Homework 2 (100 points) **Due date:** September 26, 2025, 11:59 PM

Before you start…

Keep in mind these tips from your instructor:

- Read carefully the information presented in the instructions. You will be using custom classes that need to interact with each other and with Python's data types
- An instance of a class uses public attributes and methods outside of the class using the dot (.) operator. Given an instance of the Account class (as defined in the HandsOn Video in Module 3.1)

```
>>> sara_acct = Account('Sara')
>>> tom_acct = Account('Thomas')
>>> accounts = {1: sara_acct, 2: tom_acct}
>>> accounts[1].deposit(1000)
1000
>>> sara_acct.balance
1000
```

- Ask questions using our Homework 2 channel in Microsoft Teams
- **One more time:** read the instructions carefully and outline the purpose of each method. You are expected to use methods defined in other classes to complete methods in other sections instead of repeating code.

REMINDER: As you work on your coding assignments, it is important to remember that passing the examples provided does not guarantee full credit. While these examples can serve as a helpful starting point, it is ultimately your responsibility to thoroughly test your code and ensure that it is functioning correctly and meets all the requirements and specifications outlined in the assignment instructions. Failure to thoroughly test your code can result in incomplete or incorrect outputs, which will lead to deduction in points for each failed case.

60% > Passing all test cases 40% > Clarity and design of your code

ASSIGNMENT OVERVIEW

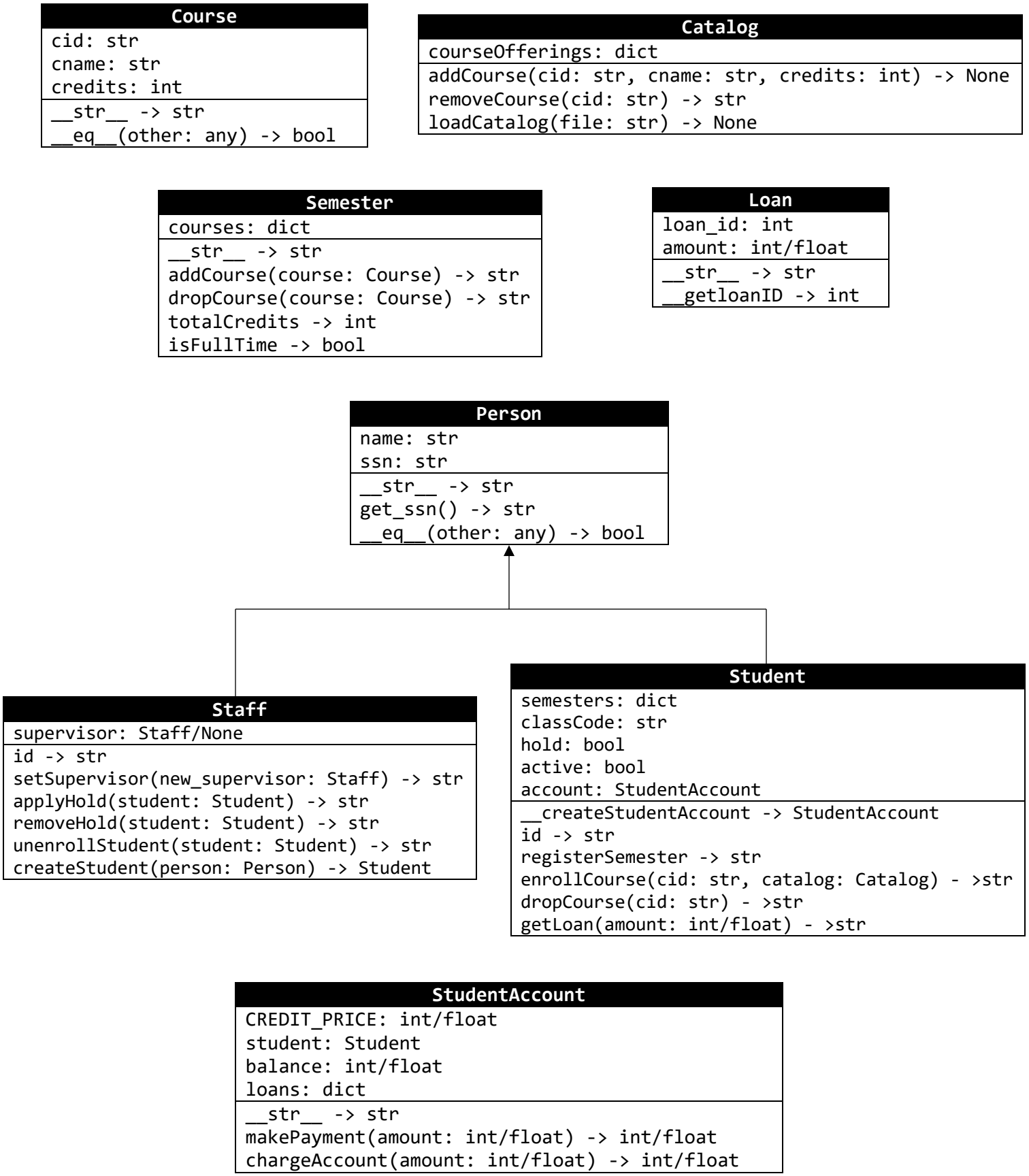
The main concept is to implement classes that represent a basic in-memory "school database". There are eight classes you must implement: Course, Catalog, Semester, Loan, Person, Staff, Student and StudentAccount. Each class has its own purpose which will be explained below.

- Course (no dependencies)
    - Represents a course, with attributes for id, name, and number of credits.
- Catalog (Course class must be implemented)
    - Stores a collection of Course objects through a dictionary, using id as the key.
- Semester (Course class must be implemented)
    - Stores a collection of Course objects taken together during a semester.
- Loan (no dependencies)
    - Represents an amount of money, with attributes for id and the loan amount.
- StudentAccount (Student class must be implemented)
    - Represents the financial status of a student.
- Person (no dependencies)
    - Represents a person, with attributes for a name and social security number.
- Staff (subclass of Person) (Person class must be implemented)
    - Represents a person who is a staff member and has a supervisor.
- Student (subclass of Person) (All classes must be implemented)
    - Represents a person who is a student that takes courses at the university.

A non-comprehensive list of concepts you should know to complete this assignment is:

- Basic class syntax / definition in Python (attributes, methods, definitions)
- Special/Magic methods
- Dictionaries
- Encapsulation
- Polymorphism through Operator and Method Overloading
- Inheritance
- Property methods
- Firm understanding of what it means for Python to be an "Object-Oriented" language

Class Diagrams for "in-memory" School Database



-

-- PART 1 ---

Section 1: The Course class

A storage class used for an individual course description. Stores the id, name, and number of credits for a course.

Attributes

Type	Name	Description
str	cid	Stands for course id, uniquely identifies a course like “CMPSC 132”.
str	cname	Stands for course name and is the long form of the course title.
int	credits	The number of credits a course is worth.

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the course as a string.
str	__repr__(self)	Returns the same formatted summary as __str__.
bool	__eq__(self, other)	Does an equality check based only on course id.

__str__(self) and __repr__(self)

Returns a formatted summary of the course as a string.

The format to use is: *cid*(*credits*): *cname*

Output	
str	Formatted summary of the course.

__eq__(self, other)

Determines if two objects are equal. For instances of this class, we will define equality when the course id of one object is the same as the course id of the other object. You can assume at least one of the objects is a Couse object.

Input (excluding self)		
any	other	The object to check for equality against a Course object.

Output	
bool	True if other is a Course object with the same course id, False otherwise.

Section 2: The Catalog class

Stores a collection of Course objects in a dictionary, key is the cid of the course, and value is the Course object that corresponds to the same cid.

Attributes

Type	Name	Description
dict	courseOfferings	Stores courses with the id as the key and Course as the value.

Methods

Type	Name	Description
str	addCourse(self, cid, cname, credits)	Adds a course with the given information.
str	removeCourse(self, cid)	Removes a course with the given id.
None	_loadCatalog(self, file)	Adds courses from a CSV file

addCourse(self, cid, cname, credits)

Creates a Course object with the parameters and stores it as a value in courseOfferings.

Inputs (excluding self)		
str	cid	The id of the course to add.
str	cname	The name of the course to add
int	credits	The number of credits the course is worth.

Output	
str	“Course added successfully”
str	“Course already added” if course is already in courseOfferings.

removeCourse(self, cid)

Removes a course with the given id.

Input (excluding self)		
str	cid	The id of the course to remove.

Output	
str	“Course removed successfully”
str	“Course not found” if a course with the given id is not in the dictionary.

_loadCatalog(self, file)

Reads a Comma Separated Values (csv) file with course information and adds its Course object to the courseOfferings dictionary. We are not using any libraries to read and process the csv file, instead, you will use string methods to clean the string for processing of the courses. The code for reading the file has been provided in the starter code and you should not modify it. The information that needs to be processed is in the string course_info.

Given the cmpsc_catalog_small.csv file with the following information (where you can assume the course name does not contain any commas):

```
CMPSC 132,Programming and Computation II,3
MATH 230,Calculus and Vector Analysis,4
PHYS 213,General Physics,2
CMPEN 270,Digital Design,4
CMPSC 311,Introduction to Systems Programming,3
CMPSC 360,Discrete Mathematics for Computer Science,3
```

The code provided to open, read and close the file sets course_info works as follows:

```
>>> file = "cmpsc_catalog_small.csv"
>>> with open(file, "r") as f:
...     course_info = f.read()
...
>>> course_info
'CMPSC 132,Programming and Computation II,3\nMATH 230,Calculus and Vector Analysis,4\nPHYS 213,General Physics,2\nCMPEN 270,Digital Design,4\nCMPSC 311,Introduction to Systems Programming,3\nCMPSC 360,Discrete Mathematics for Computer Science,3'
```

All string methods are allowed in this method, however, split() is the most helpful to process the string as needed.

Inputs (excluding self)		
str	file	Name of a csv file in the working directory

Section 3: The Semester class

Stores a collection of Course objects for a semester for a student.

Attributes

Type	Name	Description
dict	courses	Stores courses to be taken in the semester. The id of the course as the key and the Course object as the value.

Methods

Type	Name	Description
(many)	addCourse(self, course)	Adds a Course to the courses dictionary
(many)	dropCourse(self, course)	Removes a course from courses.
int	totalCredits(self)	A property method for the total number of credits.
bool	isFullTime(self)	A property method that returns True if this is full-time.

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the all the courses in this semester.
str	__repr__(self)	Returns the same formatted summary as __str__.

addCourse(self, course)

Adds a Course to the courses dictionary

Input (excluding self)		
Course	course	The Course object to add to this semester.

Output	
None	(Normal operation does not output anything)
str	“Course already added” if the course is already in this semester.

dropCourse(self, course)

Removes a course from this semester.

Input (excluding self)		
Course	course	The Course object to remove from this semester.

Output	
None	Normal operation does not output anything
str	“No such course” if the course is not in this semester.

totalCredits(self)

A property method (behaves like an attribute) for the total number of credits in this semester.

Outputs (normal)	
int	Total number of enrolled credits in this semester.

isFullTime(self)

A property method (behaves like an attribute) that checks if a student taking this semester would be considered full-time (taking 12 or more credits) or not.

Outputs (normal)	
bool	True if there are 12 or more credits in this semester, False otherwise.

__str__(self), __repr__(self)

Returns a formatted summary of the all the courses in this semester.

Use the format: *cid; cid; cid*; …

Output	
str	Formatted summary of the courses in this semester.
str	“No courses” if the semester has no courses.

Section 4: The Loan class

A class that represents an amount of money, identified by a pseudo-random number.

Attributes

Type	Name	Description
int	loan_id	The id for this loan, generated pseudo-randomly by __getloanID
int	amount	The amount of money loaned.

Methods

Type	Name	Description
int	__getloanID(self)	A property method that pseudo-randomly generates loan ids.

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the loan as a string.
str	__repr__(self)	Returns the same formatted summary as __str__.

__str__(self), __repr__(self)

Returns a formatted summary of the loan as a string.

Use the format: Balance: $*amount*

Output	
str	Formatted summary of the loan.

__getloanID(self)

A property method (behaves like an attribute) that pseudo-randomly generates loan ids. Use the random module to return a number between 10,000 and 99,999. The returned value should be saved to loan_id when initializing Loan objects. randint and randrange could be helpful here!

Outputs (normal)	
int	Pseudo-randomly generated id.

--- END OF PART 1 ---

--- PART 2 ---

Section 5: The StudentAccount class

This class represents a financial status of the student based on enrollment and is saved to a Student object as an attribute. This class should also contain an attribute that stores the price per credit, initially $1000/credit. This cost can change at any time and should affect the future price of enrollment for ALL students.

Attributes

Type	Name	Description
Student	student	The Student object that owns this StudentAccount.
numerical	balance	The balance that the student has to pay.
dict	loans	A dictionary that stores Loan objects accessible by their loan_id.
numerical	CREDIT_PRICE	Cost per credit

Methods

Type	Name	Description
numerical	makePayment(self, amount)	Makes a payment towards the balance.
numerical	chargeAccount(self, amount)	Adds an amount towards the balance.

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the loan as a string.
str	__repr__(self)	Returns the same formatted summary as __str__.

makePayment(self, amount)

Makes a payment by subtracting amount from the balance.

Input (excluding self)		
numerical	amount	The payment amount towards the balance.

Output	
numerical	Current balance amount.

chargeAccount(self, amount)

Adds amount towards the balance.

Inputs (excluding self)		
numerical	amount	The amount to add to the balance.

Output	
numerical	Updated balance amount.

__str__(self), __repr__(self)

Returns a formatted summary of the loan as a string. The format to use is (spread out over three lines):

Name: *name*
ID: *id*
Balance: *balance*

Output	
str	Formatted summary of the account.

Section 6: The Person class

This class is a basic representation of a person, storing name and social security number.

Attributes

Type	Name	Description
str	name	Full name of the person.
str	ssn	*Private* attribute of social security number formatted as “123-45-6789”.

Methods

Type	Name	Description
str	get_ssn(self)	Getter method for accessing social security number.

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the person as a string.
str	__repr__(self)	Returns the same formatted summary as __str__.
bool	__eq__(self, other)	Checks for equality by comparing only the ssn attributes.

get_ssn(self)

Getter method for accessing the private social security number attribute.

Output	
str	Social security number.

__str__(self), __repr__(self)

Returns a formatted summary of the person as a string.

The format to use is: Person(*name, ***-**-last four digits*)

Output	
str	Formatted summary of the person.

__eq__(self, other)

Determines if two objects are equal. For instances of this class, we will define equality when the SSN of one object is the same as SSN of the other object. You can assume at least one of the objects is a Person object.

Input (excluding self)		
many	other	The other object to check for equality with.

Output	
bool	True if other is a Person object with the same SSN, False otherwise.

Section 7: The Staff class

This class inherits from the Person class but adds extended functionality for staff members. Attributes, methods, and special methods inherited from Person are not listed in this section.

Attributes

Type	Name	Description
Staff	supervisor	A private attribute for this person's supervisor. By default, set to None.

Methods

Type	Name	Description
str	id(self)	Property method for generating staff's id.
(many)	setSupervisor(self, new_supervisor)	Updates the private supervisor attribute.
(many)	getSupervisor(self)	Property method for getting the supervisor.
(many)	applyHold(self, student)	Applies a hold on a student object.
(many)	removeHold(self, student)	Removes a hold on a student object.
(many)	unenrollStudent(self, student)	Sets a student's status to not active.
Student	createStudent(self, person)	Creates a Student object from a Person object

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the staff as a string.
str	__repr__(self)	Returns the same formatted summary as __str__.

id(self)

Property method (behaves like an attribute) for generating staff's id.

The format should be: 905+*initials*+*last four numbers of ssn*. (e.g.: 905abc6789). Ignore the security flaws this generation method presents and assume ids are unique.

Output	
str	Generated id for the staff member.

setSupervisor(self, new_supervisor)

Updates the private supervisor attribute.

Input (excluding self)		
Staff	new_supervisor	The new value to set for the supervisor attribute.

Output	
str	"Completed!"
None	Nothing is returned if new_supervisor is not a Staff object.

getSupervisor(self)

Property method (behaves like an attribute) for getting the supervisor.

Output	
Staff	Current value of supervisor.

applyHold(self, student)

Applies a hold on a student object (set the student's hold attribute to True).

Input (excluding self)		
Student	student	The student to apply a hold to.

Output	
str	"Completed!"
None	Nothing is returned if student is not a Student object.

removeHold(self, student)

Removes a hold on a student object (set the student's hold attribute to False).

Inputs (excluding self)		
Student	student	The student to remove a hold from.

Output	
str	"Completed!"
None	Nothing is returned if student is not a Student object.

unenrollStudent(self, student)

Unenrolls a student object (set the student's active attribute to False).

Inputs (excluding self)		
Student	student	The student to unenroll.

Output	
str	"Completed!"
None	Nothing is returned if student is not a Student object.

createStudent(self, person)

Creates a Student object from a Person object. The new student should have the same information (name, ssn) as the person and always starts out as a freshman ("Freshman").

Input		
Person	person	The Person object to create a Student object from.

Output	
Student	The new student created from the input Person object.

__str__(self), __repr__(self)

Returns a formatted summary of the staff member as a string.

The format to use is: Staff(*name, id*)

Outputs (normal)	
str	Formatted summary of the staff member.

Section 8: The Student class

This class inherits from the Person class and is heavy extended for additional functionality. Attributes, methods, and special methods inherited from Person are not listed in this section. The implementation of Course, Catalog, Semester, Loan and StudentAccount is needed for complete functionality for instances of this class, and you are expected to use methods from some of those classes in this section. This class works in conjunction with other classes, CORRECT FUNCTIONALITY OF THIS CLASS IS WORTH 30% OF YOUR HW GRADE

Attributes

Type	Name	Description
str	classCode	A string indicating the student's year ("Freshman", etc.).
dict	semesters	A collection of Semester objects accessible by an integer (1-n)
bool	hold	Indicates a hold on the student's account, defaults to False.
bool	active	Indicates if the student is actively enrolled, defaults to True.
StudentAccount	account	Creates and sets a StudentAccount object with the student's information to keep track of the student's balance.

Methods

Type	Name	Description
str	id(self)	Property method for generating student's id.
StudentAccount	__create StudentAccount (self)	Creates a StudentAccount object.
(many)	registerSemester(self)	Creates a Semester object.
str	enrollCourse(self, cid, catalog)	Enrolls the student in a course.
str	dropCourse(self, cid)	Drops a course from a semester.
(many)	getLoan(self, amount)	Creates a Loan object.

Special methods

Type	Name	Description
str	__str__(self)	Returns a formatted summary of the student as a string.
str	__repr__(self)	Returns the same formatted summary as __str__.

id(self)

Property method (behaves like an attribute) for generating student's id.

The format should be: *initials+last four numbers of ssn* (e.g.: abc6789). Ignore the security flaws this generation method presents and assume ids are unique.

Output	
str	Student's id.

__createStudentAccount(self)

Creates a StudentAccount object. This should be saved in the account attribute during initialization.

Output	
StudentAccount	Created StudentAccount object linked to the student.
None	Nothing is returned if student is not active.

registerSemester(self)

Creates a Semester object and adds it as a value to the semesters dictionary if the student is active and has no holds with key starting at 1 (next key is defined as max(key_value) +1). It also updates the student's year attribute according to the number of semesters enrolled. 'Freshman' is a first-year student (semesters 1 and 2), 'Sophomore' is a second-year student (semesters 3 and 4), 'Junior' is a third-year student (semesters 5 and 6) and 'Senior' for any enrollment with more than 6 semesters.

Output	
None	(Normal operation does not output anything)
str	"Unsuccessful operation" is returned if the student is inactive or has holds.

enrollCourse(self, cid, catalog)

Finds a Course object with the given id from the catalog and adds it to the courses attribute of the Semester object associated with the largest key in the semesters dictionary. Charge the student's account the appropriate amount of money.

Inputs (excluding self)		
str	cid	Course ID to search for.
Catalog	catalog	Catalog to search in.

Output	
str	"Course added successfully"
	"Unsuccessful operation" is returned if the student is inactive or has holds.
	"Course not found" is returned if no course with the given id is found.
	"Course already enrolled" is returned if the course is already enrolled.

dropCourse(self, cid)

Finds a Course object with the given id from the semester associated with the largest key in the semesters dictionary and removes it. When a course is dropped, only half the course cost is refunded to the student's account (cost is based on the current credit price).

Inputs (excluding self)		
str	cid	Course ID to search for.

Output	
str	"Course dropped successfully"
	"Unsuccessful operation" is returned if the student is inactive or has holds.
	"Course not found" is returned if no course with the given id is found.

getLoan(self, amount)

If the student is active and currently enrolled full-time (consider the item with the largest key in the semesters dictionary the current enrollment), it creates a Loan object for the student with the given amount, adds it to the student's account's loans dictionary, and uses the amount to make a payment in the student's account. Do NOT remove the line `random.seed(1)` from the constructor. This ensures replicable pseudo-random number generation across multiple executions, so your loan ids should match the doctest samples.

Inputs (excluding self)		
numerical	amount	The amount of money to get a loan for.

Output	
None	(Normal operation does not return anything)
str	"Unsuccessful operation" is returned if the student is inactive.
str	"Not full-time" is returned if the current semester is not full-time.

__str__(self), __repr__(self)

Returns a formatted summary of the student member as a string.
The format to use is: Student(*name*, *id*, *year*)

Outputs (normal)	
str	Formatted summary of the student member.

--- END OF PART 2 ---