Aswany

August 16, 2024

[3]:
```
print("Demo of basic data types:Numbers")
x=3
y=2.5
print("x=",x)
print("y=",y)
print("Data type of variable x:",type(x))
print("Data type of variable y:",type(y))
print("Addition:",x+y)
print("Subtraction:",x-y)
print("Multiplication:",x*2)
print("Exponentiation:",x**2)
```

```
Demo of basic data types:Numbers
x= 3
y= 2.5
Data type of variable x: <class 'int'>
Data type of variable y: <class 'float'>
Addition: 5.5
Subtraction: 0.5
Multiplication: 6
Exponentiation: 9
```

[4]:
```
print("Demo of basic data types:Boolean")
t=True
f=False
print("t=",t)
print("f=",f)
print("Data type of variable t:",type(t))
print("Data type of variable f:",type(f))
print("Logical AND operation:",t and f)
print("Logical OR operation:",t or f)
print("Logical NOT operation:",not t)
print("Logical XOR operation:",t!=f)
```

```
Demo of basic data types:Boolean
t= True
f= False
Data type of variable t: <class 'bool'>
```

```
Data type of variable f: <class 'bool'>
Logical AND operation: False
Logical OR operation: True
Logical NOT operation: False
Logical XOR operation: True
```

[3]:
```
print("Demo of basic data types:String")
s="hello"
t="world"
print("String 1:",s)
print("String 2:",t)
d=s+""+t
print("String concatenation:",d)
print("Capitalize:",d.capitalize())
print("Converted to uppercase:",s.upper())
print("Right justify a string:",s.rjust(7))
print("String at center:",s.center(7))
print("After replacing l with ell=",s.replace('l','(ell)'))
print("String after striping leading and trailing white spaces:",'world'.
 ↪strip())
```

```
Demo of basic data types:String
String 1: hello
String 2: world
String concatenation: helloworld
Capitalize: Helloworld
Converted to uppercase: HELLO
Right justify a string:   hello
String at center:  hello
After replacing l with ell= he(ell)(ell)o
String after striping leading and trailing white spaces: world
```

[5]:
```
print("Containers:Lists")
nums=list(range(5))
print("list 'nums' contains:",nums)
nums[4]="abc"
print("List can contain elements of different types.Example:",nums)
nums.append("xyz")
print("nums after inserting new element at the end:")
print("Sublists:")
print("A slice from index 2 to 4:",nums[2:4])
print("A slice from index 2 to the end:",nums[2:])
print("A slice from the start to index 2 :",nums[:2])
print("A slice of the whole list:",nums[:])
nums[4:]=[8,9]
print("after assigning a new sublist to 'nums':")
for idx,i in enumerate(nums):
```

```
    print('%d:%s' %(idx+1,i))
even_squares=[x**2 for x in nums if x%2==0]
print("List of squares of even numbers from 'nums':",even_squares)
```

```
Containers:Lists
list 'nums' contains: [0, 1, 2, 3, 4]
List can contain elements of different types.Example: [0, 1, 2, 3, 'abc']
nums after inserting new element at the end:
Sublists:
A slice from index 2 to 4: [2, 3]
A slice from index 2 to the end: [2, 3, 'abc', 'xyz']
A slice from the start to index 2 : [0, 1]
A slice of the whole list: [0, 1, 2, 3, 'abc', 'xyz']
after assigning a new sublist to 'nums':
1:0
2:1
3:2
4:3
5:8
6:9
List of squares of even numbers from 'nums': [0, 4, 64]
```

[7]:
```
print("Containers:Dictionaries")
d=dict()
d={'cat':'cute','dog':'furry'}
print("Dictionary:",d)
print("Is the dictionary has the key 'cat'?",'cat' in d)
d['fish']='wet'
print("After adding new entry to 'd':",d)
print("Get an element monkey:",d.get('monkey','N/A'))
print("Get an element fish:",d.get('fish','N/A'))
del d['fish']
print("After deleting the newly added entry from 'd':",d)
print("Demo of dictionary comprehension:")
squares={x:x*x for x in range(10)}
print("Squares of integers of range 10:")
for k,v in squares.items():
    print(k,":",v)
```

```
Containers:Dictionaries
Dictionary: {'cat': 'cute', 'dog': 'furry'}
Is the dictionary has the key 'cat'? True
After adding new entry to 'd': {'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
Get an element monkey: N/A
Get an element fish: wet
After deleting the newly added entry from 'd': {'cat': 'cute', 'dog': 'furry'}
Demo of dictionary comprehension:
Squares of integers of range 10:
```

```
0 : 0
1 : 1
2 : 4
3 : 9
4 : 16
5 : 25
6 : 36
7 : 49
8 : 64
9 : 81
```

[10]:
```
print("Containers:Sets")
num1={100,110,120}
print("Set 'num1':",num1)
num1.add(90)
print("'num1' after inserting 90:",num1)
num1.update([50,60,70])
print("'num1' after inserting multiple elements:",num1)
num1.remove(60)
print("'num1' after removing 60:",num1)
print("Set comprehension and set operations;")
n1={x for x in range(10)}
print("n1=",n1)
n2={x for x in range(10) if x%2!=0}
print("n2=",n2)
print("n1 union n2:",n1|n2)
print("n1 intersection n2:",n1&n2)
print("n1 difference n2:",n1-n2)
```

```
Containers:Sets
Set 'num1': {120, 100, 110}
'num1' after inserting 90: {120, 90, 100, 110}
'num1' after inserting multiple elements: {100, 70, 110, 50, 120, 90, 60}
'num1' after removing 60: {100, 70, 110, 50, 120, 90}
Set comprehension and set operations;
n1= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
n2= {1, 3, 5, 7, 9}
n1 union n2: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
n1 intersection n2: {1, 3, 5, 7, 9}
n1 difference n2: {0, 2, 4, 6, 8}
```

[11]:
```
print("Containers:Tuples")
d={(x,x+1):x for x in range(10)}
print("Dictionary with tuple keys:")
for k,v in d.items():
    print(k,":",v)
t=(5,6)
```

```
print("Tuple t:",t)
print(d[t])
print(d[1,2])
```

```
Containers:Tuples
Dictionary with tuple keys:
(0, 1) : 0
(1, 2) : 1
(2, 3) : 2
(3, 4) : 3
(4, 5) : 4
(5, 6) : 5
(6, 7) : 6
(7, 8) : 7
(8, 9) : 8
(9, 10) : 9
Tuple t: (5, 6)
5
1
```

[12]:
```
print("Demo of function : program to find factorial of a number")
def fact(n):
    if n == 1:
        return 1
    else:
        return(n*fact(n-1))
n=int(input("Enter a number:"))
print("Factorial:",fact(n))
```

```
Demo of function : program to find factorial of a number

Enter a number: 5

Factorial: 120
```

[17]:
```
class Greeter:
    def __init__(self, name):
        self.name = name

    def greet(self, loud=False):
        if loud:
            print('HELLO, %s!' % self.name.upper())
        else:
            print('Hello, %s' % self.name)


g = Greeter('Fred')
```

```
g.greet()
g.greet(loud=True)
```

```
Hello, Fred
HELLO, FRED!
```

[1]:
```
import numpy as np
a=np.array([1,2,3])
b=np.array([[1,2,3],[4,5,6]])

print("one dimensional a:\n ",a)2 2]
[2. 0.]
print("two dimensional b:\n",b)
print("size of array",b.shape)
print("element at indices 0,2,1:\n",a[0],a[2],a[1])
print("array before changng element at index 0",a)
a[0]=5
print("array after changng element at index 0",a)
a=np.zeros((2,3))
print("array of all zeros\n",a)
b=np.ones((2,2))
print("array of all ones\n",b)
c=np.full((2,4),7)
print("constant array\n",c)
d=np.eye(3)
print("array of 3*3 identity matrix\n",d)
print("array with random values\n",np.random.random((2,2)))
```

```
one dimensional a:
  [1 2 3]
two dimensional b:
 [[1 2 3]
 [4 5 6]]
size of array (2, 3)
element at indices 0,2,1:
 1 3 2
array before changng element at index 0 [1 2 3]
array after changng element at index 0 [5 2 3]
array of all zeros
 [[0. 0. 0.]
 [0. 0. 0.]]
array of all ones
 [[1. 1.]
 [1. 1.]]
constant array
 [[7 7 7 7]
 [7 7 7 7]]
```

```
array of 3*3 identity matrix
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
array with random values
 [[0.14713614 0.95424495]
 [0.46442828 0.13024638]]
```

[10]:
```
import numpy as np
print("Array indexing:slicing")
a1=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print("a1=",a1)
b=a1[:2,1:3]
print("Subarray consisting of first two rows and columns 1 and 2:",b)
b=a1[1:2,:]
print("Subarray consists of second row:",b)
print("Accessing columns:")
b=a1[:,1]
print(b,b.shape)
c=a1[:,1:2]
print(c,c.shape)
print("Array integer indexing:")
a2=np.array([[1,2],[3,4],[5,6]])
print("a2=",a2)
print("Example of array integer indexing:",a2[[0,1,2],[0,1,0]])
print(a2[[0,0],[1,1]])
print(np.array([a2[0,1],a[0,1]]))
a3=a=np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
print("a3=",a3)
b=np.array([0,2,0,1])
print("b=",b)
print("a3=",a3)
print("Boolean array indexing:")
a=np.array([[1,2],[3,4],[5,6]])
print("a=",a)
bool_idx=(a>2)
print("Elements greater than 2:",a[bool_idx])
```

```
Array indexing:slicing
a1= [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
Subarray consisting of first two rows and columns 1 and 2: [[2 3]
 [6 7]]
Subarray consists of second row: [[5 6 7 8]]
Accessing columns:
[ 2  6 10] (3,)
[[ 2]
```

```
 [ 6]
 [10]] (3, 1)
Array integer indexing:
a2= [[1 2]
 [3 4]
 [5 6]]
Example of array integer indexing: [1 4 5]
[2 2]
[2 2]
a3= [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
b= [0 2 0 1]
a3= [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
Boolean array indexing:
a= [[1 2]
 [3 4]
 [5 6]]
Elements greater than 2: [3 4 5 6]
```

[ ]:

[12]:
```
import numpy as n
x=np.array([[1,2],[3,4]],dtype=np.float64)
y=np.array([[6,9],[4,4]],dtype=np.float64)
print("x=",x)
print("y=",y)
print("Element wise addition:",np.add(x,x))
print("Element wise subraction:",np.subtract(x,y))
print("Element wise multiplication:",np.multiply(x,y))
print("Element wise square root of x:",np.sqrt(x))
print("Matrix multiplication:",np.dot(x,y))
print("Sum of all elements of matrix x:",np.sum(x))
print("Sum of all elements in each column of matrix y:",np.sum(y,axis=0))
print("Sum of all elements in each row matrix y:",np.sum(y,axis=1))
print("Transpose of matrix x:",x.T)
```

```
x= [[1. 2.]
 [3. 4.]]
y= [[6. 9.]
 [4. 4.]]
Element wise addition: [[2. 4.]
 [6. 8.]]
Element wise subraction: [[-5. -7.]
```

```
 [-1.  0.]]
Element wise multiplication: [[ 6. 18.]
 [12. 16.]]
Element wise square root of x: [[1.         1.41421356]
 [1.73205081 2.        ]]
Matrix multiplication: [[14. 17.]
 [34. 43.]]
Sum of all elements of matrix x: 10.0
Sum of all elements in each column of matrix y: [10. 13.]
Sum of all elements in each row matrix y: [15.  8.]
Transpose of matrix x: [[1. 3.]
 [2. 4.]]
```

[ ]:
```
import numpy as np
print("Example for broadcasting:")
v=np.array([1,2,3])
w=np.array([4,5]])
print("v=",v)
print("w=",w)
print("outer product of above vectors:")
```

[13]:
```
import numpy as np

a1=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print("a1=",a1)

b=a1[:2,1:3]
print(b)

b=a1[1:2,:]
print(b)

b=a1[:,1]
print(b,b.shape)

c=a1[:,1:2]
print(c,c.shape)


a2=np.array([[1,2],[3,4],[5,6]])
print(a2)

print(a2[[0,1,2],[0,1,0]])

print(a2[[0,0],[1,1]])

print(np.array([a2[0,1],a[0,1]]))
```

```
a3=a=np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
print("a3=",a3)

b=np.array([0,2,0,1])
print("b=",b)

print("a3=",a3)

a=np.array([[1,2],[3,4],[5,6]])
print("a=",a)

bool_idx=(a>2)
print("Elements greater than 2:",a[bool_idx])
```

```
a1= [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
[[5 6 7 8]]
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
[[1 2]
 [3 4]
 [5 6]]
[1 4 5]
[2 2]
[2 2]
a3= [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
b= [0 2 0 1]
a3= [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
a= [[1 2]
 [3 4]
 [5 6]]
Elements greater than 2: [3 4 5 6]
```

[15]:
```
import numpy as np
print("Example for broadcasting:")
```

```
v=np.array([1,2,3])
w=np.array([4,5])
print("v=",v)
print("w=",w)
print("outer product of above vectors:")
```

```
Example for broadcasting:
v= [1 2 3]
w= [4 5]
outer product of above vectors:
```

[ ]:

[ ]: