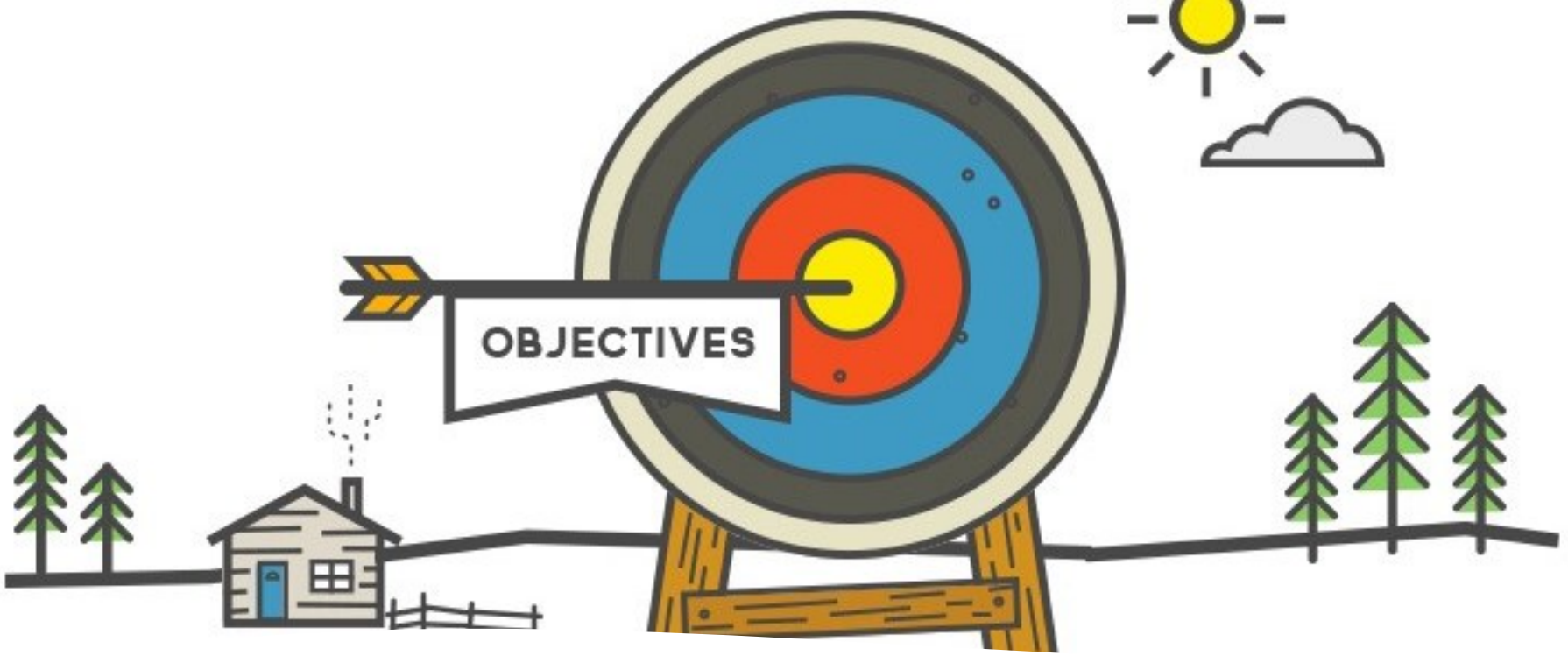
NumPy

Prof. Carlos J. Costa, PhD

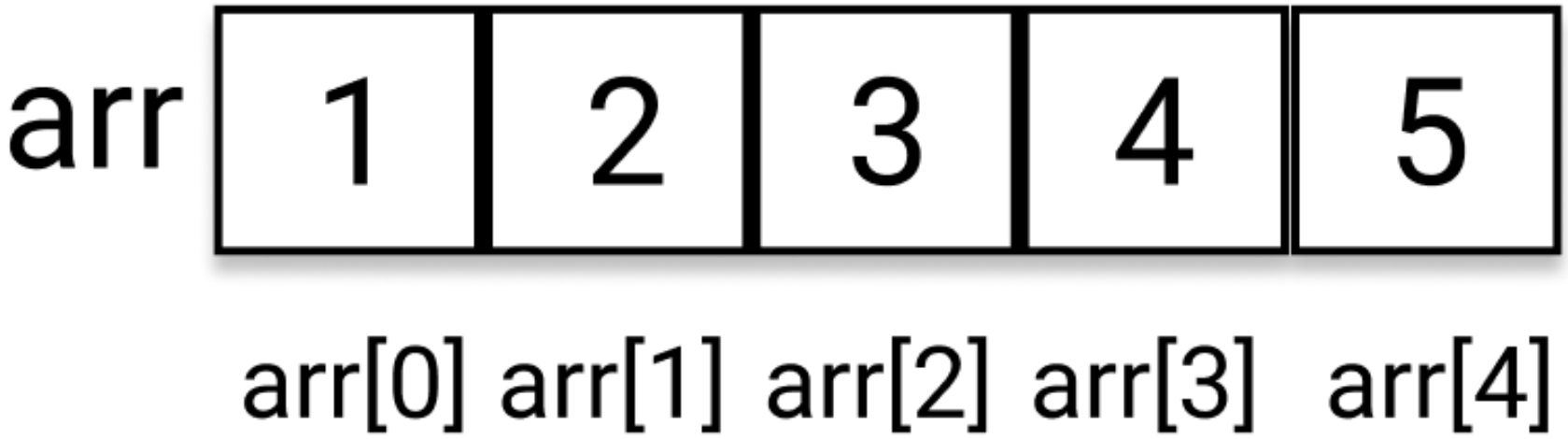


Learning Objectives

- Understand what is NumPy
- Know the main features of Numpy
- Solve mathematical problems using NumPy

Numpy

- Numerical Python
- Is an open-source Python library
- Is a fundamental Python library for scientific computing.
- Provides array related functionality
- Has higher level of performance



Numpy

```
import numpy as np
c = np.array([1,2,3,4])
print(type(c))
```

<class 'numpy.ndarray'>

Shape, Rank and Size

- The bidimensional array (matrix):

```
b = np.array([[1,2,3],[4,5,6]])
```

- What information can be obtained about this array:

```
shape = b.shape 2,3
rank = np.ndim(b) 2
size = b.size 6
```

Reshaping arrays

- Reshaping means changing the shape of an array.
- The shape of an array is the number of elements in each dimension.
- By reshaping we can add or remove dimensions or change number of elements in each dimension

```
import numpy as np
arr = np.array([4, 4, 3, 4, 5, 6, 7, 9, 0, 10, 17, 12])
newarr = arr.reshape(3, 4)
print(newarr)
```

```
[[ 4 4 3 4]
 [ 5 6 7 9]
 [ 0 10 17 12]]
```

Access an Array Element

- Change value to array:

```
a[2]=50
print(a)
```

Array with Zeros Only

- Create an array with zeros only

```
a = np.zeros((2,2))
print(a)
```

Array with Only “ones”

- Create array with only “one”

```
b = np.ones((1,2))
print(b)
```

Identity Array

- create 3x3 identity array

```
d = np.eye(3)
print(d)
```

$$I_1 = [1],$$

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\cdots\cdots\cdots\cdots\cdots\cdots,$$

$$I_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ 0 & 0 & 0 & 0 & 0 & \end{bmatrix}$$

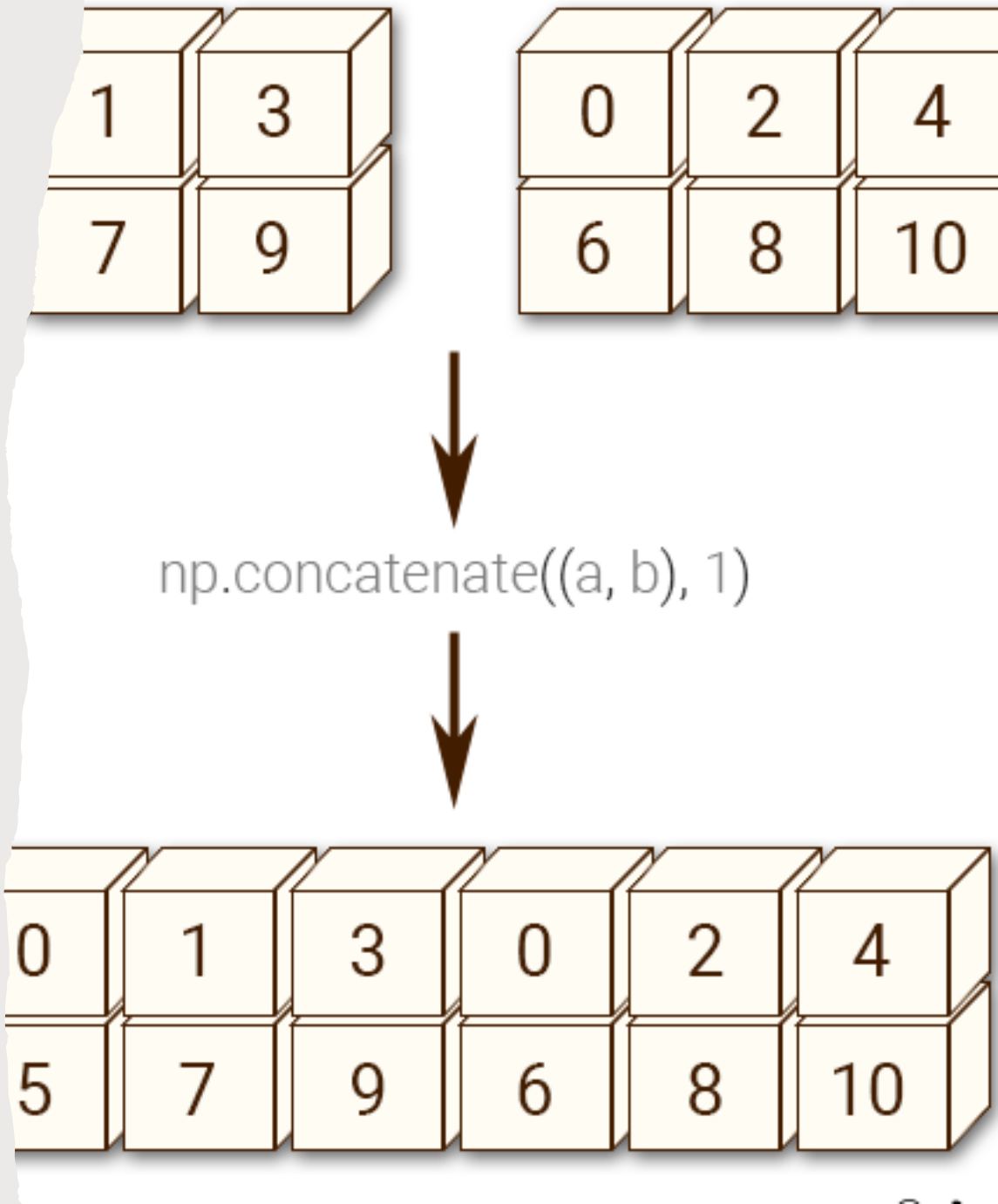
Joining NumPy Arrays

- putting contents of two or more arrays in a single array.
- In SQL we join tables based on a key, whereas in NumPy we join arrays by axes.

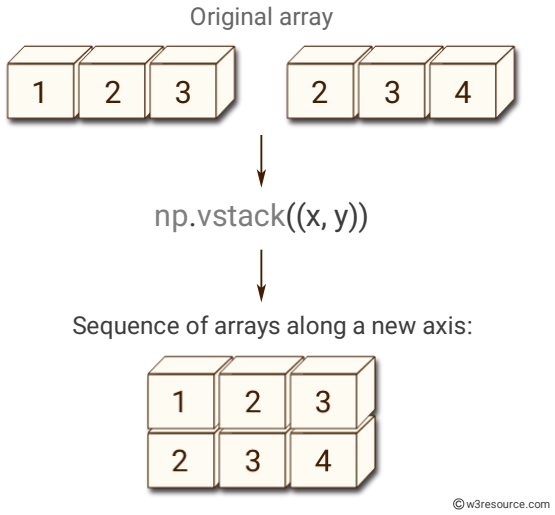
Concatenation

- We pass a sequence of arrays that we want to join to the concatenate() function, along with the axis.
- If axis is not explicitly passed, it is taken as 0.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
```

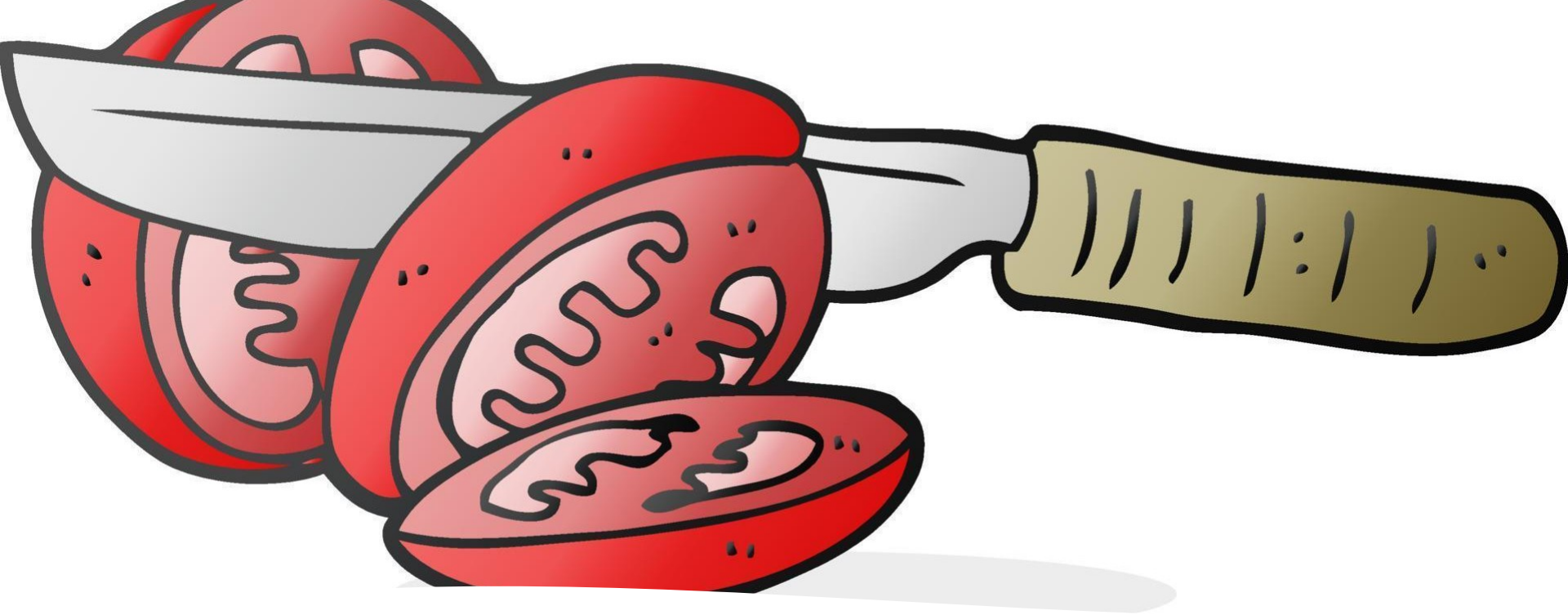


Stacking



- Stacking is same as concatenation
- The only difference is that stacking is done along a new axis.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.stack((arr1, arr2), axis=1)
print(arr)
```



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

Slicing Arrays

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

Slicing Arrays

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

print(a)



b = a[:3, 1:3]

print(b)
```

Slicing Arrays

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

b = a[:3, 1:3]

print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

Slicing Arrays

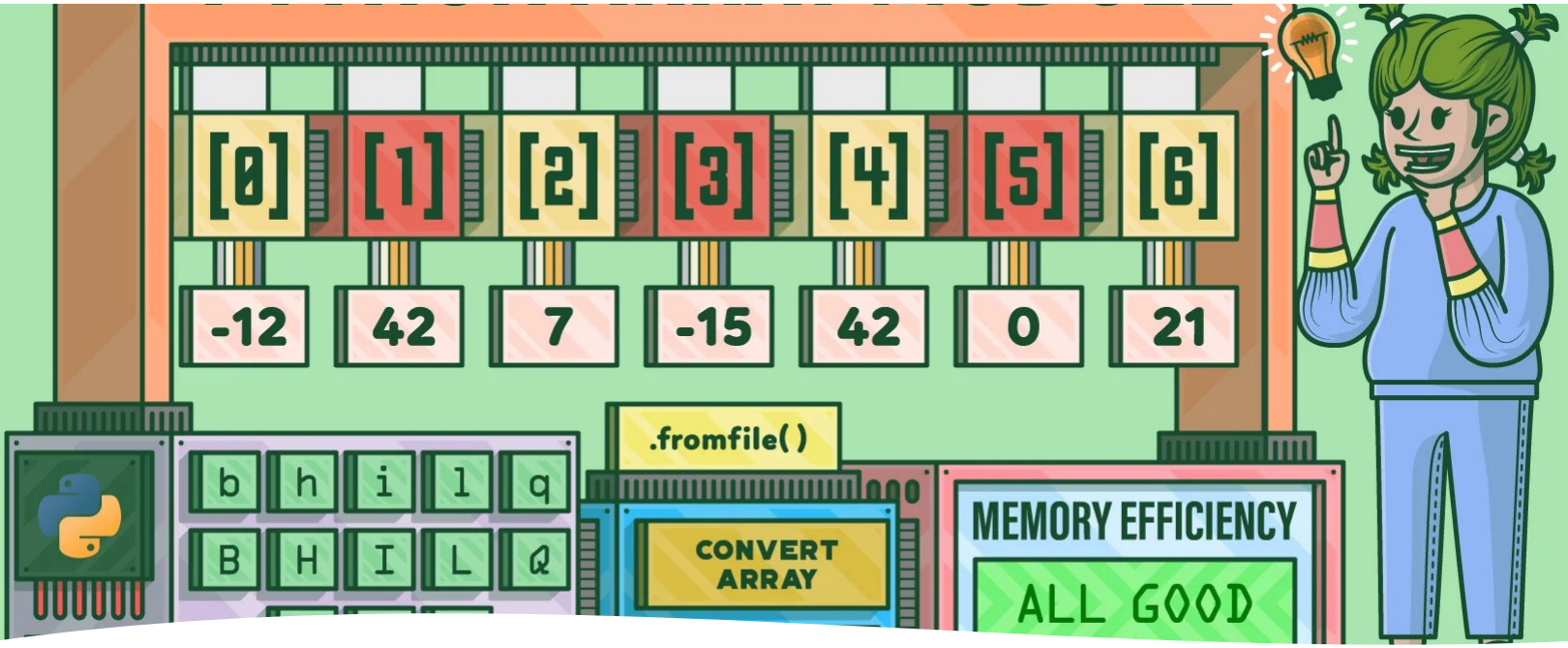
```
b[0, 0] = 99
```

- what happens to array a?

Slicing Arrays

```
b[0, 0] = 99
```

- what happens to array a?
- `b = a[:3, 1:3]`
- bCopyA= `a[:3, 1:3].copy()`



Handling Indexes

```
import numpy as np
#create a new array
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print(a)
```

Handling Indexes

- create an array of indexes

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

print(a[np.arange(4), b])
```

Handling Indexes

```
a[np.arange(4), b] += 10

print(a)
```

Handling Indexes

b=1

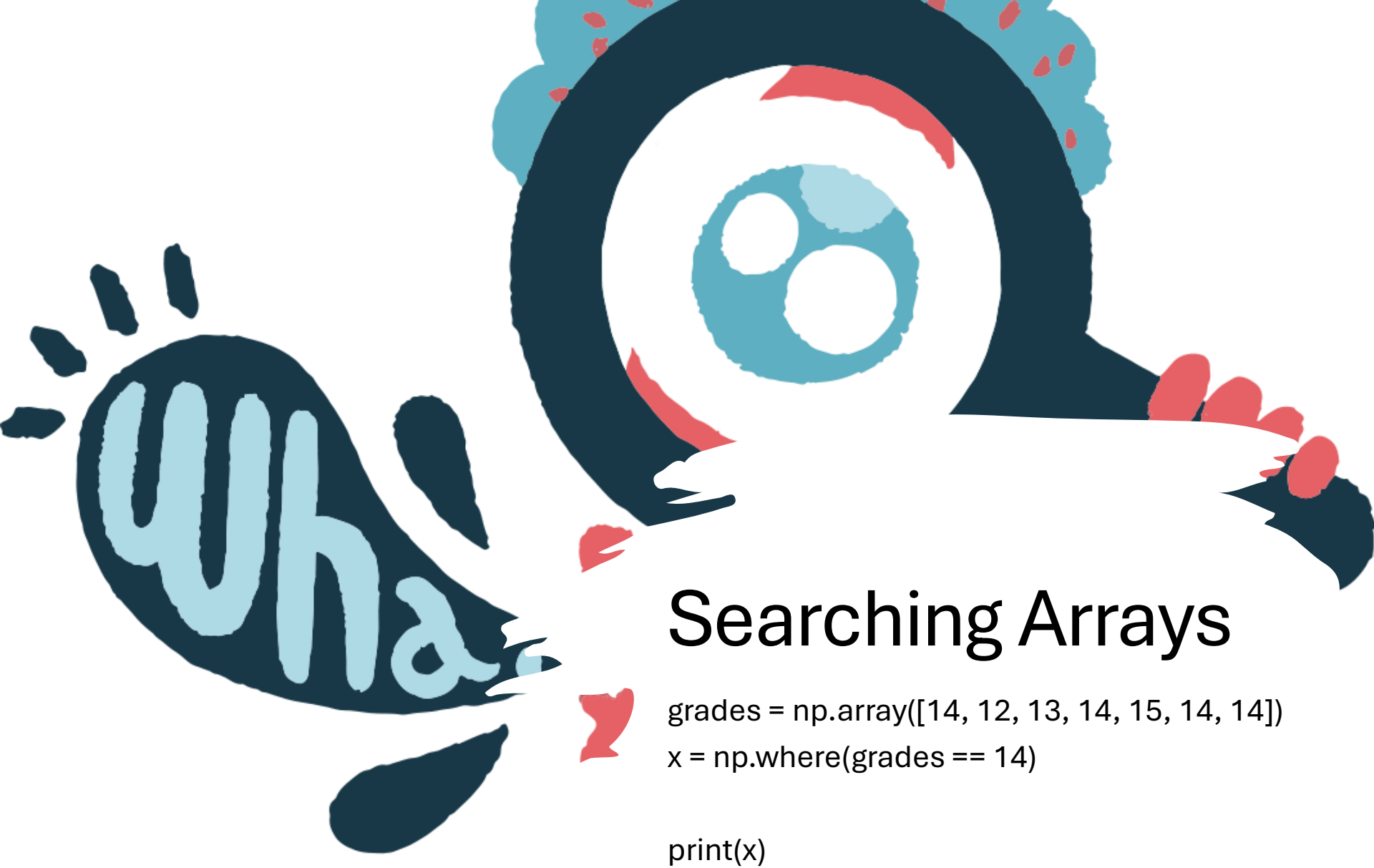
b+=10

B

b=1

b=b+10

b

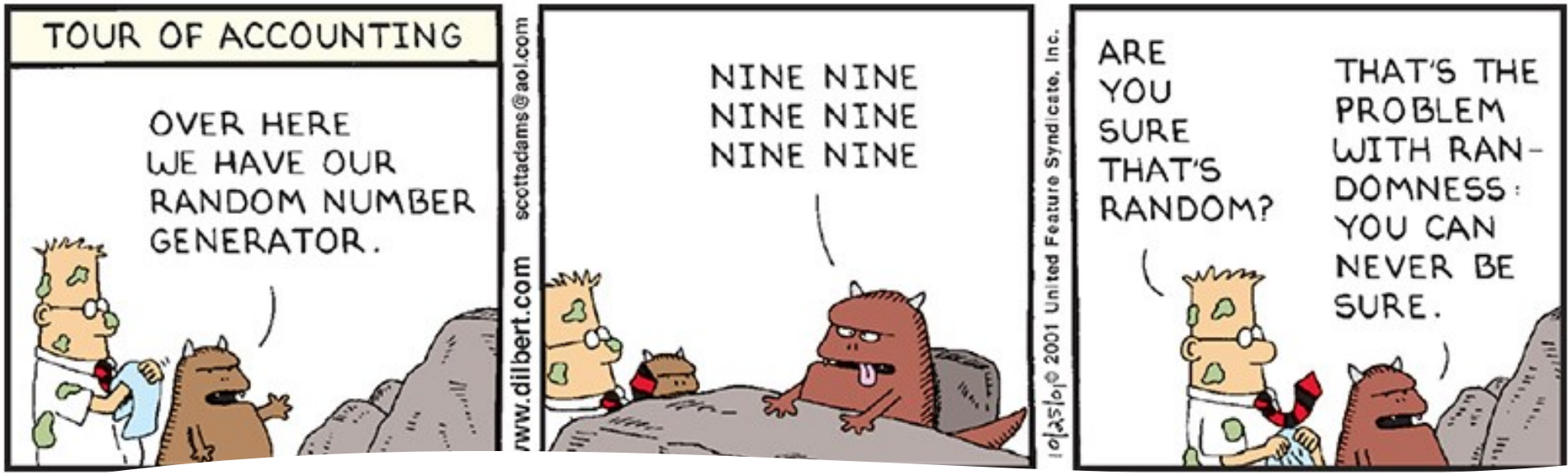


Searching Arrays

```
grades = np.array([14, 12, 13, 14, 15, 14, 14])
x = np.where(grades == 14)

print(x)
```

Searching Arrays

```
grades = np.array([14, 12, 13, 14, 15, 14, 14])
x = np.where(grades == 14)

print(x)

(array([0, 3, 5, 6], dtype=int64),)
```



Random

Random number does NOT mean a different number every time.

Random means something that can not be predicted logically.

Array filled with random numbers

- Create array filled with random numbers

```
e = np.random.random((4,4))
print(e)
```

Generate Random Number

NumPy offers the random module to work with random numbers.

```
from numpy import random
x = random.randint(100)
print(x)
```

```
from numpy import random
x = random.rand()
print(x)
```

Generate Random Array

Array of Integers:

```
from numpy import random
x=random.randint(100, size=(5))
print(x)
```

from numpy import random

```
x = random.randint(100, size=(3, 5))
print(x)
```

Array of Floats:

```
from numpy import random
x = random.rand(5)
print(x)
```

Random Data Distribution

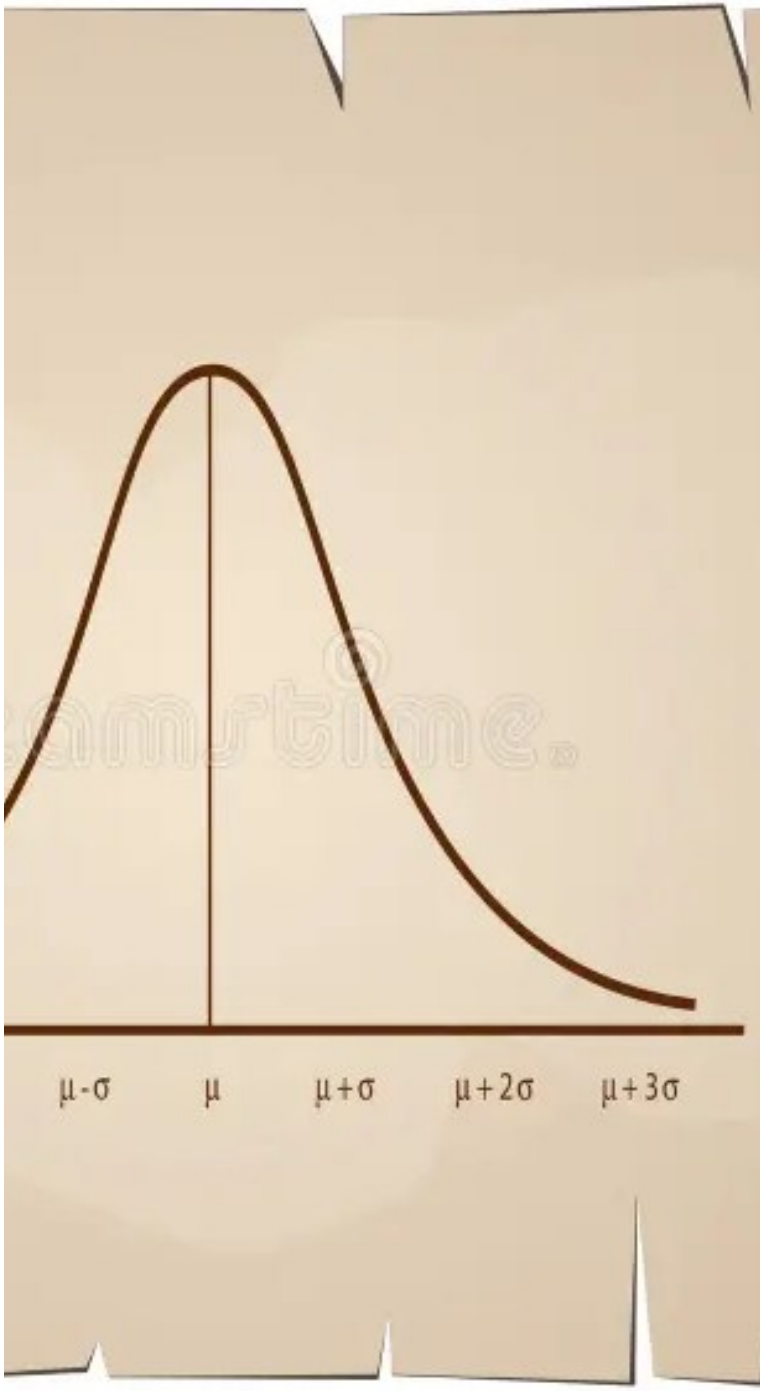
A random distribution is a set of random numbers that follow a certain *probability density function*.

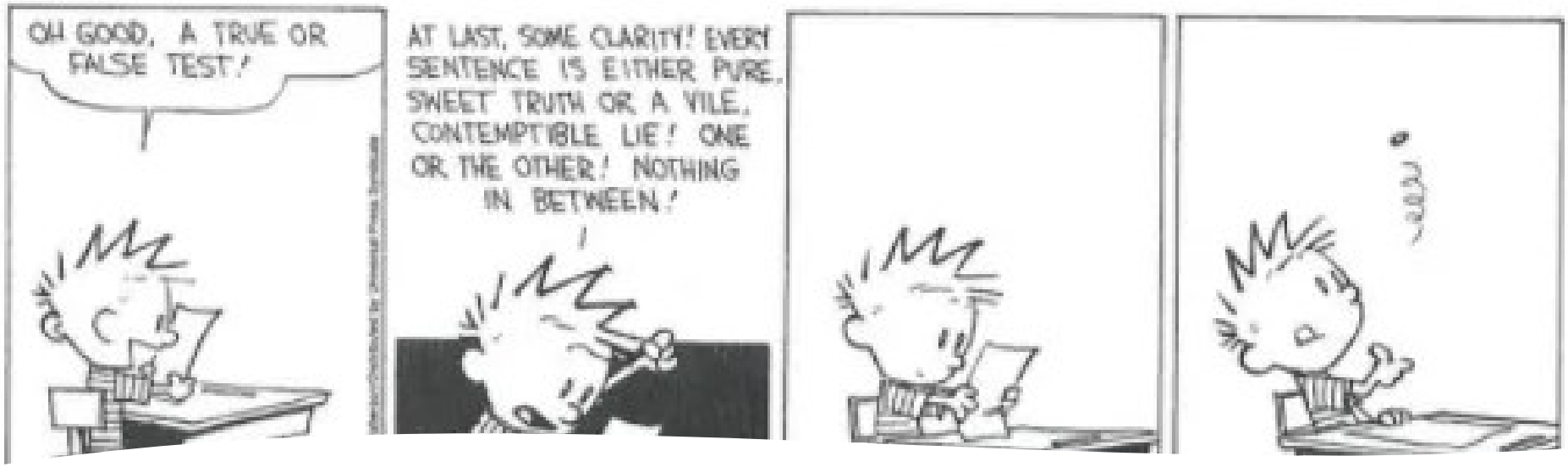
Normal Distribution

random.normal has three parameters:

- loc - (Mean) where the peak of the bell exists.
- scale - (Standard Deviation) how flat the graph distribution should be.
- size - The shape of the returned array.

```
from numpy import random
x = random.normal(loc=1, scale=2, size=(2, 3))
print(x)
```





Binomial Distribution

Binomial Distribution is a Discrete Distribution.

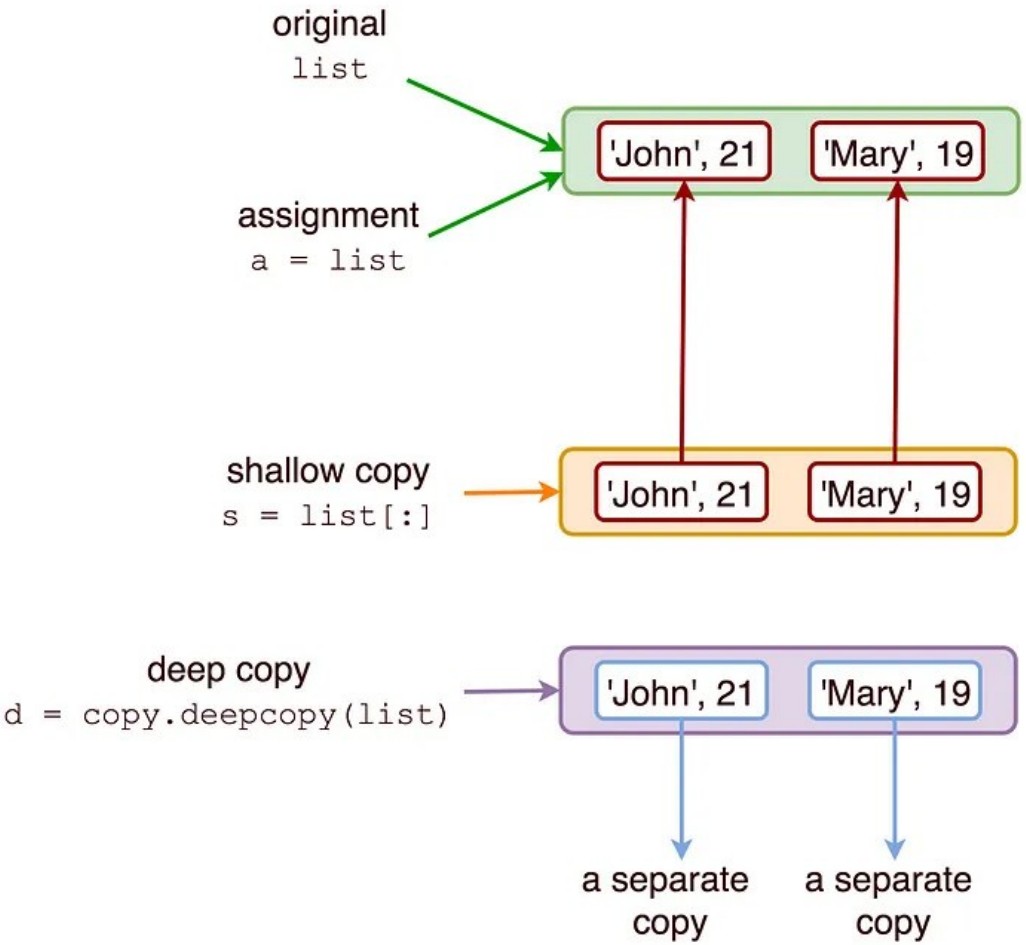
- It describes the outcome of binary scenarios, e.g. toss of a coin, it will either be head or tails.

It has three parameters:

- n - number of trials.
- p - probability of occurrence of each trial (e.g. for toss of a coin 0.5 each).
- size - The shape of the returned array.

```
from numpy import random
x = random.binomial(n=10, p=0.5, size=10)
print(x)
```

Copy in Python



Conclusions

- NumPy (Numerical Python) is an open source Python library
- Array related functionality, including searches, index handling

References

- http://www.numpy.org/