Assignment 1

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 date_adt.DateT Class Reference

An ADT that represents Date.

Public Member Functions

- def __init__ (self, d, m, y)

  Constructor that inializes the object with year, month and day.
- def day (self)

  Getter for day.
- def month (self)

  Getter for month.
- def year (self)

  Getter for year.
- def next (self)

  Find the next day from the current object.
- def prev (self)

  Find the day before current object.
- def before (self, d)

  Finds if the current date is before the d object.
- def after (self, d)

  Finds if the current date is after the d object.
- def equal (self, d)

  Checks if the dates are the same.
- def add_days (self, n)

  Adds n days to the current DateT object.
- def days_between (self, d)

  Find the days between the current object and parameter object.

Public Attributes

- **y**
- **d**
- **m**

3.1.1 Detailed Description

An ADT that represents Date.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 __init__()

```
def date_adt.DateT.__init__ (
              self,
              d,
              m,
              y )
```

Constructor that inializes the object with year, month and day.

Using a try and except, the inputted date is tested with datetime to see if its valid then save them in class variables else raise error.

Parameters

m	- Month, d - Day, y- Year

3.1.3 Member Function Documentation

3.1.3.1 add_days()

```
def date_adt.DateT.add_days (
              self,
              n )
```

Adds n days to the current DateT object.

Creates a datetime object with the current object, then uses the timedelta function to add n days and finally extracts the year, month and day. If you enter a decimal number for n it will round to nearest whole number and then add those many days. Accepts negative days (to go back days)

Parameters

n	- number of days to add on to the current date (int)

Returns

a DateT object with the new date

3.1.3.2 after()

```
def date_adt.DateT.after (
              self,
              d )
```

Finds if the current date is after the d object.

Creates a datetime object with current object and a datetime object with the parameter d. Then compare the datetime objects

Parameters

d	- DateT object to compare with

Returns

True if the current object is after param d else False

3.1.3.3 before()

```
def date_adt.DateT.before (
              self,
              d )
```

Finds if the current date is before the d object.

Creates a datetime object with current object and a datetime object with the parameter d. Then compare the datetime objects

Parameters

d	- DateT object to compare with

Returns

True if the current object is before param d else False

3.1.3.4 day()

```
def date_adt.DateT.day (
              self )
```

Getter for day.

Returns

The day as an int

3.1.3.5 days_between()

```
def date_adt.DateT.days_between (
              self,
              d )
```

Find the days between the current object and parameter object.

Creates a datetime object with the current object and a datetime object with the parameter object. Then find the difference between the two dates by subtracting.

Parameters

d	- DateT object find the difference in days from

Returns

the absolute value of the difference since the user can enter a date that is before the current date

3.1.3.6 equal()

```
def date_adt.DateT.equal (
              self,
              d )
```

Checks if the dates are the same.

Compares the day, month and year among the two objects

Parameters

d	- DateT object to compare with

Returns

True if the date are the same else False

3.1.3.7 month()

```
def date_adt.DateT.month (
              self )
```

Getter for month.

Returns

The month as an int

3.1.3.8 next()

```
def date_adt.DateT.next (
              self )
```

Find the next day from the current object.

Create a datetime object with current object and use timedelta function to add one day and extract year, month and day

Returns

DateT object of next day

3.1.3.9 prev()

```
def date_adt.DateT.prev (
              self )
```

Find the day before current object.

Creates a datetime object with current object and use timedelta function to subtract one day and extract year, month, day

Returns

DateT object of the previous day

3.1.3.10 year()

```
def date_adt.DateT.year (
              self )
```

Getter for year.

Returns

The year as an int

The documentation for this class was generated from the following file:

- src/date_adt.py

3.2 pos_adt.GPosT Class Reference

An ADT that represents global position coordinates.

Public Member Functions

- def __init__ (self, latitude, longitude)

  Constructor that creates a new GPosT object with latitude and longitude (in degrees)
- def lat (self)

  Getter for latitude.
- def long (self)

  Getter for longitude.
- def west_of (self, p)

  Checks if current position is west of parameter position.
- def north_of (self, p)

  Checks if current position is north of parameter position.
- def equal (self, p)

  Checks if the distance is equal by calculating the distance.
- def move (self, b, d)

  Moves the current object d (km) towards b (bearing)
- def distance (self, p)

  Calculate the distance between current position and position p.
- def arrival_date (self, p, d, s)

  Calculate the arrival date starting from the current position and traveling towards position p at s kilometers per day.

Public Attributes

- **latitude**
- **longitude**

3.2.1 Detailed Description

An ADT that represents global position coordinates.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 __init__()

```
def pos_adt.GPosT.__init__ (
              self,
              latitude,
              longitude )
```

Constructor that creates a new GPosT object with latitude and longitude (in degrees)

First check if latitude and longitude are in range then save them to the class variables else throw value error

Parameters

latitude	and longitude (in degrees) are real numbers

3.2.3 Member Function Documentation

3.2.3.1 arrival_date()

```
def pos_adt.GPosT.arrival_date (
              self,
              p,
              d,
              s )
```

Calculate the arrival date starting from the current position and traveling towards position p at s kilometers per day.

First calculate the distance between current position and desired position (p), then divide the distance by s to get days it takes to reach that position. Finally use the DateT method add_days to add the days needed to travel the distance. The reason I took the ceil of days is because I want to give the date that they will reach the position by. Also, it doesnt make sense to input a negative speed in this case.

3.2.3.2 distance()

```
def pos_adt.GPosT.distance (
              self,
              p )
```

Calculate the distance between current position and position p.

Using the formula provided in the specifications https://www.movable-type.co.uk/scripts/latlong.html under Distance

Parameters

p	- GPosT object that we are finding the distance between

Returns

the distance in kilometers between the two positions

3.2.3.3 equal()

```
def pos_adt.GPosT.equal (
              self,
              p )
```

Checks if the distance is equal by calculating the distance.

Using the formula provided in specifications https://www.movable-type.co.uk/scripts/latlong.↩html

Parameters

p	- GPosT object comparing with

Returns

True if the distance is less than 1 km else False

3.2.3.4 lat()

```
def pos_adt.GPosT.lat (
              self )
```

Getter for latitude.

Returns

the latitude as a double

3.2.3.5 long()

```
def pos_adt.GPosT.long (
              self )
```

Getter for longitude.

Returns

the longitude as a double

3.2.3.6 move()

```
def pos_adt.GPosT.move (
              self,
              b,
              d )
```

Moves the current object d (km) towards b (bearing)

Using the formula provided in the specifications https://www.movable-type.co.uk/scripts/latlong.html under Destination point given distance and bearing from start point then the new position gets updated to be the current. Firstly convert all degrees into radians as the formula requires radians. Also I check if the new positions are greater than their ranges, if so then I apply a formula to normalize them which can be found from the website given in the specifications https://www.movable-type.co.uk/scripts/latlong.html

Parameters

b	- bearing type real and d - distance in km type real

3.2.3.7 north_of()

```
def pos_adt.GPosT.north_of (
              self,
              p )
```

Checks if current position is north of parameter position.

Parameters

p	- GPosT object which you are comparing with

Returns

True if the current position is north of parameter position else False

3.2.3.8 west_of()

```
def pos_adt.GPosT.west_of (
              self,
              p )
```

Checks if current position is west of parameter position.

Parameters

p	- GPosT object which you are comparing with

Returns

True if the current position is west of parameter position else False

The documentation for this class was generated from the following file:

- src/pos_adt.py

Chapter 4

File Documentation

4.1 src/date_adt.py File Reference

An abstract data type that represents Date and allows for other functions to manipulate Date.

Classes

- class date_adt.DateT

  An ADT that represents Date.

4.1.1 Detailed Description

An abstract data type that represents Date and allows for other functions to manipulate Date.

Author

Dhruv Bhavsar

Date

Jan 15 2020

4.2 src/pos_adt.py File Reference

An abstract date type for global position coordinates and allows for several functions on them.

Classes

- class pos_adt.GPosT

  An ADT that represents global position coordinates.

4.2.1 Detailed Description

An abstract date type for global position coordinates and allows for several functions on them.

Author

Dhruv Bhavsar

Date

Jan 16 2020

Index