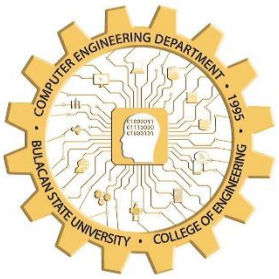
MATHEMATICS OF DISCOUNTING

(DiscntPY)



CPE 102L

OBJECT ORIENTED PROGRAMMING

GROUP 3

PREPARED BY:

CATACUTAN, ZIAN JOLO M.

PASCUAL, STEVEN ROD C.

CABUHAT, LAURENT SJ

VENERACION, JOHN ELBERT C.

NOVEMBER, 2019

INTRODUCTION

In this capstone project, the proponents sought to create a function which entails the basic mathematics of discounting. According to Merriam Webster (2019) discounting is a reduction made from the gross amount or value of something. While discounting may seem a basic mathematic as we have been taught in our high school curriculum, it is actually a much broader subject. The proponents decided to include other discounting methods in the function, thus making the capstone cover a much wider range of computations in the mathematics of discounting.

In order to program the function, variables representing the values used in computing discount were used:

Amount – This is the principal amount wherein all of the computations will be based on.

Proceeds – In other words the loan amount, are the total amount the creditor will receive from the debtor. Proceeds can be computed by getting the difference of maturity value and discount.

Discount – It is a reduction made from the gross amount or value of something.

Discount rate – Is the percentage amount that will be deducted from the original amount.

Time in years – This the amount of time given by the creditor before the amount reaches its maturity value.

The developers used accepted keywords as variable representations, the said keywords are as follows:

Variable	Meaning	Default Value
‘A’	Amount	0
‘MV’	Maturity Value	0
‘FV’	Future Value	0
'td_o_actual'	Actual number of days in an Ordinary Interest	0
'td_e_actual'	Actual number of days in an Exact Interest	0
'td_o_approx'	Approximated number of days in an Ordinary Interest	0

'td_e_approx'	Approximated number of days in an Exact Interest	0
'tm'	Time in MONTHS	0
'ty'	Time in YEAR	0
'P'	Proceeds	0
'D'	Discount	0
‘B’	Bank Discount	0
‘I’	Interest	0

It is important to note that the user needs to input values in these variables, elsewise, the function won’t do anything.

The capstone is a self-computing function, which means once the user is done in inputs, the function will take care of the rest and shall output a result based on the given values.

On the other hand, the developers programmed the function in a way the it will automatically solve the unknown variables and will return the main variables. On addition, inputs and results can be also viewed in a formatted way as well as data of calculations can be saved in a text document for future purposes.

As a result, the proponents successfully created a function that will serve to cater the needs of a user in terms of mathematics of discounting in an accessible and convenient way.

MANUAL

Our function has a built-in docstring to help you understand how it works and what it only accepts. We suggest that after importing the module you type the function ‘help(module_name)’ and on the ‘module_name’ is the name of the module that suites the version of your interpreter.

Importing the Module:

The Module has two versions.

1. **DiscntPy** → is for Python interpreter 3.7 and above
2. **DiscntPy27** → is for Python 2.7 interpreters

It was done with two compatible versions because of the print function that has a slight difference in both versions of interpreters. So be sure to use the correct version that, matches the interpreter you use to avoid errors.

HOW TO USE:

Option 1:

```
from DiscntPy import Simple_Discount
```

Option 2:

```
from DiscntPy import *
```

Note:

DiscntPy can be changed to **DiscntPy27** if you are using Python 2.7

- **Option 1: Simple_Discount** is the class that contains all the useable methods.
- **Option 2:** The asterisk indicates ‘all that is inside the module name **DiscntPy’.**
- Any other syntax to import the module would cause an error.

Initializing the Module:

The Function was written as Object Oriented (it has a class and objects). And all the functions it contains is a ‘method’ so to access

them we first need to create an object for the ‘main class’ and we use the (.) dot operator to call each methods.

The module only contains one main class and it is called:

‘Simple_Discount’

To create an object for the class use the syntax of:

object_name = Simple_Discount()

Note: ‘**object_name**’ can be **any name** of your choice this will serve as the storage to access all the process performed by the function.

After creating an object for the class, the method:

def __init__(self)

will initialize.

This method is **not callable**, and it **takes no arguments** it is only for the initializations of variables and values to be returned after calculation. The default values are:

- → **Amount** of Face Value or Maturity Value is Zero = **0**
- → **Discount Rate** is Zero = **0**
- → **Time** is Zero = **0**
- → **Proceed** is Zero = **0**
- → **Discount** or Bank Discount or Interest is Zero = **0**

To call each variables use the syntax of:

object_name.variable_name

→**'object_name'** must be an object of the class 'Simple_Discount()'

→**'variable_name'** are

Amount, dis_rate, ty, tm, td, Proceed, Discount

→**'td'** will only return approximated number of days if your input is in months.

HOW TO USE: syntax is for Python 3.7

```
Line 1:|   a = Simple_Discount()
Line 2:|   print (a.Amount)
Line 3:|   x = a.Proceed
```

Line 1: setting 'a' as an object of the class Simple_Discount()

Line 2: printing out the value for the Amount or Face Value or Maturity Value

Line 3: You can also assign it to a **variable**. Here x is equal to the amount of 'Proceed' for object 'a'

List of useable Function/Methods:

The module only offers **five (5)** main **functions/methods** to be used.

This are:

1) discnt_inputs()
2) formula_for()
3) save()
4) show_inputs()
5) show_computed()

- **discnt_inputs(self, val=[0, 0, 0, 0, 0], var=[], decimal=4)**

This method is use to input all the 'values' and 'variables' that you have based on the problem you are trying to solve.

NOTE:

→ **'val':**

- acceptable values are of type --> [int, float]
- only accepts 5 values as minimum and max if you don't have that 'value' for a certain 'variable' place 0 or 0.0 and it will be automatically detected as unknown.

→ **'var':**

- acceptable variables are of type -- > [str]
- you can show all the accepted variables by using the method '.accepted_var':

syntax: **print (object_name.accepted_var)**

List of accepted var:

'A','d','td_o_actual','td_e_actual','td_o_approx','td_e_approx','ty','tm','P','D','B','I','MV','FV'

Accepted variables 'var'	Meaning
'A' or **'MV'** or **'FV'**	Amount, Maturity Value, or Future Value
'd'	Lower case 'd' is for discount rate
'td_(...)'	Time in DAYS. It contains four types: → **'td_o_actual'** for Actual number of days in an Ordinary Interest → **'td_e_actual'** for Actual number of days in an Exact Interest

	→ **'td_o_approx'** for Approximated number of days in an Ordinary Interest → **'td_e_approx'** for Approximated number of days in an Exact Interest
'tm'	Time in Days
'ty'	Time in Years
'P'	Upper case 'P' is for Proceed
'D' or **'B'** or **'I'**	Discount or Bank Discount or Interest

→ **'decimal':**

- You can customize the number of decimal of the results by a number of your choice. By default four (4) is set.

- **def formula_for(self,discount_var_name,title='')**

This method returns all possible formulas for a certain unknown in relation to DISCOUNTING.

- **Not Case Sensitive.**

It takes one argument **'dicount_var_name'.**

HOW TO USE:

object_name.formula_for(discount_var_name, title = '')

discount_var_name	Meaning
'ALL'	Returns all the formula for Simple Discount.
'LEGEND'	Returns the variable definition. It is suggested to call it first before calling a table for a single

	formula. If you use ‘ALL’ legend is automatically included.
'AMOUNT' or 'MATURITY VALUE' or 'FACE VALUE'	Returns all the formula for Amount.
'DISCOUNT' or 'BANK DISCOUNT' or 'INTEREST'	Returns all the formula for Discount.
'DISCOUNT RATE'	Returns all the formula for Discount rate.
'TIME'	Returns all the formula for Time.

List of Accepted: ‘discount_var_name’

The second argument **‘title’** is optional, this only change the default header title for each table of formula you call, if you leave it blank the default titles will be returned. In **‘LEGEND’** and **‘ALL’** this does nothing.

EXAMPLE: Leaving it Blank or Unchanged.

```
Line 1:|    x = Simple_Discount()
Line 2:|    x.formula_for('amount')
```

Output:

```
ALL THE FORMULA TO SOLVE FOR THE VALUE OF 'AMOUNT' OR
'MATURITY VALUE' OR 'FACE VALUE':

 ======================
 |USING |FORMULA       |
 |D,d,t |A = D/(d x t) |
 |D,P   |A = D + P     |
 |P,d,t |A = P/(1 - dt)|
 ======================
```

Giving a Custom Title.

```
Line 1:|    x = Simple_Discount()
Line 2:|    x.formula_for('amount',’This is the new Title’)
```

Output:

```
This is the new Title
 ======================
 |USING |FORMULA       |
 |D,d,t |A = D/(d x t) |
 |D,P   |A = D + P     |
 |P,d,t |A = P/(1 - dt)|
 ======================
```

- **def save(self,border = '',title = '',smode = '')**

This method will allow you to save your **INPUTS** and **RESULTS** in a text document named **'RESULTS_Discount.txt'**

It takes three arguments **'border'** and **'title'** and **'smode'.**

Arguments	Meaning
'border'	Let you change the boundary decorators for inputs and results.
'title'	Set a custom title for each result.
'smode'	- Will allow you to choose if you want to append current computation on the file or replace all saved computation in the file. - If you leave it empty by, default the saving mode is set to **'append'** or **'a'**. - It is **CASE SENSITIVE.** **Only accept:** **'a'** to append. **'w'** to write or replace.

HOW TO USE:

object_name.save('-','COMPUTATION NUMBER 1','a')

- **def show_inputs(self,border='')**

This method will show the inputs in a more arranged and organized way,

→ It only takes one argument 'border' which in default was set to empty character.

→ By placing any character in 'border' it will be printed as the border for the output.

→ If you want a new line to be printed out at the top and bottom of the showed inputs just make the border equal to space ' '.

→ The number of space will be treated as the number of new lines to be printed.

HOW TO USE:

object_name.show_inputs('-')

- **def show_computed(self,border='')**

This method will show the calculated unknowns in a more arranged and organized way

→ It only takes one argument 'border' which in default was set to empty character.

→ By placing any character in 'border' it will be printed as the border for the output.

→ If you want a new line to be printed out at the top and bottom of the showed computed output just make the border equal to space ' '.

→ The number of space will be treated as the number of new lines to be printed.

HOW TO USE:

object_name.show_computed('-')

REFERENCES:

Southwest Tech Math/Science Center. (2016, February, 1). *Introduction to Simple Discount Notes – Math with Business Applications, Simple Interest Chapter*. Retrieved from: https://youtu.be/3RbL6BkQ8Jo

discount. 2019. In *Merriam-Webster.com*. Retrieved November 4, 2019, from https://www.merriam-webster.com/dictionary/discount