

1. (a) Write a class for a *Student* with the below instance variable and methods.  
A *Student* should start (be initialized) with both a name and major.  
You may pick anything you like for the string representation of the object.

Student
name
major
get_major
set_major
__str__

- (b) Write a class for *Course* here at MnSU. The *Course* should start with a name and a number, but with no students. That is, it should not have the ability to be initialized with students in it. However, the *Course* should have the ability to add students as well as show all of the students in the *Course*.  
You may pick anything you like for the string representation of the object.

Course
course_name
course_number
students
get_number
set_number
add_student
show_student_enrollment
__str__

- (c) Create an instance of the *Course* class and add 2 *Students* to it.

2. (a) Write a class for a Duck with the below instance variables and methods.  
 The Duck object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A Duck says, “Quack.”  
 As part of your class, write a method called speak, that makes a duck quack!  
 Hint: for this method, you can just print the word Quack!

Duck
name
color
get_color
set_color
speak
__str__

- (b) Write a class for a Pond.  
 The class should start (instantiate) with a name, and no Ducks in it.  
 Write a method to add a Duck.  
 Write a method that makes all of the ducks in the pond quack one time each.  
 You may pick anything you like for the string representation of the object.

Pond
name
ducks
add_duck
ducks_quack
__str__

- (c) Create an instance of the Pond class and add two Ducks to it.  
 Call the method to make all ducks in your pond quack (ducks\_quack).  
 You can make up any names or colors for Ducks and a Pond.

3. (a) Write a class for a Lion with the below instance variables and methods.  
 The Lion object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A Lion says, "Roar."  
 As part of your class, write a method called roar, that makes a lion roar!  
 Hint: for this method, you can just print the word Roar!

Lion
name
age
get_age
set_age
roar
__str__

- (b) Write a class for a Zoo.  
 The class should start (instantiate) with a name, and no Lions in it.  
 Write a method to add a Lion.  
 Write a method that makes all of the lions in the zoo roar one time each.  
 You may pick anything you like for the string representation of the object.

Zoo
location
lions
add_lion
lions_roar
__str__

- (c) Create an instance of the Zoo class and add two Lions to it.  
 Call the method to make all lions in your zoo roar (lions\_roar).  
 You can make up any names or ages for Lions and a location for a Zoo.

4. (a) Write a class for an *Employee* with the below instance variables and methods.  
 An *Employee* should start (be initialized) with both a name and position.  
 You may pick anything you like for the string representation of the object.

Employee
name
position
get_position
set_position
__str__

- (b) Write a class for a *Department* within a company. The *Department* should start with a name and a budget, but with no employees. That is, it should not have the ability to be initialized with employees in it. However, the *Department* should have the ability to add employees as well as show all of the employees in the *Department*.  
 You may pick anything you like for the string representation of the object.

Department
dept_name
budget
employees
get_budget
set_budget
add_employee
show_staff_list
__str__

- (c) Create an instance of the *Department* class and add 2 *Employees* to it.

5. (a) Write a class for a Droid with the below instance variables and methods.  
 The Droid object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A Droid says, "Beep-Bloop-Blop."  
 As part of your class, write a method called `communicate`, that makes a droid communicate!  
 Hint: for this method, you can just print the words Beep-Bloop-Blop!

Droid
designation
series
get_series
set_series
communicate
__str__

- (b) Write a class for a Starship.  
 The class should start (instantiate) with a name, and no Droids in it.  
 Write a method to add a Droid.  
 Write a method that makes all of the droids in the starship communicate one time each.  
 You may pick anything you like for the string representation of the object.

Starship
name
droids
add_droid
droids_communicate
__str__

- (c) Create an instance of the Starship class and add two Droids to it.  
 Call the method to make all droids in your starship communicate (`droids_communicate`).  
 You can make up any designations or series for Droids and a name for a Starship.

6. (a) Write a class for a Post with the below instance variables and methods.  
 The Post object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A Post can be liked with a heart.  
 As part of your class, write a method called `add_like`, that adds 1 like to the post.  
 As part of your class, write a method called `display`, that prints out the post's caption.  
 Hint: for this method, you should increase a like counter.

Post
caption
likes
get_likes
add_like
display
<code>__str__</code>

- (b) Write a class for a Profile.  
 The class should start (instantiate) with a username, and no Posts in it.  
 Write a method to add a Post.  
 Write a method that displays all of the posts in the profile one time each.  
 You may pick anything you like for the string representation of the object.

Profile
username
posts
add_post
display_posts
<code>__str__</code>

- (c) Create an instance of the Profile class and add two Posts to it.  
 Call the method to display all posts in your profile (`display_posts`).  
 You can make up any captions for Posts and a username for a Profile.

7. (a) Write a class for a Product with the below instance variables and methods.  
 The Product object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A Product has a price and can be added to a cart.  
 As part of your class, write a method called `display_details`, that displays the product information.  
 Hint: for this method, you should print the name and price of the product.

Product
name
price
get_price
set_price
display_details
<code>__str__</code>

- (b) Write a class for a ShoppingCart.  
 The class should start (instantiate) with a `customer_id`, and no Products in it.  
 Write a method to add a Product.  
 Write a method that calculates the total price of all products in the cart.  
 You may pick anything you like for the string representation of the object.

ShoppingCart
customer_id
products
add_product
calculate_total
<code>__str__</code>

- (c) Create an instance of the ShoppingCart class and add two Products to it.  
 Call the method to calculate the total price of all products in your cart (`calculate_total`).  
 You can make up any names and prices for Products and a `customer_id` for a ShoppingCart.

8. (a) Write a class for a *LLM* with the below instance variables and methods.  
 A *LLM* should start (be initialized) with both a name and token\_limit.  
 You may pick anything you like for the string representation of the object.

LLM
name token_limit
get_token_limit set_token_limit __str__

- (b) Write a class for *AICompany*. The *AICompany* should start with a company\_name and a founding\_year, but with no LLMs. That is, it should not have the ability to be initialized with LLMs in it. However, the *AICompany* should have the ability to add LLMs as well as display all of the LLMs developed by the *AICompany*.  
 You may pick anything you like for the string representation of the object.

AICompany
company_name founding_year headquarters llms
get_headquarters set_headquarters add_llm display_models __str__

- (c) Create an instance of the *AICompany* class and add 2 *LLMs* to it.



9. (a) Write a class for a MenuItem with the below instance variables and methods.  
 The MenuItem object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A MenuItem has a price and can be added to an order.  
 As part of your class, write a method called show\_description, that displays the menu item information.  
 Hint: for this method, you should print the name and price of the menu item.

MenuItem
name
price
get_price
set_price
show_description
__str__

- (b) Write a class for a Restaurant.  
 The class should start (instantiate) with a restaurant\_name, and no MenuItems in it.  
 Write a method to add a MenuItem.  
 Write a method that displays all menu items with their prices.  
 You may pick anything you like for the string representation of the object.

Restaurant
restaurant_name
menu_items
add_menu_item
display_menu
__str__

- (c) Create an instance of the Restaurant class and add two MenuItems to it.  
 Call the method to display all menu items (display\_menu).  
 You can make up any names and prices for MenuItems and a restaurant\_name for a Restaurant.

10. (a) Write a class for a Book with the below instance variables and methods.  
The Book object should have the ability to be passed both initial values.  
You may pick anything you like for the string representation of the object.  
A Book has an author and can be checked out from a library.  
As part of your class, write a method called `display_info`, that displays the book information.  
Hint: for this method, you should print the title and author of the book.

Book
title
author
get_author
set_author
display_info
__str__

- (b) Write a class for a Library.  
The class should start (instantiate) with a `library_name`, and no Books in it.  
Write a method to add a Book.  
Write a method that displays all books with their authors.  
You may pick anything you like for the string representation of the object.

Library
library_name
books
add_book
display_catalog
__str__

- (c) Create an instance of the Library class and add two Books to it.  
Call the method to display all books in the library (`display_catalog`).  
You can make up any titles and authors for Books and a `library_name` for a Library.

11. (a) Write a class for a Song with the below instance variables and methods.  
 The Song object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A Song has an artist and can be added to a playlist.  
 As part of your class, write a method called play, that displays the song information.  
 Hint: for this method, you should print the title and artist of the song.

Song
title
artist
get_artist
set_artist
play
__str__

- (b) Write a class for a Playlist.  
 The class should start (instantiate) with a playlist\_name, and no Songs in it.  
 Write a method to add a Song.  
 Write a method that plays all songs one after another.  
 You may pick anything you like for the string representation of the object.

Playlist
playlist_name
songs
add_song
play_all
__str__

- (c) Create an instance of the Playlist class and add two Songs to it.  
 Call the method to play all songs in your playlist (play\_all).  
 You can make up any titles and artists for Songs and a playlist\_name for a Playlist.

12. (a) Write a class for a TVShow with the below instance variables and methods.  
 The TVShow object should have the ability to be passed both initial values.  
 You may pick anything you like for the string representation of the object.  
 A TVShow has a genre and can be added to a dashboard.  
 As part of your class, write a method called preview, that displays the show information.  
 Hint: for this method, you should print the title and genre of the show.

TVShow
title
genre
get_genre
set_genre
preview
__str__

- (b) Write a class for a NetflixDashboard.  
 The class should start (instantiate) with a profile\_name, and no TVShows in it.  
 Write a method to add a TVShow.  
 Write a method that displays all shows saved in the dashboard.  
 You may pick anything you like for the string representation of the object.

NetflixDashboard
profile_name
shows
add_show
display_recommendations
__str__

- (c) Create an instance of the NetflixDashboard class and add two TVShows to it.  
 Call the method to display all shows in your dashboard (display\_recommendations).  
 You can make up any titles and genres for TVShows and a profile\_name for a NetflixDashboard.