COSC21 PRELIMS

Arrays

Arrays

- are used to store multiple values in one single variable.

Example

Create an array containing car names:

```
cars=[“Ford”, “Volvo”, “BMW”]
```

What is an Array?

- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
cars1 = “Ford”
cars2 = “Volvo”
cars3 = “BMW”
```

Access the Elements of an Array

- You refer to an array element by referring to the index number.

Example

Get the value of the first array item:

```
x = cars[0]
```

Modify the value of the first array item:

```
cars[0] = “Toyota”
```

Example Program

```
cars=[“Ford”, “Volvo”, “BMW”]
x = cars[0]
print(x)
```

Example Output

```
Ford
```

The Length of an Array

- Use the len() method to return the length of an array (the number of elements in an array).

Example

Return the number of elements in the cars array:

```
x = len(cars)
```

Example Program

```
cars=[“Ford”, “Volvo”, “BMW”]
x = len(cars)
print(x)
```

Example Output

```
3
```

Looping Array Elements

- You can use the for in loop to loop through all the elements of an array.

Example

Print each item in the cars array:

```
for x in cars:
  print(x)
```

Example Program

```
cars=[“Ford”, “Volvo”, “BMW”]
for x in cars:
  print(x)
```

Example Output

```
Ford
Volvo
BMW
```

Array, Method and Single Dimensional Array

Methods

append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes an element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

NOTE:

index – starts from 0

element – starts from 1

Adding Array Elements

- You can use the append() method to add an element to an array.

Example Program

```
cars=[“Ford”, “Volvo”, “BMW”]
cars.append(“Honda”)
print(cars)
```

Example Output

```
[‘Ford’, ‘Volvo’, ‘BMW’, ‘Honda’]
```

Python List clear() Method
- The clear() method removes all the elements from a list.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
fruits.clear()
print(fruits)
```

Example Output

```
[]
```

Python List copy() Method
- The copy() method returns a copy of the specified list.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
x = fruits.copy()
print(x)
```

Example Output

```
[‘apple’, ‘banana’, ‘cherry’]
```

Python List count() Method
- The count() returns the number of elements with the specified value.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
x = fruits.count(“cherry”)
print(x)
```

Example Output

```
1
```

Python List extend() Method
- The extend() method removes all the elements from a list.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
cars=[“Ford”, “BMW”, “Volvo”]
fruits.extend(cars)
print(fruits)
```

Example Output

```
[‘apple’, ‘banana’, ‘cherry’, ‘Ford’, ‘BMW’, ‘Volvo’]
```

Python List index() Method
- The index() method returns the position at the first occurrence of the specified value.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
x = fruits.index(“cherry”)
print(x)
```

Example Output

```
2
```

Python List insert() Method
- The insert() method inserts the specified value at the specified position.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
fruits.insert(1, “orange”)
print(fruits)
```

Example Output

```
[‘apple’, ‘orange’, ‘banana’, ‘cherry’]
```

Removing Array Elements
- You can use the pop() method to remove an element from the array.

Example Program

```
cars=[“Ford”, “Volvo”, “BMW”]
cars.pop(1)
print(cars)
```

Example Output

```
[‘Ford’,  ‘BMW’]
```

Python List remove() Method
- You can also use remove() method to remove an element from the array.

Example Program

```
cars=[“Ford”, “Volvo”, “BMW”]
cars.remove(“Volvo”)
print(cars)
```

Example Output

```
[‘Ford’,  ‘BMW’]
```

Python List reverse() Method
- The reverse() method reverse the sorting order of the elements.

Example Program

```
fruits=[“apple”, “banana”, “cherry”]
fruits.reverse()
print(fruits)
```

Example Output

```
[‘cherry’, ‘banana’, ‘apple’]
```

Python List sort() Method

- The sort() method sorts the list ascending by default

Example Program

```
cars=[“Ford”, “BMW”, “Volvo”]
cars.sort()
print(cars)
```

Example Output

```
[‘BMW’, ‘Ford’, ‘Volvo’]
```

Single Dimensional Arrays

import array	a = array.array (‘i’, [4, 6, 2, 9])
import array as arr	a = arr.array (‘i’, [4, 6, 2, 9])
from array import *	a = array (‘i’, [4, 6, 2, 9])

Example Program

```
import array

a = array.array(‘i’, [1, 2, 3, 4])

print(“Items are: “)
for x in a:
    print(x)
```

Example Output

```
Items are:
1
2
3
4
```

Example Program

```
import array as arr

a = arr.array(‘u’, [‘a’, ‘b’, ‘c’, ‘d’])

print(“Items are: “)
for ch in a:
    print(ch)
```

Example Output

```
Items are:
1
2
3
4
```

Example Program

```
from array import *
a = array(‘i’, [1, 2, 3, 4])
b = array(a.typecode, (i for i in a))
print(“Items are:”)
for i in b:
    print(i)

c = array(a.typecode, (i*3 for i in a))
print(“Items are:”)
for i in c:
    print(i)
```

Example Output

```
Items are:
1
2
3
4
Items are:
3
6
9
12
```

NOTE:

i – integer

u – Unicode

Arrays List

Array List

- it is a collection which is ordered and changeable. Allows duplicate members.
- A list is a collection which is ordered and changeable. In python lists are written with square brackets.

Example

Create a List:

```
thislist  = [“apple”, “banana”, “cherry”]
print(thislist)
```

Access Items

- You access the list by referring to the index number:

Example

Print the second item of the list:

```
thislist  = [“apple”, “banana”, “cherry”]
print(thislist[1])
```

Negative Indexing

- Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc.

Example

Print the last item of the list:

```
thislist  = [“apple”, “banana”, “cherry”]
print(thislist[-1])
```

Range of Indexes

- You can specify a range of indexes by specifying where to start and where to end the range. When specifying a range, the return value will be a new list with the specified items.

Note: The search will start at index 2 (included) and end at index 5 (not included).

Example

Return the third, fourth, and fifth item:

```
thislist  = [“apple”, “banana”, “cherry”, “orange”, “kiwi”, “melon”, “mango”]
print(thislist[2:5])
```

Range of Negative Indexes

- Specify negative indexes if you want to start the search from the end of the list:

Example

This example returns the items from index -4 (included) to index -1 (excluded)

```
thislist  = [“apple”, “banana”, “cherry”, “orange”, “kiwi”, “melon”, “mango”]
print(thislist[-4:-1])
```

Change Item Value

- To change the value of a specific item, refer to the index number:

Example

Change the second item:

```
thislist  = [“apple”, “banana”, “cherry”]
thislist[1] = “blackcurrant”
print(thislist)
```

Slicing and Indexing Arrays

How to access and Length the array elements?

Example Program

```
from array import *
a = array(‘i’, [1, 2, 3, 4])
n = len(a)
print(“the total length of value: ”, n)
```

Example Output

```
the total length of value: 4
```

How to Slice the array elements?

Example Program

```
from array import *
x = array(‘i’, [10, 20, 30, 40, 50, 60])

y = x[1:4]
print(y)

y = x[0: ]
print(y)

y = x[ :4]
print(y)

y = x[-4: ]
print(y)

y = x[0:7:2]
print(y)

for i in x[2:5]:
    print(i)
```

Example Output

```
array(‘i’, [20, 30, 40])
array(‘i’, [10, 20, 30, 40, 50, 60])
array(‘i’, [10, 20, 30, 40])
array(‘i’, [30, 40, 50, 60])
array(‘i’, [10, 30, 50])
30
40
50
```

Tuple in Array

Tuple

- A tuple is a collection which is ordered and unchangeable. In python, tuples are written with round brackets.

Example

Create a Tuple:

```
thistuple  = (“apple”, “banana”, “cherry”)
print(thistuple)
```

Access Tuple Items

- You can access tuple items by referring to the index number, inside square brackets:

Example

Print the second item in the tuple:

```
thistuple  = (“apple”, “banana”, “cherry”)
print(thistuple[1])
```

Negative Indexing

- Negative indexing means beginning from the end, -1 refers to the last items, -2 refers to the second last item etc.

Example

Print the last item of the tuple:

```
thistuple  = (“apple”, “banana”, “cherry”)
print(thistuple[-1])
```

Range of Indexes

- You can specify a range of indexes by specifying where to start and where to end the range.
When specifying a range, the return value will be a new tuple with the specified items.
Note: The search will start at index 2 (included) and end at index 5 (not included).

Example

Return the third, fourth, and fifth item:

```
thistuple  = (“apple”, “banana”, “cherry”, “orange”, “kiwi”, “melon”, “mango”)
print(thistuple[2:5])
```

Range of Negative Indexes

- Specify negative indexes if you want to start the search from the end of the tuple:

Example

This example returns the items from index -4 (included) to index -1 (excluded)

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

Change Tuple Values

- Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called. But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

Example

Convert the tuple into a list to be able to change it:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Loop Through a Tuple

- You can loop through the tuple by using a ***for*** loop

Example

Iterate through the items and print the values:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
    print(x)
```

Check if Item Exists

- To determine if a specified item is present in a tuple use the ***in*** keyword

Example

Check if "apple" is present in the tuple:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
    print("Yes, 'apple' is in the fruits tuple")
```

Tuple Length

- To determine how many items a tuple has, use the ***len()*** method

Example

Print the number of items in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Add Items

- Once a tuple is created, you cannot add items to it. Tuples are unchangeable.

Create Tuple with One Item

- To create a tuple with only one item, you have add a comma after the item, unless Python will not recognize the variable as a tuple.

Example

One item tuple, remember the comma:

```
thistuple = ("apple", )
print(type(thistuple))
```

Remove Items

- Tuples are unchangeable, so you cannot remove items from it, but you can delete the tuple completely:

Example

the del keyword can delete the tuple completely:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
```

Join Two Tuples

- To join two or more tuples you can use the + operator

Example

Join two tuples:

```
tuple1 = ("a", "b", "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

The tuple() Constructor

- It also possible to use the tuple() constructor to make a tuple

Example

Using the tuple() method to make a tuple:

```
thistuple = tuple(("apple", "banana", "cherry"))
print(thistuple)
```

Tuple Methods

- Python has two built-in methods that you can use on tuples

count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Good luck, CS 1-2!