CS101 - Data Abstraction

OOPS - Module 1

Aravind Mohan

Allegheny College

March 9, 2021

Last Week

- Data types
- Conditional contructs
- Iterative constructs
- Functions (methods)

Refer Week2 slides, video, and notes ...

Homework Follow up

Find if n is a multiple of m?!

```
def is_multiple(n,m):
    if (n%m == 0):
      return True
    else:
      return False
print(is_multiple(4,2))
print(is_multiple(5,2))
```

PS the question R1 on page 51.

Homework Follow up

Find the sum of squares?!

```
def sum_of_squares(n):
    total = 0
    for i in range(0,n):
      print(i)
      total += (i*i)
    return total
print(sum_of_squares(4))
```

PS the question R4 on page 51.

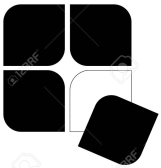
Discussion Based On ...

- GT (Goodrich Textbook) Chapter 01,02 [2.1,2.2,2.4]

Software Goals

Robustness - In addition to producing the correct output for anticipated inputs, we also want the software to handle unexpected inputs not known in advance.

Software Goals



Adaptability - Software should be able to evolve over time to changing conditions and environment.

Software Goals

Reusability - The same code should be usable as a component in different systems with varying applications.

Software Goals

- **Robustness**
- **Adaptability**
- **Reusability**

Can OOPS support these goals? ...

Classes and Objects

- A class defines behavior and data
- An object is an instance of a class
- Methods define behavior and variables store the data

GOAL: REUSABILITY

Constructor

- Definition: A special type of method called to create an object.
- This special method is called when a new object is created. Intialization happens in the constructor.

How does OOPS support these goals?

- Abstraction - Distill a complicated system down into fundamental parts. Specify what each operation does, and how it does it.
- Encapsulation - Different components of a software system should not reveal the internal details of their respective implementations. Data accessed through public interfaces.
- Modularity - Different components of a software system are divided into separate functional units, which later get integrated into a larger software system.

Object Oriented Programming (OOPs)

Display Student Report Card(OOPs way)!

```
class student:
  def __init__(self,id,name,gpa):
    self.id = id
    self.name = name
    self.gpa = gpa
  def report(self):
    print("————————————————————————")
    print("Student Id:", self.id)
    print("Student Name:", self.name)
    print("Student GPA:", self.gpa)
    print("————————————————————————")
```

PS student.py & stud-driver.py in the repo

Object Oriented Programming (OOPs)

Display Student Report Card (OOPs way)!

```
from student import student
s1 = student(101,"Alice",3.7)
s2 = student(102,"Bob",3.8)
s3 = student(103,"Cathy",3.9)
s1.report()
s2.report()
s3.report()
```

PS student.py & stud-driver.py in the repo

Can we store multiple values in one unit?

- Lists provide a structure to store any number of items.
- Items inside the list can be of different data type [Both Homogeneous and Heterogeneous].
- Indexing a list can lead to out of bound exception if not properly accessed.

An Implementation Of List

Display Places Visited!

```
visited = ['New York','London','India','China','Japan','Germany','S
print(visited)
```

PS places.py in the repo

Homework - Try Out Yourself

Coding challenge: Write a Python program that takes a list of exam scores and find the minimum, maximum, and average exam score.

Note: This is a very important problem in computer science and if one gets comfortable with this, then any future list related tasks may become easier.

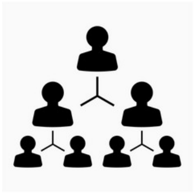
An Implementation Of Exception Handling

Divide user provided numbers!

```
first = int(input("Enter first:"))
second = int(input("Enter second:"))
try:
    divide = first/second
    print(divide)
except ZeroDivisionError:
    print ("WARNING: Invalid Equation")
```

PS divide.py in the repo

Inheritance



- Definition: A programming technique or mechanism for creating a hierarchy of classes.
- Automatically parent class methods are available in child class.
- Code redundancy is always a big problem.
- Single, Multilevel, and Multiple inheritance.

Inheritance Implementation

Single Inheritance

```
class dad():
  d_fname = "Peter"
  d_lname = "Smith"
  d_age = 50

from dad import dad
class daughter(dad):
  dg_fname = "Diana"
  dg_age = 18

from dad import dad
class son(dad):
  s_fname = "Bob"
  s_age = 20
```

PS the oops folder in repo.

Inheritance Implementation

Single Inheritance

```
from son import son
from daughter import daughter
s1 = son()
d1 = daughter()
print("Dad: " + s1.d_fname + " " +
          s1.d_lname + " is " +
          str(s1.d_age) + " years old.")
print("Son: " + s1.s_fname + " " +
          s1.d_lname + " is " +
          str(s1.s_age) + " years old.")
print("Daughter: " + d1.dg_fname
          + " " + d1.d_lname + " is " +
          str(d1.dg_age) + " years old.")
```

PS the oops/single folder in repo.

Inheritance Implementation

Multilevel Inheritance

```
class grandpa():
  g_fname = "Charles"
  g_lname = "Smith"
  g_age = 80

from grandpa import grandpa
class dad(grandpa):
  d_fname = "Peter"
  d_age = 50
```

PS the oops/multilevel folder in repo.

Inheritance Implementation

Multiple Inheritance

```
from grandpa import grandpa
class dad(grandpa):
    d_fname = "Peter"
    d_age = 50

class mom():
    m_fname = "Alice"
    m_lname = "Nicholas"
    m_age = 45

from dad import dad
from mom import mom
class daughter(dad,mom):
    dg_fname = "Diana"
    dg_age = 18
```

PS the oops/multiple folder in repo.

Reading Assignment

- GT (Goodrich Textbook) Chapter 01,02 [2.1,2.2,2.4]

Questions?

Please ask if there are any Questions!