Algorithm and Data Structures
Week 4

Endang Wahyu Pamungkas, P.hD.

Arrays

- Array is a data structure that contains a set of items or elements. Usually each item in the array has the same data type, for example integer or string or other object. Each item also can be accessed using an index.
- Arrays are used in computers to organize data so that each item in an array is easy to be found or sorted.
- The definition of an array is not rigid, it is loose. So the implementation can vary depending on the needs.

String as Array

- A string can be thought of as a collection of characters. Character of strings can be accessed by using its index. However, string as an array is immutable.

- Here is an example of creating and accessing a string item.

```
data = “saya mahasiswa UMS”
x = data[2]
print(data[5])
```

Tuple as Array

- A tuple is a collection of any item. If you need an array whose items are numbers, strings, or any other object, then tuples can be used. Tuples allow indexing. The only constraint of tuples is that the items in the tuple are immutable.

```
data = (("Z1001", "Badu", 20), ("Z1002",
"Joko", 21), ("Z1003", "Maya", 19))
mhs1 = data[1]
print(data[2])
print(data[0][1], "age", data[0][2])
```

List as Array

- List is a collection of arbitrary items and is mutable. Lists allow indexing and array sizes can be made dynamic. List even has methods for searching and sorting data items.

```
data = [("Z1001", "Badu", 20), ("Z1002",
"Joko", 21), ("Z1003", "Maya", 19)]
mhs1 = data[1]
data[0] = ("Z1001", "Budi", 18)
print(data[2])
print(data[0][1], "age", data[0][2])
data.index(("Z1001", "Budi", 18))
```

Array Modul

- Python has a module with array names for creating array objects. The array object with this module has the property of only being able to store numbers. Items are accessed by indexing, items can be modified and array size can be changed dynamically.

```
import array as arr

a = arr.array('d', [1.1, 3.5, 4.5])
b = a[2]
a[0] = 2.2
print(a)
```

Class and Array

- Arrays can be implemented with classes. Classes created can be tailored to the needs (customized). Since arrays are generally used to store mutable sets of data, their internal properties can be lists. The important thing is that this list can only be accessed internally within the class/object.

```
class Contoh():
    _internal = [None, None, None, None,
None]
```

Class and Array

- Set and get item are the most important method for the array class.

```
def __setitem__(self, key, value):
        self._internal[key] = value

def __getitem__(self, key):
        return self._internal[key]
```

Array with Limit Size

```
 1.  class Array2(object):
 2.      def __init__(self, n):
 3.          self._internal = n * [None]
 4.
 5.      def __getitem__(self, key):
 6.          if 0 <= key < len(self._internal):
 7.              return self._internal[key]
 8.          else:
 9.              raise KeyError("Limit Exceeded")
10.
11.      def __setitem__(self, key, value):
12.          if 0 <= key < len(self._internal):
13.              self._internal[key] = value
14.          else:
15.              raise KeyError("Limit Exceeded")
```

Array with Dynamic Size

```
1.  class Array3(Array2):
2.      def __setitem__(self, key, value):
3.          if key >= len(self._internal):
4.              kurang = key - len(self._internal) + 1
5.              li = kurang * [None]
6.              self._internal.extend(li)
7.          self._internal[key] = value
```

Case Study

The following is a class with the type of each item being an object of the Student class.

```
1.  class Mahasiswa():
2.      def __init__(self, nim, nama, umur):
3.          self.nim = nim
4.          self.nama = nama
5.          self.umur = umur
```

Case Study

```
1.  class Array4(object):
2.      def __init__(self, n):
3.          self._internal = n * [None]
4.
5.      def __getitem__(self, key):
6.          return self._internal[key]
7.
8.      def __setitem__(self, key, value):
9.          if isinstance(value, Mahasiswa):
10.             self._internal[key] = value
11.         else:
12.             raise ValueError("Object not
    Mahasiswa")
```

Numpy

- Arrays can have any number of dimensions, including zero (a scalar).
- Arrays are typed: np.uint8, np.int64, np.float32, np.float64
- Arrays are dense. Each element of the array exists and has the same type.

```
import numpy as np

a = np.array([[1,2,3],[4,5,6]],dtype=np.float32)

print a.ndim, a.shape, a.dtype
```

Array Creation

- **np.ones, np.zeros**
- np.arange
- np.concatenate
- np.astype
- np.random.random

```
>>> np.ones((3,5),dtype=np.float32)
array([[ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.]], dtype=float32)
```

```
>>> np.zeros((6,2),dtype=np.int8)
array([[0, 0],
       [0, 0],
       [0, 0],
       [0, 0],
       [0, 0],
       [0, 0]], dtype=int8)
```

Array Creation

- np.ones, np.zeros
- **np.arange**
- np.concatenate
- np.astype
- np.random.random

```
>>> np.arange(1334,1338)
array([1334, 1335, 1336, 1337])
```

Array Creation

- np.ones, np.zeros
- np.arange
- **np.concatenate**
- np.astype
- np.random.random

```
>>> A = np.ones((2,3))
>>> B = np.zeros((4,3))
>>> np.concatenate([A,B])
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>>
```

Array Creation

- np.ones, np.zeros
- np.arange
- np.concatenate
- **np.astype**
- np.random.random

```
>>> A
array([[ 4670.5,  4670.5,  4670.5],
       [ 4670.5,  4670.5,  4670.5],
       [ 4670.5,  4670.5,  4670.5],
       [ 4670.5,  4670.5,  4670.5],
       [ 4670.5,  4670.5,  4670.5]], dtype=float32)
>>> print(A.astype(np.uint16))
[[4670 4670 4670]
 [4670 4670 4670]
 [4670 4670 4670]
 [4670 4670 4670]
 [4670 4670 4670]]
```

Array Creation

- np.ones, np.zeros
- np.arange
- np.concatenate
- np.astype
- **np.random.random**

```
>>> np.random.random((10,3))
array([[ 0.61481644,  0.55453657,  0.04320502],
       [ 0.08973085,  0.25959573,  0.27566721],
       [ 0.84375899,  0.2949532 ,  0.29712833],
       [ 0.44564992,  0.37728361,  0.29471536],
       [ 0.71256698,  0.53193976,  0.63061914],
       [ 0.03738061,  0.96497761,  0.01481647],
       [ 0.09924332,  0.73128868,  0.22521644],
       [ 0.94249399,  0.72355378,  0.94034095],
       [ 0.35742243,  0.91085299,  0.15669063],
       [ 0.54259617,  0.85891392,  0.77224443]])
```

Shaping Array

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

a = a.reshape(3,2)

a = a.reshape(2,-1)

a = a.ravel()
```

1. Total number of elements cannot change.
2. Use -1 to infer axis shape
3. Row-major by default (MATLAB is column-major)

Mathematical Operations

Mathematical operators

- **Arithmetic operations are element-wise**
- Logical operator return a bool array
- In place operations modify the array

```
>>> a
array([1, 2, 3])
>>> b
array([ 4,  4, 10])
>>> a * b
array([ 4,  8, 30])
```

Mathematical Operations

Mathematical operators

- Arithmetic operations are element-wise
- **Logical operator return a bool array**
- In place operations modify the array

```
>>> a
array([[ 0.93445601,  0.42984044,  0.12228461],
       [ 0.06239738,  0.76019703,  0.11123116],
       [ 0.14617578,  0.90159137,  0.89746818]])
>>> a > 0.5
array([[ True, False, False],
       [False,  True, False],
       [False,  True,  True]], dtype=bool)
```

Mathematical Operations

Mathematical operators

- Arithmetic operations are element-wise
- Logical operator return a bool array
- **In place operations modify the array**

```
>>> a
array([[ 4, 15],
       [20, 75]])
>>> b
array([[ 2,  5],
       [ 5, 15]])
>>> a /= b
>>> a
array([[2, 3],
       [4, 5]])
```

Indexing

```
x[0,0]    # top-left element
x[0,-1]   # first row, last column
x[0,:]# first row (many entries)
x[:,0]# first column (many entries)
```

Notes:

- Zero-indexing
- Multi-dimensional indices are comma-separated (i.e., a tuple)

Slicing

Syntax: start:stop:step

```
a = list(range(10))

a[:3] # indices 0, 1, 2

a[-3:] # indices 7, 8, 9

a[3:8:2] # indices 3, 5, 7

a[4:1:-1] # indices 4, 3, 2 (this one is tricky)
```

Exercise

Finding the Longest Word

Write a Python program to find **the longest word** in a given String.

Example:

Input: I love to learn python

Output: the longest word is “python”

Exercise

Finding missing Items.

Write a Python program to find **one missing number** in a given array of numbers between 1 and 10.

Example:

Input: [1, 2, 3, 4, 5, 6, 8, 9, 10] (array)

Output: Missing number is 7.