Object-Oriented Programming

- The progress of abstraction
  - Assembly Language
  - FORTRAN, BASIC, C

Procedural Programming

- Focus on processes
- Collection of functions
- Data is declared separately
- Data is passed as arguments into functions
- Easy to learn

Procedural Programming

- Need to know the structure of the data
  - Changing data structure will cause functions not work
- As program gets larger
  - Difficult to maintain, understand, debug, extend, reuse, ..etc.

Object-Oriented Programming

- The progress of abstraction
  - Assembly Language
  - FORTRAN, BASIC, C
  - LISP, PROLOG
  - ………….C++, Java
  - ……………..Python

Object-Oriented Programming

- Everything is an object
- A program is a bunch of objects telling each other what to do by sending message
- Each object has its own memory made up of other objects
- Every object has a type
- All objects of a particular type can receive the
- same messages

Object-Oriented Programming

- What we really do in object-oriented programming is create new data types!

Object Oriented Programming

- Classes and Objects
  - Focus on classes that model real-world domain entities
  - Think at a higher level of abstraction
  - Easier to maintain in large programs

Concept of an Object

Object Oriented Programming

- Encapsulation
  - Objects contain data AND operations that work on that data
  - Abstract Data Type (ADT)

C++ Object-Oriented Programming

Object Oriented Programming

- Information-hiding
  - Implementation-specific logic can be hide
  - More abstraction
  - Users of the class code to the interface
  - Easier to test, debug, extend, and maintain

Object-Oriented-Programming

- Reusability
  - Reuse classes (encapsulation unit of data and operation)
  - Faster and higher quality

Object-Oriented-Programming

- Inheritance
  - Can create new classes in terms of existing classes
  - Reusability
  - Polymorphic classes
- Polymorphism and more….

Object-Oriented Programming

- Encapsulation
- Abstraction
- Inheritance
- Poly Morphism

Object-Oriented Programming

- Python

Class

- class describes a set of objects that have identical characteristics (data elements) and behaviors (functionality)

A type that is not built into Python

Class – instant

- class checking_account:

Class – Constructor

A function used to make new objects

```
class checking_account:
    def __init__(self):
```

Class – Constructor

A function used to initialize member data

```
class checking_account:
    def __init__(self, acct_number, balance):
```

Object-Oriented Programming

- Python

Class – Class vs. instance variables

```
class checking_account:
    bank_branch = 05421
    def __init__(self, acct_number, balance):
```

Object-Oriented Programming

- Python

Class –

Public vs. Private

Attributes

Functions (getter & setter)

Object-Oriented Programming

- Python

Class – Inheritance

```
class CS101:
   def __init__(self, fName, lName ):
      self.fName = fName
      self.lName = lName

class students(cs101):
   def __init__(self, fName, lName, grade):
      super().__init__(fName, lName)
      self.grade = grade
```

Python Data Types

- Python

Data Science: Python Data Types

Integer numbers	int	100
Float, real numbers	float	100.0
Text, string	str	“data science” ‘data science’
Boolean	bool	True or False

Data Science: Python Data Types

Array		
Dictionary (unordered, changeable, indexed, no duplicates) (Can access items by [“key”])	dict	{“course": “Data Science Fundamentals", “mode": “Online", “term": “Fall 2019” }
Lists (ordered, changeable, duplicates are ok) (can access items by [index#])	list	[“Adam”, “Betsy”, “Cherry“, “Betsy”]
Tuples (ordered, unchangeable, duplicates ok) (can access items by [index#])	tuple	(“may”, “june”, “july”,”july”, August)
Sets (unordered, unindexed, no duplicates) (cannot access items by index#)	set	{"apple", "banana", "craneberry"}