguments

A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument.The following example illustrates Default arguments.

###### #EXAMPLE

*# default arguments*

def akash(x,y=10): print("x:",x)

print("y:",y)

akash(20) x: 20

y: 10

akash(60) x: 60

y: 10

akash(55) x: 55

y: 10

akash(101) x: 101

y: 10

### #Keyword arguments

The idea is to allow caller to specify argument name with values so that caller does not need to remember order of parameters.

###### Example:

*def emplloye(firstname,midname,lastname): print(firstname,midname,lastname)*

*emplloye(firstname="akash",midname="ramkrit",lastname="yadav") akash ramkrit yadav*

### #Variable length arguments

We can have both normal and keyword variable number of arguments.

The special syntax \*args in function definitions in Python is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list.

The special syntax \*\*kwargs in function definitions in python is used to pass a keyworded, variable-length argument list. We use the name kwargs with the double star. The reason is because the double star allows us to pass through keyword arguments (and any number of them).

##### Example:

*# \*args and \*\*kwargs*

*def f1(\*argv):*

*for arg in argv: print (arg)*

*def f2(\*\*kwargs):*

*for key, value in kwargs.items():*

*print ("% s == % s" %(key, value))*

*#output*

*f1("my name ,is AKASH YADAV ") my name ,is AKASH YADAV*

*f1("akash","suraj","viram") akash*

*suraj viram*

*f2(firstname="akash",lastname="yadav") firstname == akash*

*lastname == yadav*

### #Pass by Reference or pass by value

*One important thing to note is, in Python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created. Parameter passing in Python is same as reference passing in Java. To confirm this Python’s built-in id() function is used in below example.*

*Example:*

###### #EXAMPLE

*# verify pass by reference def f1(x):*

*print("Value received:", x, "id:", id(x))*

*# Driver's code x = 12*

*print("Value passed:", x, "id:", id(x)) Value passed: 12 id: 140735330051208 f1(x)*

*Value received: 12 id: 140735330051208*

### #Function with return value

*A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller. The statements after the return statements are not executed. If the return statement is without any expression, then the special value None is returned.*

**Syntax:**

def fun():

statements return [expression]

***#EXAMPLE***

# demonstrate return statement def add(a, b):

# returning sum of a and b return a + b

def is\_true(a):

# returning boolean of a return bool(a)

# calling function res = add(2, 3)

print("Result of add function is {}".format(res))

Result of add function is 5 res = is\_true(3<5)

print("\nResult of is\_true function is {}".format(res)) Result of is\_true function is True