Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [1]:
```
NAME = "Sharjil Mohsin, MAC ID: mohsis2, Student #: 400139443, Date: January 18, 2018"
COLLABORATORS = "N/A"
```

Minor Assignment 2

ENGINEER 1D04
Dr. Ashgar Bokhari
McMaster University, Fall 2017

Background

The formula for computing the future value of an investment with *compound interest* is

$$p(1 + \frac{r}{n})^{nt}$$

where

- *p* is the initial principal
- *r* is the interest rate
- *n* is the number of times the interest is compounded per year
- *t* is the time expressed as a number of years.

The formula for computing the future value of an investment with *continuous compound interest* is:

$$\lim_{n\to\infty} p\left(1+\frac{r}{n}\right)^{nt} = pe^{rt}$$

(Notice the similarity between this limit and the limit given in Minor 1.)

Overview

Banks sometimes offer continuous compound interest, which is, roughly speaking, compounding interest infinitely often. The purpose of this assignment is to write a Python program that computes the future values of investments with compound interest and with continuous compound interest and then compares the difference between them.

The program takes as input the following amounts:
a. The initial principal (*p*)
b. The interest rate (*r*)
c. The number of times the interest is compounded per year (*n*)
d. The time expressed as a number of years (*t*)

Design, implement, and test a program that satisfies the requirements below.

Requirements

1. Using these values, implement a function to compute 5 values:

   a. A = the future value of *p* with compound interest, given in the first equation under the **Background** section.

   b. B = the future value of p with compound interest, given in the first equation under the **Background** section, but calculated using a `for` loop instead. (Instead of using the exponent nt, we want you to use a for loop implementing product notation. For example instead of doing 2**5 we could use a for loop and do 2*2*2*2*2) Store all of the calculated values in a list, and make sure the function returns the list of values and not a single value.Hint: Remember that sum and product notation can be converted in to a `for` loop easily. The product notation for the equation would be:

$$p * \prod_{i=1}^{nt}(1+\frac{r}{n})$$

c. C = the future value of p with continuous compound interest using the formula given in the second equation under the **Background** section.

d. D = $100 \cdot \frac{|A - B|}{B}$, which is the *relative difference* between A and the *last value* of B expressed as a percentage.
To index the last value of B, use the syntax `B[-1]`

e. E = $100 \cdot \frac{|A - C|}{C}$, which is the *relative difference* between A and C expressed as a percentage.

2. Your name, MacID, student number, and the date are given in comments at the top of the first Python cell in the notebook.

3. Your answers to the design questions and test plan are given in the appropriate Markdown cells below.

4. Your program MUST have valid Python syntax and it must run without errors. Ensure that your program runs properly by running it before you submit.

5. You must sign out with a TA or IAI after you have submitted your lab at the submission station. Failure to do so could result in a mark of zero.

Design and Implementation Instructions

---

1. You can use this futval.py program as a starting point.

```
# futval.py
# A program to compute the value of an investment carried 10 years in to the future
print("This program calculates the future value of a 10 year investment")
def futval(principal, apr):
    value_list = []
    for i in range(10):
        principal = principal*(1+apr)
        value_list.append(principal)
```

1. Use the `abs` function to compute the absolute values D and E.
2. Use the `exp` function from the `math` library to compute B.

In [2]:
```
# DO NOT PUT AN INPUT STATEMENT IN THIS CELL
import math
#################---minor2(p, r, n, t) - (A: 2, B: 2, C: 2, D: 2, E: 2 Marks)---################
def minor2(p, r, n, t):
    #Place your code between here...

    A = p*(1+(r/n))**(n*t)     #change this value to your answer for A

    B = []
    total = p
    for i in range(1, (n*t)+1):
        total = total * (1+(r/n))
        B.append(total)
    #change this value to your answer for B

    C = p*math.exp(r*t)     #change this value to your answer for C

    D = 100*((A-B[-1])/B[-1])     #change this value to your answer for D

    E = 100*((abs(A-C)/C))     #change this value to your answer for E

    #...and here

    return A, B, C, D, E
#Change the values (p, r, n, t) here to test your code
result = minor2(100, 0.5, 4, 5)
print("A =", result[0])
print("B =", result[1])
print("C =", result[2])
print("D =", result[3])
print("E =", result[4])
```

```
A = 1054.50938424492
B = [112.5, 126.5625, 142.3828125, 160.1806640625, 180.2032470703125, 202.72865295410156, 228.06973457336426, 2
56.5784513950348, 288.65075781941414, 324.7321025468409, 365.323615365196, 410.9890672858455, 462.362700696576
2, 520.1580382836482, 585.1777930691043, 658.3250172027423, 740.615644353085, 833.1925998972207, 937.3416748843
733, 1054.50938424492]
C = 1218.2493960703473
D = 0.0
E = 13.440598645367368
```

In [3]:
```
###########---TESTING CELL---#############

#Run this cell to see your autograded mark

'''
Sample Input:
100, 0.5, 4, 5

Sample Output:
A = 1054.50938424492
B = [112.5, 126.5625, 142.3828125, 160.1806640625, 180.2032470703125, 202.72865295410156, 228.06973457336426, 256
C = 1218.2493960703473
D = 0.0
E = 13.440598645367368
'''

from autograder import autograde
checkMinor2Mark = autograde()
checkMinor2Mark.marker([minor2], "minor2")
```

```
Automarked Score =  10.0  /  10.0

(This mark is an estimate of your grade and may not accurately represent the mark you receive)
Use your testing plan to ensure your code is working as intended

BEFORE LEAVING THE LAB:
1. Restart your kernel and run all cells to check that your code is working as intended
        Kernel --> Restart & Run All
2. Make sure to answer the design questions and testing plan
3. Save your work
        File --> Save and Checkpoint
4. Submit your assignment
        On Jhub: Assignments --> Downloaded Assignments --> assign#_section# --> Submit
5. Sign out with your IAI or one of your TAs
```

Design Questions

1. Does your program enforce *n* and *t* to be integers? If so, why is this necessary? If not, is this a problem?

2. What is another way to calculate C? i.e without using the `exp` function.
3. Do you need to import `math`? If so, why? If not, why not?
4. Does the range of the `for` loop in B need to start at 1? If so, why? If not, why not?

Enter your answers into the Markdown cell below.

1. Yes, my program enforces n and t to be an integer. This is because n and t are set to be integer values so if we try to use a float variable instead, the program will give us an error. Also, the number of times the interest is compounded per year and the time expressed as a number of years cannot be expressed as a decimal, it must be defined as a whole number in real life which is why the program forces the n and t variables to be integers.

2. Another way to calculate C is using the left hand side of the second equation given in the background section, which is p*(1+(r/n))**(n*t), where the limit is when n approaches infinity.

3. Yes, I do need to import math. I had to import math in order to use the exponential of x function: e**x to find the value of C.

4. Yes, the range of the for loop in B does need to start at 1 because 1 is the smallest value in order to use the product notation function successfully. If I had used 0 instead of 1, the for loop would multiply any number by the starting value 0, which would return a value of 0 each time.

Testing Plan

Produce a test plan in the Markdown cell below, in the following form:

```
Test i
Input: [p, r, n, t]
Expected Output: [A, B (just the last value in the list), C, D, E]
Actual Output: [Aa, Ba (just the last value in the list), Ca, Da, Ea]
Result: Pass/Fail
```

where i is the test number (i.e Test 1, Test 2, etc), p, r, n, t are the four inputs for the program. A, B, C, D, E are the outputs the program is expected to produce (find these with your calculator). Aa, Ba, Ca, Da, Ea are the outputs the program actually produces.

Example:

```
Test 1
Input: [100, 0.5, 4, 5]
Expected Output: [1054.51, 1054.51, 1218.25, 0.0, 13.44]
Actual Output: [1054.51, 1054.51, 1218.25, 0.0, 13.44]
Result: Pass (Expected and actual output match! A Fail would be if the expected output differs from the actual output.)
```

Note: The actual output should be what the program produces, even if your output does not match the expected output.

You must have $NO\ LESS\ THAN\ 3\ TEST\ CASES$. Have at least 1 case where your program does not output an error. For the other cases, we encourage you to try and find test cases where your program would output an error (not mandatory, just recommended). That is, where the expected output is an error.

```
Test 1
Input: [100, 0.5, 4, 5]
Expected Output: [1054.509384, 1054.509384, 1218.249396, 0, 13.44059864]
Actual Output: [1054.50938424492, 1054.50938424492, 1218.2493960703473, 0.0, 13.440598645367368]
Result: Pass

Test 2
Input: [100, 0.5, 4, 5.5]
Expected Output: [1334.613439, 1334.613439, 1564.263188, 0, 14.68101728]
Actual Output: TypeError: 'float' object cannot be interpreted as an integer
Result: Fail

Test 3
Input: [100, -0.5, 4, 5]
Expected Output: [6.920875877, 6.920875877, 8.208499862, 0, 15.68647142]
Actual Output: [6.920875877393026, 6.920875877393026, 8.20849986238988, 0.0, 15.686471420881109]
Result: Pass
```

In [4]:
```
#Ignore this cell
```

In [5]:
```
#Ignore this cell
```

In [6]:
```
#Ignore this cell
```

In [7]:
```
#Ignore this cell
```

In [8]:
```
#Ignore this cell
```

In [9]:
```
#Ignore this cell
```

In [10]:
```
#Ignore this cell
```

In [11]:
```
#Ignore this cell
```

In [12]:
```
#Ignore this cell
```

In [13]:
```
#Ignore this cell
```