Information Technology

Year 2020/2021

Introduction to Programming

Programming in Python

Lists, Tuples, Sets

- How to create each structure?
  - myList = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
  - myTuple = ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday')
  - mySet = {'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'}

Lists

- What is a List?
  - Ordered collection of items
  - Can have any number of elements and of different data types
    - A list can have another list as an element
  - It is possible to search, add, and remove items from the list
    - Mutable data type because it can be altered by adding or removing elements
- E.g.:
  - myList = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday‘]

Working with Lists

```
In [1]: myList = [ 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday' ]
        print( myList )
```

```
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

```
In [2]: myList.append('Saturday') #Append a new value
        print(myList)
```

```
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
```

```
In [3]: print(myList[3]) #Print a specific item (4th in this case - 0, 1, 2, 3)
```

```
Thursday
```

```
In [4]: lastItem = myList.pop()  #Remove the last item from the list
        print(lastItem)
        print(myList)
```

```
Saturday
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

```
In [5]: print(myList[:3])  #Print the first 3 items
```

```
['Monday', 'Tuesday', 'Wednesday']
```

Operations with Lists

- append() – Adds an element to the end of the list
- clear() – Removes all elements from the list
- copy() – Returns a shallow copy of the list
- count() – Returns the total number of items passed as an argument
- extend() – Adds all elements of a list to some other list
- index() – Returns the index of an element (Note: If the same element appears multiple times in the list then the index of the very first match is returned)
- insert() – Inserts an element to the list at the defined index
- pop() – Eliminates and returns an element from the list
- remove() – Eliminates an element from the list
- reverse() – Reverses the order of all elements of the list
- sort() – Sort all elements of a list in the ascending order

Tuples

- What is a Tuple?
  - Similar to a List
    - Ordered collection of items
    - Can have any number of elements and of different data types
      - A tuple can have another tuple as an element
  - Unlike lists, tuples are immutable i.e. they can't be modified once created
- Operations on Tuples:
  - len() – Gives out the total length of some tuple
  - max() – Returns the biggest value from a tuple
  - min() – Returns the smallest value from a tuple
  - tuple() – Converts some list into a tuple

Working with Tuples

```
In [1]: myTuple = (55,97,82,85,87,99,88,64,88,62)
        print(myTuple)

        (55, 97, 82, 85, 87, 99, 88, 64, 88, 62)
```

```
In [2]: print(myTuple[3]) #Print the 4th item

        85
```

```
In [3]: print(myTuple[-4:]) #Print the last 4 items

        (88, 64, 88, 62)
```

```
In [4]: print(len(myTuple)) #Total number of items

        10
```

```
In [5]: print(max(myTuple)) #Biggest value
        print(min(myTuple)) #Smallest value

        99
        55
```

Sets

- What is a Set?
  - an unordered collection of simple objects in Python
  - It is mutable
  - has no duplicate elements
  - Can have any number of elements and of different data types
  - Allow testing for membership, checking whether a set is a subset of some other set and finding the intersection between two sets
    - Mathematical Set Theory

Operations with Sets

- add() – Adds an item to the set
  - Note: As sets don't have repeating values, the item that is to be added to a set must not be already a member of the set.
- clear() – Removes all items of the set
- difference() – Returns a set with all elements of the invoking set but not of the second set
- intersection() – Returns an intersection of two sets
- union() – Returns a union of two sets

Working with sets

In [1]:
```
oddNumbers = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19}
evenNumbers = {0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
primeNumbers = {1, 2, 3, 5, 7, 11, 13, 17, 19}
```

In [2]:
```
3 in oddNumbers  #Check whether a number belongs to a set
```

Out[2]: True

In [3]:
```
print(oddNumbers.difference(primeNumbers)) #Odd numbers that are not Prime
print(evenNumbers.difference(primeNumbers)) #Even Numbers that are nor prime
```

```
{9, 15}
{0, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

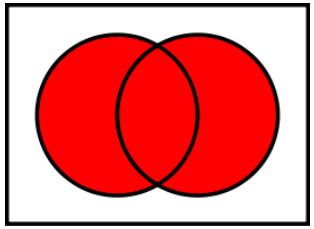
In [4]:
```
print(oddNumbers.intersection(primeNumbers)) #Odd numbers that are also prime
print(evenNumbers.intersection(primeNumbers))#Even Numbers that are also prime
```

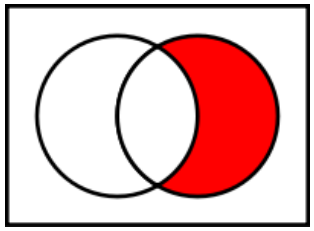
```
{1, 3, 5, 7, 11, 13, 17, 19}
{2}
```

In [5]:
```
print(oddNumbers.union(primeNumbers)) #Numbers that are odd or prime
print(evenNumbers.union(primeNumbers)) #numbers that are even or prime
```

```
{1, 2, 3, 5, 7, 9, 11, 13, 15, 17, 19}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
```

More Set Examples

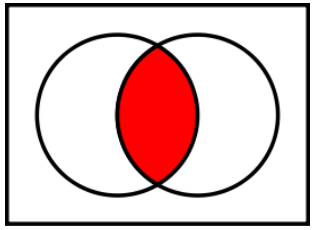


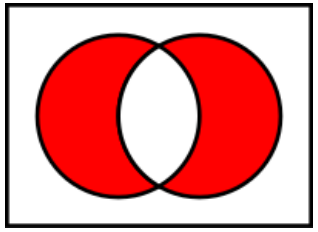


```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}

print (a.union(b))
print (a.intersection(b))
print (b.difference(a))
print (a.symmetric_difference(b))
```

```
{1, 2, 3, 4, 5, 6}
{3, 4}
{5, 6}
{1, 2, 5, 6}
```





Contributors

- List of authors/contributors to these materials:
  - Jesualdo Fernandes (2019)

- Credits
  - Part of these slides were based on previous work from Prof. Carlos Costa (ISEG)