Numpy ufuncs

ufuncs stands for 'Universal functions' and th are Numpy functions that operates on the ndarray object.

Why use ufuncs?

ufuncs are used to implement vectorizati in Numpy which is way faster than iterati over elements.

where — booleam array/ condition
dtype — type of element
out — value should be copied.

What is Vectorization?

Converting iterative statements into a vect based operation is called vectorization.
It is faster as modorn CPUs are optimized for such operations.

Add the Elements of Two Lists

list 1 : [1,2,3,4]

list 2 : [4,5,6,7]

Without ufunc, we can use Python's built-in zip() method:

```
x = [1,2,3,4]
y = [4,5,6,7]
z = []
for i, j in zip(x,y):
    z.append(i+j)
print (z)
```

O/P

```
[5,7,9,11]
```

NumPy has a ufunc for this, called add(x,y) that will produce the same result.

~~e.g~~ With ufunc, we can use the add() function.

```
import numpy as np
x = [1, 2, 3, 4]
y = [4, 5 , 6,7)
z = np.add(x,y)
print (z)
```

O/P

```
[5 7 9 11]
```

Create Your Own ufunc

Add it to your NumPy ufunc library with the frompyfunc() method.

The frompyfunc() method takes the following arguments:

1. function – the name of function.
2. inputs – The no. of input argument (array)
3. outputs – output arrays.

⊙ Create your own ufunc for addition

```
import numpy as np
def myadd(x,y):
        return x+y
myadd = np.frompyfunc(myadd,2,1)
print(myadd([1,2,3,4],[5,6,7,8]))
```

O/P

```
[6 8 10 12]
```

⊙ Check if a Function is a ufunc

A ufunc should return <class 'numpy.u

```
import numpy as np
print(type(np.add))
```

O/P

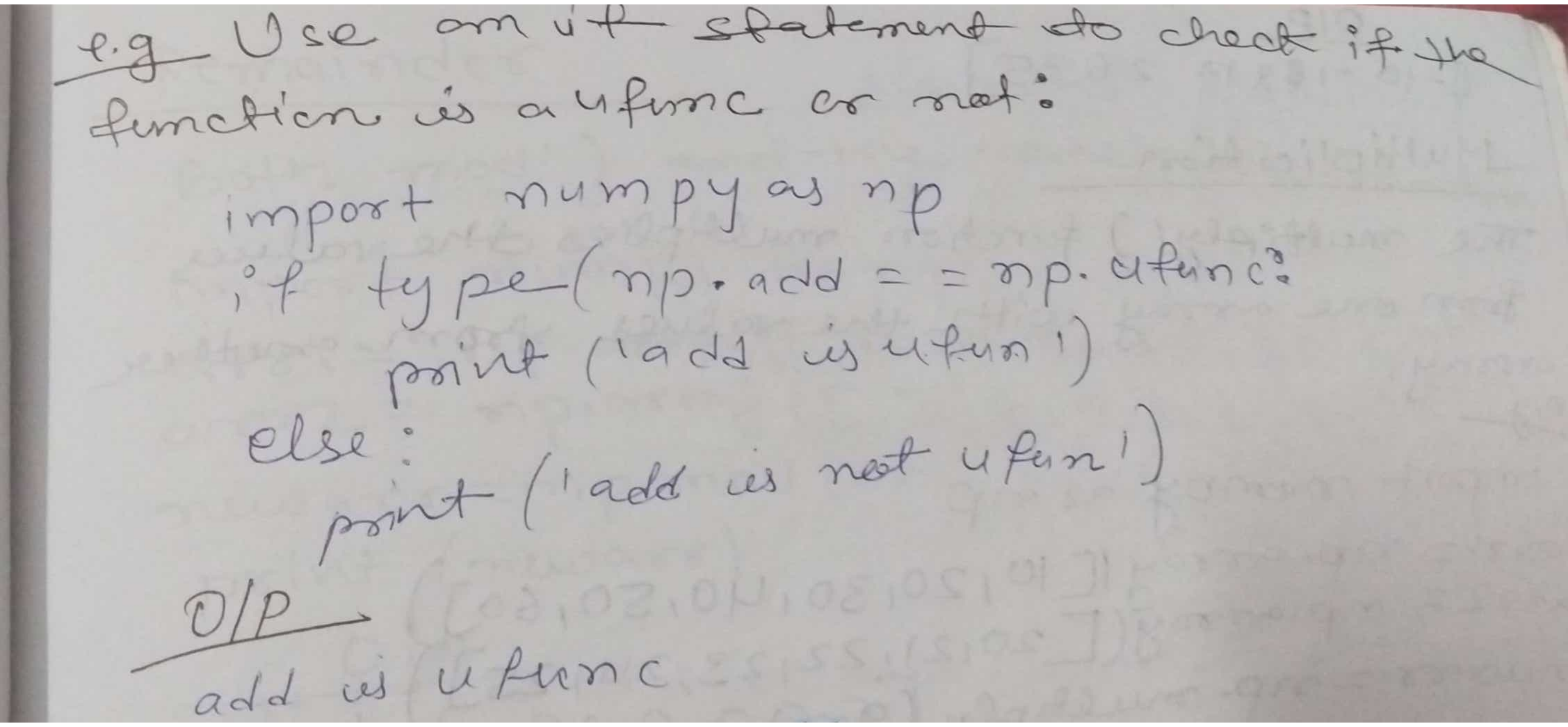
```
<class 'numpy.ufunc'>
```

e.g Check the type of another function: concat

```
import numpy as np
print(type(np.concatenate)
```

O/P

```
<class 'builtin_function_or_method'>
```



e.g Use an if statement to check if the function is a ufunc or not:

```
import numpy as np
if type(np.add = = np.ufunc:
    print ('add is ufun')
else:
    print ('add is not ufun')
```

O/P

```
add is ufunc
```