project.classes package

Submodules

project.classes.Address module

class `project.classes.Address.`**`Address`**(*street*, *city*, *state*, *zip_code*)

Bases: **`object`**

Stateful class that holds address attributes.

street (str): Street of user's address. city (str): City of user's address. state (str): State of user's address. zip_code (str): Zip Code of user's address.

`get_city`()

`get_state`()

`get_street`()

`get_zip_code`()

project.classes.Climber module

class `project.classes.Climber.`**`Climber`**(*username*, *info=None*, *routes=None*)

Bases: **`project.classes.Climber.User`**

The climber class holds all of a user's information. Parameters ----------

username (str): Unique string denoting user's username. info (ClimberInfo): Object holding personal information about the user. routes (list<Route>): List of route objects created by the user.

`get_id`()

`get_info`()

`get_routes`()

`get_username`()

class `project.classes.Climber.`**`User`**

Bases: **`object`**

Abstract class used to make Flask's login library work with
climber class objects by adding required methods to the child class.

None

classmethod **`get_id`**()

`is_active`()

`is_anonymous`()

`is_authenticated`()

project.classes.ClimberInfo module

project.classes.ContactInfo module

class `project.classes.ContactInfo.`**`ContactInfo`**(*address*, *phone_number*)

Bases: **`object`**

Info class that takes in

address and phone number as parameters.

address (Address): Address object holding user's address information. phone_number (str): The user's phone number.

`get_address`()

`get_phone_number`()

project.classes.Hold module

class `project.classes.Hold.`**`BaseHold`**(*hold_type*)

Bases: **`project.classes.Hold.Hold`**

Concrete class used in the decorator design pattern. Implements return holds

and doesn't recursively class decorators because its of BaseHold type.

None

`return_holds`()

class `project.classes.Hold.`**`Decorator`**(*hold_type*, *component*)

Bases: **`project.classes.Hold.Hold`**, **`abc.ABC`**

Abstract class used in the decorator design pattern. Forces children to

take in hold_types and component, as well as implement return holds.

hold_type (str): String denoting the type of holds selected from the HoltTypes enum. component (BaseHold | DecoratorHold):

classmethod **`return_holds`**()

Returns list of hold types.

class `project.classes.Hold.`**`DecoratorHold`**(*hold_type*, *component*)

Bases: **`project.classes.Hold.Decorator`**

Concrete hold class used in the decorator design pattern. Builds up a list

recursively by calling return_holds on the component object.

None

`return_holds`()

Returns list of hold types.

class `project.classes.Hold.`**`Hold`**(*hold_type*)

Bases: **`abc.ABC`**

Abstract class used in the decorator design pattern. Forces children to

take in hold_type as a parameter.

hold_type (str): String denoting the type of holds selected from the HoltTypes enum.

classmethod **`return_holds`**()

project.classes.Route module

class `project.classes.Route.`**`Bouldering`**(*name*, *location*, *holds*, *actual_difficulty*, *felt_difficulty*)

Bases: **`project.classes.Route.Route`**

Child class that inherits from Route. Implements the calculate_effort() method. Parameters ----------

None

`calculate_effort`()

`gear_required` = *False*

`required_climbers` = *1*

class `project.classes.Route.`**`Lead`**(*name*, *location*, *holds*, *actual_difficulty*, *felt_difficulty*)

Bases: **`project.classes.Route.Route`**

Child class that inherits from Route. Implements the calculate_effort() method. Parameters ----------

None

`calculate_effort`()

`gear_required` = *True*

`required_climbers` = *2*

class `project.classes.Route.`**`Route`**(*name*, *location*, *holds*, *actual_difficulty*, *felt_difficulty*)

Bases: **`abc.ABC`**

The route class manages all aspects of a route object that a climber adds.

The route class is an abstract class forcing subclasses to implement the calculate_effort() method which is dependent on the type of route to be created.

name (str): The nane of the route. location (str): The location of the route. holds (Hold): Hold object with each hold found on the route. actual_difficulty (int): The rating given by the gym. felt_difficulty (int): The rating that the climber gives to the route.

`calculate_effort`()

`get_actual_difficulty`()

`get_felt_difficulty`()

`get_holds`()

`get_location`()

`get_name`()

class `project.classes.Route.`**`TopRope`**(*name*, *location*, *holds*, *actual_difficulty*, *felt_difficulty*)

Bases: **`project.classes.Route.Route`**

Child class that inherits from Route. Implements the calculate_effort() method. Parameters ----------

None

`calculate_effort`()

`gear_required` = *True*

`required_climbers` = *2*

project.classes.RouteFactory module

project.classes.Workout module

class `project.classes.Workout.`**`Workout`**(*routes*, *requested_workout_info*, *workout_algorithm*)

Bases: **`object`**

Stores all the information regarding to a workout. Parameters ----------

routes (list<Route>): List of routes that matched a climber's preference. requested_workout_info (dict): Object containing all of the climber's workout preferences. workout_algorithm (WorkoutStrategy): Workout algorithm determed by the strategy design pattern.

`get_name`()

`get_routes`()

project.classes.WorkoutStrategy module

Module contents