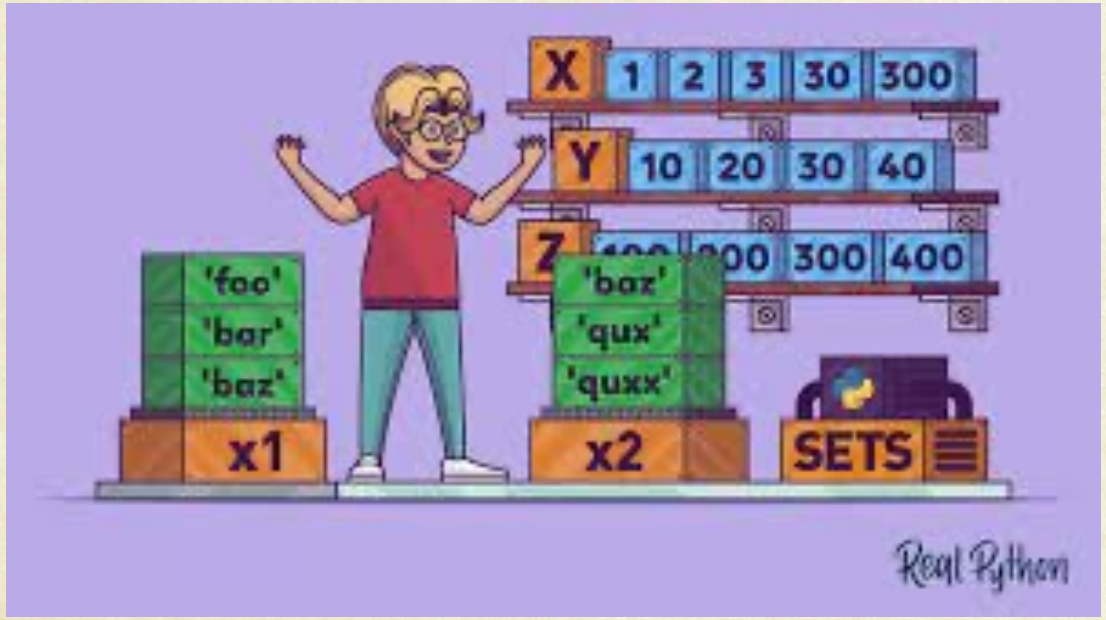
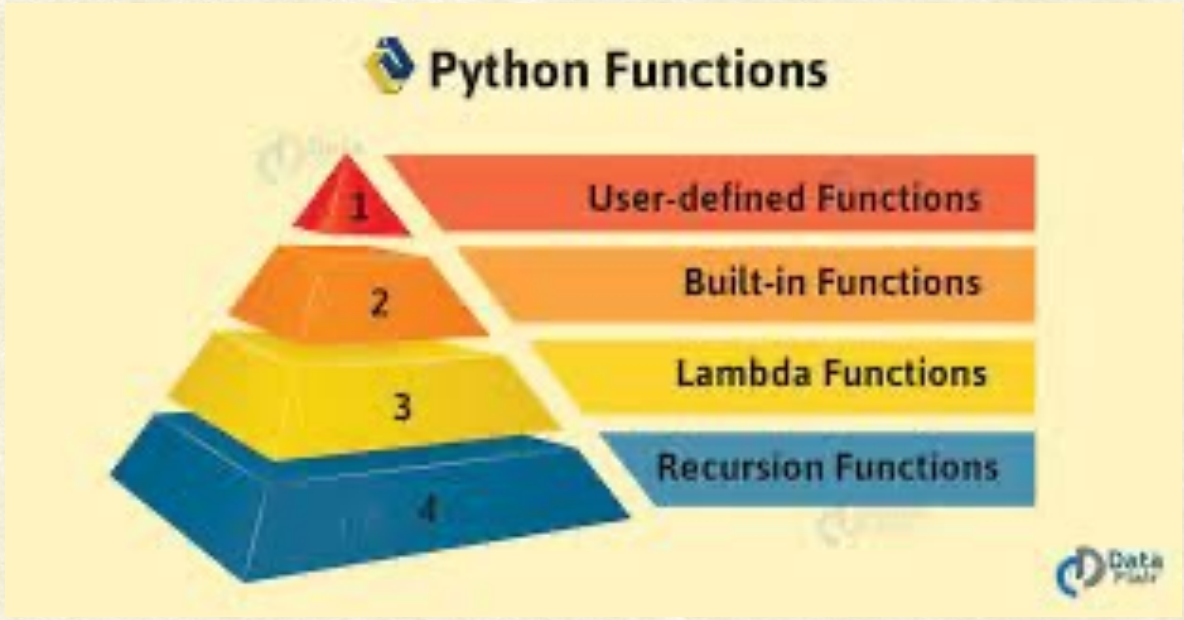
Lecture 7: Sets, Functions





Anoop M. Namboodiri

IIIT Hyderabad

Python Sets

The Unordered Collection

Set: The Unordered Collection

- Creating a set from a list or string or explicitly:
  - set1 = set(list1) or set1 = set(string1) or
  - set1 = { elt1, elt2, .., eltn}
- Supported operations:
  - Membership check: elt1 in set1 (returns True/False)
  - Union: set1 | set2
  - Intersection: set1 & set2
  - Difference: set1 - set2
  - Symmetric Difference: set1 ^ set2
- Elements in a set are unique. Duplicate elements are removed during creation (say from list)

Let us Code: Set

- Write a program to create two sets corresponding to words used in two different articles.
- Find the words used in article 1, but not in article 2
- Find the words used in article 2, but not in article 1
- Find the words that are used in both articles

```
words1 = set()
f = open("article1.txt", 'r')
for line in f:
    words1 = words1 | set(split(line))
```

Summary: Files, Data Structures

- Files: open, close, read, write, readline(s), “for line in file:”
- Lists: Heterogeneous, Mutable, Ordered: [a, b, c]
- Tuples: Immutable, Ordered, Can be a key: (a, b, c)
- Dictionaries: Mapping, Efficient: {k1:a, k2:b, k3:c}
- Sets: Unordered, Set operations: {a, b, c}

Python Functions

The Abstract Worker

Functions in Python

- Functions abstract a set of meaningful operations (a task)
- Essential to create: readable, reusable, scalable code.
  - Rule of Thumb: “A function should fit in a screen”
- Types of Functions:
  - Built-in; User-defined; Lambda; Recursion

```
def functionName(parameters):
    statement 1
    …
    statement n
    return expression
```

Function body

Function definition in Python

```
def functionName(par1:type, par2:type) -> retType:
    """ Documentation String (or docstring) """
    statement 1
    …
    statement n
    return expression
```

- Can define parameter and return types from Python 3.5
- Function documentation: print(functionName.__doc__)

Lambda Functions

- The ***lambda*** keywords defines an anonymous function
  - Returns a function object.
- Syntax:

```
fnName = lambda arguments: expression
```

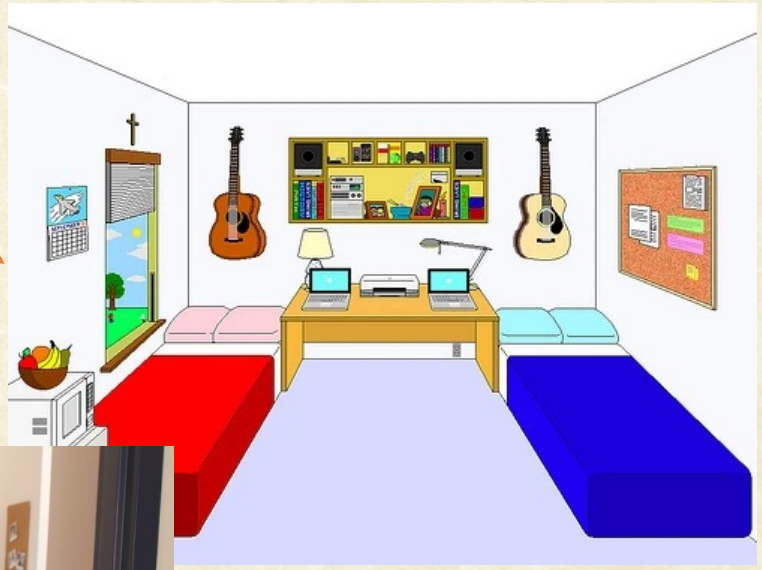
- Example:

```
lCube = lambda x: x*x*x
print(f"Cube of 3 is:{lCube(3)}")
```

- Single line functions
- Function arguments: `filter(fn,lst)`, `map(fn,lst)`, etc.

Classes: OOP with Python

OOPS: Give a separate room to each one !!!



Why Object Oriented?

- Groups data (attributes) of an object and the associated functions (methods)
- “Encapsulate” all data associated with an entity into its own storage or container
- Put all the functions related to these data also into the same container.
- External functions process the data through the associated functions
- Closely related to the concept of Data Structures

Public vs Private data

Private

- Medical History
- Finances
- Thoughts, fears, dreams
- Account Passwords

Public

- Name
- Designation
- Contact
- Face
- Resume
- Achievements

Classes in C++ vs Python

Class definition in Python vs C++

```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
```

```cpp
class Person{
  public:
    Person(myName, myAge){
      name = myName
      age = myAge
    }

  private:
   char name[32];
   int age;
};
```