CBM-334

Complementos de matemáticas financieras para matemáticos

Nombre:

Daniel Alonso – 1066734

Profesor:

Alejandro Aceituno

Fecha:

18 de noviembre, 2018

Práctica #1

NOTA: TODAS LAS FUNCIONES UTILIZADAS ESTÁN ADJUNTAS EN EL ZIP

1. Calcular el valor capitalizado de 6000 euros durante 320 días al 2.5% anual para los casos nA = 360 y nA = 365
2. Calcular el valor capitalizado de 30000 euros ingresados el 1/02/2016 y retirados el 1/3/2016 al 2.5% anual con nA = 365. Repetir el cálculo para la misma inversión entre el 1/3/2016 y el 1/4/2016.

Algoritmo utilizado:

```python
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import datetime as dt
def int_simple(y, P, nd=365, na=365, F=1, inv=False, fi = (0,0,0), ff = (0,0,0)):
    print("y = Tasa de interés \nnd = numero de días de contrato \nna = número de días en el período de pag (ej: 360 ó 365 = 1 año) \nP = monto inicial")
    if y >= 1:
        y = y/100
    elif y < 0:
        print('La tasa de interés debe ser y > -1, y < 1')
    elif (nd <= 0 or na <= 0 or P <=0):
        print('nd, na o P deben ser todos mayores que 0')
    try:
        float(y), float(na), float(P), float(nd), float(F)
        (di,mi,ai) = fi; (df,mf,af) = ff
        datecond = (di > 31 or di < 0  or mi > 12 or mi < 0 or ai < 0 or df > 31 or df < 0 or mf > 12 or mf < 0 or af < 0 or fi > ff);
        fi = np.abs(fi); ff = np.abs(ff)
        int(di), int(mi), int(ai)
        if datecond == True:
            raise ValueError
    except (ValueError, TypeError):
        print('Los valores deben ser numéricos y en caso de fechas asegúrese que se encuentran bien introducidas')
        return
    datecond2 = (di > 0 and mi > 0 and ai > 0 and df > 0 and mf > 0 and af > 0)
    if datecond2 == False:
        if inv == False:
            F = P*(1 + y*(nd/na))
            return F
        elif inv == True:
            P = F*((1 + y*(nd/na))**(-1))
            return P
    if datecond2 == True:
        print('\n Fecha introducida: desde ({}/{}/{}) hasta ({}/{}/{}) \n'.format(fi[0],fi[1],fi[2],ff[0],ff[1],ff[2]))
        fi = dt.date(ai,mi,di); ff = dt.date(af,mf,df); nd = (ff-fi).days
        if inv == False:
            F = P*(1 + y*(nd/na))
            return F
        elif inv == True:
            P = F*((1 + y*(nd/na))**(-1))
            return P
    else:
        print('Error, inv no está correctamente introducido, asegúrese que sea True o False')
```

PARÁMETROS: y = tasa de interés, P = monto inicial, nd = días de contrato, na = número de días en el período, F = valor futuro (para inverso), inv = verdadero cuando es inverso o falso cuando es normal, fi = fecha inicial (solo se introduce cuando se trabaja con fechas), ff = fecha final

1. Para nA = 360 introducimos los siguientes parámetros en la función:

   int_simple(2.5, 6000, nd=320, na=360)

```
In [14]: int_simple(2.5,6000,nd=320,na=360)
y = Tasa de interés
nd = numero de días de contrato
na = número de días en el período de pag (ej: 360 ó 365 = 1 año)
P = monto inicial
Out[14]: 6133.333333333333
```

En **out[14]:** podemos ver el resultado, **6133.33333 euros**

Para nA = 365 introducimos los siguientes parámetros en la función:

int_simple(2.5, 6000, nd=320, na=365)

```
In [15]: int_simple(2.5,6000,nd=320,na=365)
y = Tasa de interés
nd = numero de días de contrato
na = número de días en el período de pag (ej: 360 ó 365 = 1 año)
P = monto inicial
Out[15]: 6131.506849315068
```

En out[15]: podemos ver el resultado, 6131.506849315068 euros

2. nA = 365 por default, así que introducimos los datos en la función de la siguiente manera:

    - Para 1/2/2016 – 1/3/2016

introducimos los siguientes parámetros en la función:
int_simple(2.5, 30000, fi=(1,2,2016), ff=(1,3,2016))

```
In [17]: int_simple(2.5, 30000, fi=(1,2,2016), ff=(1,3,2016))
y = Tasa de interés
nd = numero de días de contrato
na = número de días en el período de pag (ej: 360 ó 365 = 1 año)
P = monto inicial

 Fecha introducida: desde (1/2/2016) hasta (1/3/2016)

Out[17]: 30059.589041095893
```

en out[17]: podemos ver el resultado, 30059.589041095893 euros

    - Para 1/3/2016 – 1/4/2016

introducimos los siguientes parámetros en la función:
int_simple(2.5, 30000, fi=(1,3,2016), ff=(1,4,2016))

```
In [18]: int_simple(2.5, 30000, fi=(1,3,2016), ff=(1,4,2016))
y = Tasa de interés
nd = numero de días de contrato
na = número de días en el período de pag (ej: 360 ó 365 = 1 año)
P = monto inicial

 Fecha introducida: desde (1/3/2016) hasta (1/4/2016)

Out[18]: 30063.698630136987
```

en out[18]: podemos ver el resultado, 30059.589041095893 euros

3. Para una inversión de 6000 euros al 2.5%. Determinar el valor capitalizado al cabo de un año cuando los intereses se acumulan anual, semestral, trimestral, mensual, semanal o diariamente. Repetir el cálculo para una inversión a 10 años

Algoritmo utilizado:

```
def int_compuesto_general(y, P, n, m=1, F=1, inv=False, fi = (0,0,0), ff = (0,0,0)):
    print("y = Tasa de interés \nn = numero de años de contrato \nm = cantidad de rentas por año (ej: 4 = trimestral) \nP = monto inicial, \nF = valor futuro (en caso inverso)")
    if y >= 1 and y < 100:
        y = y/100
    elif y < 0:
        print('La tasa de interés debe ser y > -1, y < 100')
    elif (n <= 0 or P <=0):
        print('n y P deben ser todos mayores que 0')
    else:
        print('Error de introducción de datos.')
    try:
        float(y), float(n), float(P), float(m), float(F)
        (di,mi,ai) = fi; (df,mf,af) = ff
        datecond = (di > 31 or di < 0  or mi > 12 or mi < 0 or ai < 0 or df > 31 or df < 0 or mf > 12 or mf < 0 or af < 0 or fi > ff);
        fi = np.abs(fi); ff = np.abs(ff)
        int(di), int(mi), int(ai)
        if datecond == True:
            raise ValueError
    except (ValueError, TypeError):
        print('Los valores deben ser numéricos y en caso de fechas asegúrese que se encuentran bien introducidas')
        return
    datecond2 = (di > 0 and mi > 0 and ai > 0 and df > 0 and mf > 0 and af > 0)
    if datecond2 == False:
        if inv == False:
            F = P*(1 + y/m)**(n*m)
            return F
        elif inv == True:
            P = F*((1 + y/m)**(-n*m))
            return P
    if datecond2 == True:
        print('\n Fecha introducida: desde ({}/{}/{}) hasta ({}/{}/{}) \n'.format(fi[0],fi[1],fi[2],ff[0],ff[1],ff[2]))
        fi = dt.date(ai,mi,di); ff = dt.date(af,mf,df); n = (int((ff-fi).days/365))*m
        if inv == False:
            F = P*(1 + y/m)**(n)
            return F
        elif inv == True:
            P = F*((1 + y/m)**(-n))
            return P
```

PARÁMETROS: y = tasa de interés, P = monto inicial, n = duración del contrato en años, m = frecuencia de las capitalizaciones en 1 año, F = valor futuro (para inverso), inv = verdadero cuando es inverso o falso cuando es normal, fi = fecha inicial (solo se introduce cuando se trabaja con fechas), ff = fecha final

a. 6000 euros, tasa de 2.5% anual, duración de 1 año

m	Período de tiempo (capitalización)	Parámetros a introducir en la función
1	Anual	int_compuesto_general(2.5, 6000, 1, m=1)
2	Semestral	int_compuesto_general(2.5, 6000, 1, m=2)
4	Trimestral	int_compuesto_general(2.5, 6000, 1, m=4)
12	Mensual	int_compuesto_general(2.5, 6000, 1, m=12)
52	Semanal	int_compuesto_general(2.5, 6000, 1, m=52)
365	Diario	int_compuesto_general(2.5, 6000, 1, m=365)

Comando	Output	Respuesta
int_compuesto_general(2.5, 6000, 1, m=1)	Out[36]: 6149.999999999999	6149.9999 euros
int_compuesto_general(2.5, 6000, 1, m=2)	Out[37]: 6150.9375	6150.9375 euros
int_compuesto_general(2.5, 6000, 1, m=4)	Out[38]: 6151.412118530276	6151.41212 euros
int_compuesto_general(2.5, 6000, 1, m=12)	Out[39]: 6151.730741899741	6151.730142 euros
int_compuesto_general(2.5, 6000, 1, m=52)	Out[40]: 6151.853764605738	6151.853765 euros
int_compuesto_general(2.5, 6000, 1, m=365)	Out[41]: 6151.885456359545	6151.8854564 euros

b. 6000 euros, tasa de 2.5% anual, duración de 10 años

m	Período de tiempo (capitalización)	Parámetros a introducir en la función
1	Anual	int_compuesto_general(2.5, 6000, 10, m=1)
2	Semestral	int_compuesto_general(2.5, 6000, 10, m=2)
4	Trimestral	int_compuesto_general(2.5, 6000, 10, m=4)
12	Mensual	int_compuesto_general(2.5, 6000, 10, m=12)
52	Semanal	int_compuesto_general(2.5, 6000, 10, m=52)
365	Diario	int_compuesto_general(2.5, 6000, 10, m=365)

Comando	Output	Respuesta
int_compuesto_general(2.5, 6000, 10, m=1)	Out[42]: 7680.50726517814	6149.9999 euros
int_compuesto_general(2.5, 6000, 10, m=2)	Out[43]: 7692.223390251506	6150.9375 euros
int_compuesto_general(2.5, 6000, 10, m=4)	Out[44]: 7698.160923691209	6151.41212 euros
int_compuesto_general(2.5, 6000, 10, m=12)	Out[45]: 7702.1492530688165	6151.730142 euros
int_compuesto_general(2.5, 6000, 10, m=52)	Out[46]: 7703.689672436472	6151.853765 euros
int_compuesto_general(2.5, 6000, 10, m=365)	Out[47]: 7704.086543210197	6151.8854564 euros

4. Suponiendo que conoce la siguiente tabla de tipos aplicables para una inversión de 6000 euros a 10 años

Años	Tipo (%)
0 – 5	2.5
5 – 7	2.65
7 – 8	2.8
8 – 9	2.9
9 – 10	3.05

Determinar el valor capitalizado al final de la inversión

Algoritmo utilizado:

```python
def int_compuesto_variable(P, n=1, m=1, F=1, inv=False):
    i = 0; k = 1; tiempo = ['años', 'semestres', 'cuatrimestres', 'trimestres', 'meses (2)', 'meses', 'semanas', 'días', 'medios días', 'horas', 'minutos', 'segundos']
    val_m = [1,2,3,4,6,12,52,365,730,8760,525600,31536000]; dic_m = dict(zip(val_m,tiempo)); Tiempo = []; Acumulado = []; Tasa = []; interv = []; temp = []; tempsum = [0]; loopelement = 0; errcount = 0
    try:
        if m not in val_m:
            raise ValueError
        if inv == False:
            while i < n:
                try:
                    y = float(input('introduzca la tasa número {}: '.format(k)))
                    if y >= 1:
                        if y > 99.99:
                            raise ValueError
                        y = y/100
                    elif y <= -1:
                        if y < -99.99:
                            raise ValueError
                        y = y/100
                    s = int(input('cantidad de {} utilizando la tasa {}: '.format(dic_m[m],y)))
                    if (s > n or s > (n - i)) :
                        raise ValueError
                    F = F*((1 + y/m)**(s*m))
                    i += s; k += 1
                    Tasa.append(y); interv.append(s); Acumulado.append(F*P)
                except ValueError:
                    print('Error de introducción de datos, inténtelo de nuevo')
                    errcount +=1;
                    if errcount == 20:
                        return
            F = F*P
            for num in interv:
                if (loopelement == len(interv)):
                    break
                temp.append(num); tempsum.append(sum(temp))
                elem = str(tempsum[loopelement]) + ' - ' + str(tempsum[loopelement+1]) + ' {}'.format(dic_m[m])
                loopelement += 1; Tiempo.append(elem)
            print('\nValor presente: {} \nTasas introducidas: '.format(P))
            print(pd.DataFrame({'Tiempo': Tiempo, 'Tasa' : Tasa, 'Acumulado': Acumulado}))
            return F
        if inv == True:
            P = 0; K = 1
            while i < n:
                try:
                    y = float(input('introduzca la tasa número {}:'.format(k)))
                    if y >= 1:
                        if y > 99.99:
                            raise ValueError
                        y = y/100
                    elif y <= -1:
                        if y < -99.99:
                            raise ValueError
                        y = y/100
                    elif y > 99.99:
                        raise ValueError
                    elif y <= 0:
                        raise ValueError
                    s = int(input('cantidad de {} utilizando la tasa {}'.format(dic_m[m],y)))
                    if (s > n or s > (n - i)) :
                        raise ValueError
                    K = K*(1 + y/m)**(s)
                    i += s; k += 1
                    Tasa.append(y); interv.append(s);  Acumulado.append(F/K)
                except ValueError:
                    print('Error de introducción de datos, inténtelo de nuevo')
                    errcount +=1;
                    if errcount == 20:
                        return
            P = F/K
            for num in interv:
                if (loopelement == len(interv)):
                    break
                temp.append(num); tempsum.append(sum(temp))
                elem = str(tempsum[loopelement]) + ' - ' + str(tempsum[loopelement+1]) + ' {}'.format(dic_m[m])
                loopelement += 1; Tiempo.append(elem)
            print('\nValor presente: {} \n\nTasas introducidas: '.format(P))
            print(pd.DataFrame({'Tiempo': Tiempo, 'Tasa' : Tasa, 'Acumulado': Acumulado}))
            print('\n Resultado: ')
            return P
    except (ValueError, KeyError):
        print('Error en la introducción de datos, asegúrese que los datos de todas las variables son numéricos y mayores que 0. \nAsegúrese en caso de no haberse dado cuenta, que su cantidad de años en un periodo determinado sea menor que "n", \nya que la duración de un
        print(' \n¿Desea intentar de nuevo con los mismos datos introducidos?  (Y/N) ')
        while True:
            resp = input('(Y/N) : ')
            if resp == 'N' or resp == 'n':
                print('\nOperación abortada\n')
                return; break
            elif resp == 'S' or resp == 's' or resp == 'Y' or resp == 'y':
                print('\nReintroduciendo datos\n')
                int_compuesto_variable(P, n, m=m, F=F, inv=inv)
                break
            else:
                print('Error, intente de nuevo')
                pass
```

PARÁMETROS: P = monto inicial, n = duración del contrato en años, m = frecuencia de las capitalizaciones en 1 año, F = valor futuro (para inverso), inv = verdadero cuando es inverso o falso cuando es normal

El parámetro m (cantidad de capitalizaciones en 1 año) está establecido por default como m=1, entonces las capitalizaciones son anuales (1 vez al año).

Con la información que nos provee el ejercicio, ejecutamos la función con los siguientes parámetros:

int_compuesto_variable(6000,n=10)

Luego de la ejecución se nos presenta un input para introducir las tasas, en este caso, la primera tasa a utilizarse que es 2.5:

```
In [49]: int_compuesto_variable(6000,n=10)

introduzca la tasa número 1: 2.5
```

(todas las tasas a introducir en todas las funciones programadas para esta práctica reciben la tasa de interés en porcentaje como vemos en el recorte anterior o como decimal, en este caso sería 0.025)

Luego de introducir la primera tasa de interés nos pregunta la cantidad de [períodos] en los cuales se utilizará dicha tasa, en este caso, años.

```
In [52]: int_compuesto_variable(6000,n=10)

introduzca la tasa número 1: 2.5

cantidad de años utilizando la tasa 0.025: 5
```

Y completamos los datos que faltan:

```
In [70]: int_compuesto_variable(6000,n=10)

introduzca la tasa número 1: 2.5

cantidad de años utilizando la tasa 0.025: 5

introduzca la tasa número 2: 2.65

cantidad de años utilizando la tasa 0.0265: 2

introduzca la tasa número 3: 2.8

cantidad de años utilizando la tasa 0.027999999999999997: 1

introduzca la tasa número 4: 2.9

cantidad de años utilizando la tasa 0.028999999999999998: 1

introduzca la tasa número 5: 3.05

cantidad de años utilizando la tasa 0.0305: 1

Valor presente: 6000
Tasas introducidas:
        Tiempo    Tasa    Acumulado
0   0 - 5 años  0.0250  6788.449277
1   5 - 7 años  0.0265  7153.004278
2   7 - 8 años  0.0280  7353.288397
3   8 - 9 años  0.0290  7566.533761
4  9 - 10 años  0.0305  7797.313041
Out[70]: 7797.31304054725
```

Como podemos ver en **Out[70]:** nuestro resultado será de **7797.31304054725 euros**

5. Para una inversión de 6000 euros al 2.75% a 1 año, compara la capitalización obtenida en el caso simple, en los casos compuestos acumulando interés trimestral, mensual, semanal, y diariamente, y la capitalización continua. Calcula la capitalización compuesta acumulando intereses cada hora.
6. Obtén la capitalización continua para la inversión del ejercicio propuesto en el caso de capitalización compuesta.

Algoritmo utilizado:

```python
def int_continuo(y, P, n, F=1, inv=False, fi = (0,0,0), ff = (0,0,0)):
    print("y = Tasa de interés \nn = numero de años de contrato \nP = monto inicial, \nF = valor futuro (en caso inverso)")
    if y >= 1:
        y = y/100
    elif y < 0:
        print('La tasa de interés debe ser y > -1, y < 1')
    elif (n <= 0 or P <= 0):
        print('n y P deben ser todos mayores que 0')
    try:
        float(y), float(n), float(P), float(F)
        (di,mi,ai) = fi; (df,mf,af) = ff
        datecond = (di > 31 or di < 0  or mi > 12 or mi < 0 or ai < 0 or df > 31 or df < 0 or mf > 12 or mf < 0 or af < 0 or fi > ff);
        fi = np.abs(fi); ff = np.abs(ff)
        int(di), int(mi), int(ai)
        if datecond == True:
            raise ValueError
    except (ValueError, TypeError):
        print('Los valores deben ser numéricos y en caso de fechas asegúrese que se encuentran bien introducidas')
        return
    datecond2 = (di > 0 and mi > 0 and ai > 0 and df > 0 and mf > 0 and af > 0)
    if datecond2 == False:
        if inv == False:
            F = P*np.exp(y*n)
            return F
        elif inv == True:
            P = F*np.exp(-y*n)
            return P
    if datecond2 == True:
        print('\n Fecha introducida: desde ({}/{}/{}) hasta ({}/{}/{}) \n'.format(fi[0],fi[1],fi[2],ff[0],ff[1],ff[2]))
        fi = dt.date(ai,mi,di); ff = dt.date(af,mf,df); n = (ff-fi).days/365; y = y/365
        if inv == False:
            F = P*np.exp(y*n)
            return F
        elif inv == True:
            P = F*np.exp(-y*n)
            return P
    else:
        print('Error, inv no está correctamente introducido, asegúrese que sea True o False')
```

PARÁMETROS: P = monto inicial, n = duración del contrato en años, F = valor futuro (para inverso), inv = verdadero cuando es inverso o falso cuando es normal

5. Para obtener la información que se nos pide debemos introducir los parámetros de la función de la siguiente manera:

int_continuo(2.75, 6000, 1)

```
In [74]: int_continuo(2.75, 6000, 1)
y = Tasa de interés
n = numero de años de contrato
P = monto inicial,
F = valor futuro (en caso inverso)
Out[74]: 6167.289690643516
```

En **Out[74]:** podemos ver que el resultado es de **6167.2897 euros**

Claramente vemos que mientras más frecuente sea la capitalización, más se acerca al monto obtenido en interés continuo. En el ejercicio anterior vemos que el resultado obtenido al capitalizar diariamente está bastante cerca al de capitalización continua. Aún más en el próximo calculo que se pide en el cual la capitalización compuesta es por hora.

Como se pide en el ejercicio escribimos la función de la siguiente manera:

int_compuesto_general(2.75, 6000, 1, m=8760)

Ya que 8760 corresponde al número de horas en un año. Y obtenemos lo siguiente:

```
In [76]: int_compuesto_general(2.75,6000,1,m=8760)
y = Tasa de interés
n = numero de años de contrato
m = cantidad de rentas por año (ej: 4 = trimestral)
P = monto inicial,
F = valor futuro (en caso inverso)
Out[76]: 6167.289424435152
```

En **Out[76]:** podemos ver que la respuesta es casi prácticamente idéntica al resultado obtenido aplicando interés continuo, **6167.289424 euros**

6. Para obtener la información que se nos pide debemos introducir los parámetros de la función de la siguiente manera (6000 euros, 2.5% anual, 10 años) :

int_continuo(2.5, 6000, 10)

Obtenemos lo siguiente:

```
In [81]: int_continuo(2.5, 6000, 10)
y = Tasa de interés
n = numero de años de contrato
P = monto inicial,
F = valor futuro (en caso inverso)
Out[81]: 7704.152500126448
```

En **Out[81]:** podemos ver que el resultado es de **7704.152500126448 euros**

7. Calcula la TAE en % de una inversión al tipo nominal de 2.5% que paga intereses cada mes. Repite el calculo si los intereses son anuales, semestrales, trimestrales, semanales, diarios o continuos.
8. Para los tipos nominales de 2%, 2.5%, 3% y 5% aplicados a una capitalización continua, determina el % de TAE
9. Compara en términos de TAE, una inversión que paga intereses del 2.1% anual con una que paga el 2% semanal.
10. Elabora un programa en PYTHON que proporcione la TAE de una inversión en el caso de capitalización continua y compuesta.

Algoritmo utilizado (10):

```python
def TAE(y, n, m=1, tae=1, cont=False, inv=False, fi = (0,0,0), ff = (0,0,0)):
    print("y = Tasa de interés \nn = numero de años de contrato \nm = cantidad de rentas por año (ej: 4 = trimestral) \nP = monto inicial, \nF = valor futuro
    if y >= 1 and y < 100:
        y = y/100
    elif y < 0:
        print('La tasa de interés debe ser y > -1, y < 100')
    elif (n <= 0 or m <= 0):
        print('n y m deben ser todos mayores que 0')
    else:
        print('Error de introducción de datos.')
    try:
        float(y), float(tae), float(m)
        (di,mi,ai) = fi; (df,mf,af) = ff
        datecond = (di > 31 or di < 0  or mi > 12 or mi < 0 or ai < 0 or df > 31 or df < 0 or mf > 12 or mf < 0 or af < 0 or fi > ff);
        fi = np.abs(fi); ff = np.abs(ff)
        int(di), int(mi), int(ai)
        if datecond == True:
            raise ValueError
    except (ValueError, TypeError):
        print('Los valores deben ser numéricos y en caso de fechas asegúrese que se encuentran bien introducidas')
        return
    datecond2 = (di > 0 and mi > 0 and ai > 0 and df > 0 and mf > 0 and af > 0)
    if datecond2 == False:
        if cont == False:
            if inv == False:
                tae = (1 + y/m)**(n*m) -1
                return tae
            elif inv == True:
                taeinv = (tae+1)*((1 + y/m)**(-n*m))
                return taeinv
        elif cont== True:
            if inv == False:
                tae = np.exp(y*n) - 1
                return tae
            elif inv == True:
                taeinv = (tae+1)*np.exp(-n*y)
                return taeinv
    if datecond2 == True:
        print('\n Fecha introducida: desde ({}/{}/{}) hasta ({}/{}/{}) \n'.format(fi[0],fi[1],fi[2],ff[0],ff[1],ff[2]))
        fi = dt.date(ai,mi,di); ff = dt.date(af,mf,df); n = (int((ff-fi).days/365))*m
        if cont == False:
            if inv == False:
                tae = (1 + y/m)**(n) - 1
                return tae
            elif inv == True:
                taeinv = (tae+1)*((1 + y/m)**(-n))
                return taeinv
        elif cont == True:
            if inv == False:
                tae = np.exp(n*y) - 1
                return tae
            elif inv == True:
                taeinv = (tae+1)*(np.exp(-n*y))
                return taeinv
    else:
        print('Error, inv no está correctamente introducido, asegúrese que sea True o False')
```

PARÁMETROS: y = tasa de interés, n = duración del contrato en años, cont = verdadero o falso dependiendo de que se desee la TAE con capitalización continua, fi y ff = fecha inicial y final de la inversión (en caso de ser con fecha

7. Tabla de comandos con su resultado

m	Período de tiempo (capitalización)	Parámetros a introducir en la función
1	Anual	TAE(2.5, 1, m=1)
2	Semestral	TAE(2.5, 1, m=2)
4	Trimestral	TAE(2.5, 1, m=4)
12	Mensual	TAE(2.5, 1, m=12)
52	Semanal	TAE(2.5, 1, m=52)
365	Diario	TAE(2.5, 1, m=365)
-	continuo	TAE(2.5, 1, cont=True)

Comando	Respuesta
TAE(2.5, 1, m=1)	0.02499999999999991
TAE(2.5, 1, m=2)	0.02515624999999999
TAE(2.5, 1, m=4)	0.02523535308837932
TAE(2.5, 1, m=12)	0.025288456983290075
TAE(2.5, 1, m=52)	0.025308960767623123
TAE(2.5, 1, m=365)	0.025314242726590885
TAE(2.5, 1, cont=True)	0.025315120524428858

8. Tabla de comandos con su resultado

y	Parámetros a introducir en la función	Respuesta
2	TAE(2, 1, cont=True)	0.020201340026755776
2.5	TAE(2.5, 1, cont=True)	0.025315120524428858
3	TAE(3, 1, cont=True)	0.030454533953516938
5	TAE(5, 1, cont=True)	0.05127109637602412

9. Tabla de comandos con su resultado

Tipo	Parámetros a introducir en la función	Respuesta
2% anual	TAE(2, 1, m=1)	0.020000000000000018
2.1% semanal	TAE(2.1, 1, m=52)	0.021217722437950615

11. Calcula el valor actualizado de 6000 euros a recibir dentro de 6 años para una tasa de interés constante de 3% anual en el caso simple, compuesto mensual y continuo.
12. En los casos del ejercicio anterior calcula el valor actualizado, simple y con composición semestral de 6000 euros a recibir dentro de 10 años, sabiendo que la tasa de intereses viene dada por la tabla:

Años	Tipo (%)
0 – 5	2.5
5 – 7	2.65
7 – 8	2.8
8 – 9	2.9
9 – 10	3.05

Los algoritmos presentados anteriormente pueden calcular valores actualizados con un valor booleano llamado '**inv**' que por default se encuentra '**inv = False**', sin embargo el usuario puede escribir '**inv = True**' para utilizar esta funcionalidad. En dado caso, debe darle un valor a **F.**

11. Utilizando las funciones anteriores, podemos calcular actualización de la siguiente manera:
Para interés simple introducimos:

int_simple(3, 1, nd=365*6, na=365, F=6000, inv=True)

Explicación de cada parámetro:

- **3** es la tasa (3%)
- **1** es un parámetro default, ponga 1 e ignórelo
- nd es la cantidad de días en 6 años que se calcularía **365*6**, así que establecemos el parámetro nd = 365*6
- na son los días en un año (**365**)
- F es el valor futuro, **F=6000**
- inv es estableciendo que es un cálculo de actualización, **inv=True**

Input y output:

```
In [21]: int_simple(3, 1, nd=365*6, na=365, F=6000, inv=True)
y = Tasa de interés
nd = numero de días de contrato
na = número de días en el período de pag (ej: 360 ó 365 = 1 año)
P = monto inicial
Out[21]: 5084.745762711865
```

Vemos que el resultado es **5084.7458 euros**

Para interés compuesto mensual introducimos:

int_compuesto_general(3, 1, 6, m=12, F=6000, inv=True)

- **3** es la tasa (3%)
- **1** es un parámetro default, ponga 1 e ignórelo
- **6** es la duración total (6 años)
- **m=12** indica que es mensual (cantidad de cap. por año)
- F es el valor futuro, **F=6000**
- inv es estableciendo que es un cálculo de actualización, **inv=True**

Input y output:

```
In [22]: int_compuesto_general(3, 1, 6, m=12, F=6000, inv=True)
y = Tasa de interés
n = numero de años de contrato
m = cantidad de rentas por año (ej: 4 = trimestral)
P = monto inicial,
F = valor futuro (en caso inverso)
Out[22]: 5012.747133856262
```

Vemos que el resultado **5012.747134 euros**

Para interés continuo introducimos lo siguiente:

int_continuo(3, 1, 6, F=6000, inv=True)

- **3** es la tasa (3%)
- **1** es un parámetro default, ponga 1 e ignórelo
- **6** es la duración total (6 años)
- F es el valor futuro, **F=6000**
- inv es estableciendo que es un cálculo de actualización, **inv=True**

Input y output:

```
In [23]: int_continuo(3, 1, 6, F=6000, inv=True)
y = Tasa de interés
n = numero de años de contrato
P = monto inicial,
F = valor futuro (en caso inverso)
Out[23]: 5011.621268467632
```

Vemos que el resultado es **5011.6212685 euros**

12. Introduciendo los parámetros siguientes en la segunda función:

int_compuesto_variable(1, n=20, m=2, F=6000, inv=True)

- 1 es un parámetro default, ponga 1 e ignórelo
- 20 es la duración total (10 años = 20 semestres)
- m es la cantidad de capitalizaciones por año (2) , **m=2**
- inv es estableciendo que es un cálculo de actualización, **inv=True**

y obtenemos lo siguiente:

```
In [27]: int_compuesto_variable(1, n=20, m=2, F=6000, inv=True)

introduzca la tasa número 1: 
```

Donde introducimos la tasa número 1 de la tabla **2.5%**

```
In [27]: int_compuesto_variable(1, n=20, m=2, F=6000, inv=True)

introduzca la tasa número 1: 2.5

cantidad de semestres utilizando la tasa 0.025: 
```

Y luego introducimos la cantidad de semestres, 5 años = **10 semestres**

```
In [27]: int_compuesto_variable(1, n=20, m=2, F=6000, inv=True)

introduzca la tasa número 1: 2.5

cantidad de semestres utilizando la tasa 0.025: 10

introduzca la tasa número 2: 
```

Y así sucesivamente hasta terminar:

```
In [26]: int_compuesto_variable(1, n=20, m=2, F=6000, inv=True)

introduzca la tasa número 1: 2.5

cantidad de semestres utilizando la tasa 0.025: 10

introduzca la tasa número 2: 2.65

cantidad de semestres utilizando la tasa 0.0265: 4

introduzca la tasa número 3: 2.8

cantidad de semestres utilizando la tasa 0.027999999999999997: 2

introduzca la tasa número 4: 2.9

cantidad de semestres utilizando la tasa 0.028999999999999998: 2

introduzca la tasa número 5: 3.05

cantidad de semestres utilizando la tasa 0.0305: 2
```

Donde obtenemos el siguiente resultado:

```
Tasas introducidas:
              Tiempo    Tasa    Acumulado
0   0 - 10 semestres  0.0250  5299.085557
1  10 - 14 semestres  0.0265  5027.296292
2  14 - 16 semestres  0.0280  4889.433816
3  16 - 18 semestres  0.0290  4750.665684
4  18 - 20 semestres  0.0305  4609.018728

 Resultado:
Out[26]: 4609.018727902981
```

Podemos ver en **Out[26]:** la respuesta, serán **4609.018727902981 euros**

13. Calcula el valor de un contrato de futuros a 24 meses y un precio pactado de 23 euros sobre un activo, cuyo precio actual es de 30 euros y siendo el tipo de interés anual del 5% para ese vencimiento.
14. Repite el cálculo para el caso en que el activo pague 2 euros al cabo de seis, doce, dieciocho y veinticuatro meses, con tipos de 2, 3, 4 y 5 por cuento a los meses antes indicados.
15. Repite el cálculo para el caso en que el activo pague una tasa de dividendo del 2%.

Algoritmo utilizado:

```
def val_futuros(r, S_t, t, T, K, cant_r, R=0, renta=False):
    suma = 0;
    if r >= 1:
        r = r/100
    elif r < 0:
        print('La tasa de interés debe ser r > -1, r < 1')
    elif (T < 0 or t < 0 or K <= 0):
        print('Todos los valores (excepto t) deben ser todos mayores que 0')
    try:
        float(r), float(t), float(T), float(K), float(S_t)
    except (ValueError, TypeError):
        print('Los valores deben ser numéricos y en caso de fechas asegúrese que se encuentran bien introducidas')
        return
    if renta == False:
        Ft = S_t - K*np.exp(r*(t-T))
        return Ft
    if renta == True:
        k = 1
        while k <= cant_r:
            s = float(input('Tasa número {}: '.format(k))); s = s/100
            m = float(input('Tiempo con {}%: '.format(s*100))); m=m/12
            suma = suma + np.exp(-s*m); k += 1
        I_t = R*(suma); print()
        Ft = S_t - I_t - K*np.exp(r*(t-T))
        print(K*np.exp(r*(t-T)))
        return Ft
```

13. Fórmula utilizada:

$$F_T(t) = S(t) - K\, e^{-r(t,T)\,(T-t)}$$

Parámetros introducidos para el ejercicio a continuación:

val_futuros(0.05, 30, 0, 2, 23, 0)

Ya que:

S(t) = 30 euros
r = 5%
T = 2
t = 0
K = 23 euros

Y obtenemos el siguiente output:

```
Out[5]: 9.18873938517293
```

Lo que indica que el valor del contrato es de aprox. 9.18874 euros

14. Utilizando la siguiente fórmula:

$$F_T(t) = S(t) - I(t) - Ke^{-r(T-t)}$$

$$I(t) = 2(e^{-0.02/2} + e^{-0.03} + e^{-0.04\times 3/2} + e^{-0.05\times 2})$$

Introducimos los siguientes parámetros en la función:

val_futuros(0.05, 30, 2, 0, 23, 4, R=2, renta=True)

Ya que:

S(t) = 30 euros
r = 5%
T = 2
t = 0
K = 23 euros
Renta = 2 euros

De la siguiente manera:

```
In [31]: val_futuros(0.05, 30, 2, 0, 23, 4, R=2, renta=True)

Tasa número 1: 2

Tiempo con 2.0%: 6

Tasa número 2: 3

Tiempo con 3.0%: 12

Tasa número 3: 4

Tiempo con 4.0%: 18

Tasa número 4: 5

Tiempo con 5.0%: 24
```

Y obtenemos el siguiente output:

```
Out[23]: 1.5747393851729292
```

Esto quiere decir que el valor es de **1.5747393851729292 euros**

15. Utilizando la siguiente fórmula:

$$F_T(t) = S(t)e^{-d(T-t)} - Ke^{-r(T-t)}$$

Introducimos los parámetros de la siguiente manera:

val_futuros(0.05, 30, 0, 2, 23, d=0.02)

Ya que:

S(t) = 30 euros
r = 5%
T = 2
t = 0
K = 23 euros
d = 0.02

y obtenemos la siguiente respuesta:

```
Out[41]: 8.012422559742625
```

Esto quiere decir que el valor es de **8.012423 euros**

16. Calcular el precio de emisión de un bono a 10 años con valor principal de 1000 euros, que paga cupones anuales de 6 euros, suponiendo que el tipo de interés con vencimientos de 1 a 5 años es del 3% y a cada año adicional hasta 10 se incrementa un 0.25%. Haz el cálculo suponiendo que los tipos corresponden a capitalización simple, compuesta y continua

Precio de emisión de un bono para:

Capitalización simple:

$$B = \sum_{i=1}^{m} \frac{C_i}{1 + y_i t_i} + \frac{P}{1 + y_m t_m}$$

Calculado en código:

```
C = 6; y = 0.03
P = 1000; d = 0.0025
B = 0; n = 10; h = []
for k in range(1,11):
    if k >= 6:
        y += d
        B += C/(1+(y)*k)
        h.append(C/(1+(y)*k))
        continue
    B += C/(1+y*k)
    h.append(C/(1+y*k))
print(B+(P/(1+y*n)))
print(h)
```

Elementos de la sumatoria:

5.825242718446602 + 5.660377358490566 + 5.504587155963303 + 5.357142857142857 + 5.217391304347826 + 5.02092050209205 + 4.8192771084337345 + 4.615384615384615 + 4.411764705882352 + 4.2105263157894735

Resultado:

B = 752.397 Euros

Capitalización Compuesta:

$$B = \sum_{i=1}^{m} \frac{C_i}{(1+y_i)^{t_i}} + \frac{P}{(1+y_m)^{t_m}}$$

Calculado en código:

```
C = 6; y = 0.03
P = 1000; d = 0.0025
B = 0; n = 10; h = []
for k in range(1,11):
    if k >= 6:
        y += d
        B += C/((1+y)**k)
        h.append(C/((1+y)**k))
        continue
    B += C/((1+y)**k)
    h.append(C/((1+y)**k))
print(B+(P/((1+y)**k)))
print(h)
```

Elementos de la sumatoria:

5.825242718446602 + 5.655575454802526 + 5.490849956118957 + 5.330922287494133 + 5.175652706304984 + 4.952344962033029 + 4.715945764102916 + 4.4693710062835965 + 4.215520413472984 + 3.9572238139192693

Resultado:

B = 709.3259514028572 Euros

Capitalización Continua:

$$B = \sum_{i=1}^{m} C_i e^{-y_i t_i} + P e^{-y_m t_m}$$

Calculado en código:

```
C = 6; y = 0.03
P = 1000; d = 0.0025
B = 0; n = 10; h = []
for k in range(1,11):
    if k >= 6:
        y += d
        B += C/((1+y)**k)
        h.append(C/((1+y)**k))
        continue
    B += C/((1+y)**k)
    h.append(C/((1+y)**k))
print(B+(P/((1+y)**k)))
print(h)
```

Elementos de la sumatoria:

5.822673201291049 + 5.650587201505492 + 5.483587111627369 + 5.321522620302945 + 5.164247858550347 + 4.9370079483361105 + 4.696227229451209 + 4.4449093240903075 + 4.186057956426186 + 3.922618710779083

Resultado:

B = 703.3992242922072 Euros

17 y 18:

- Calcular la expresión de duración y la convexidad de un bono en el caso de capitalización compuesta y continua
- Calcular la expresión de duración y la convexidad de un bonoa 10 años con valor principal de 1000 euros, que paga cupones anuales de 6 euros, suponiendo que el tipo de interés con vencimientos de 1 a 5 años es del 3% y a cada año adicional hasta 10 se incrementa un 0.25%. Haz el cálculo suponiendo que los tipos corresponden a capitalización compuesta y capitalización continua

17.

En el caso de capitalización compuesta:

La fórmula de la duración de un bono se calcula en base a la TIR y es la siguiente:

$$\text{Duración} = -\frac{(1+y)}{B}\frac{dB}{dy} \quad \text{para} \quad B = \sum_{i=1}^{m} \frac{C_i}{(1+y)^{t_i}} + \frac{P}{(1+y_m)^{t_m}}$$

La derivada de B con respecto a la tasa de interés y:

$$\frac{dB}{dy} = -\left(\sum_{i=1}^{m} t_i \frac{C_i}{(1+y)^{t_i+1}} + t_m \frac{P}{(1+y)^{t_m+1}}\right)$$

Y tenemos la duración:

$$-\frac{(1+y)}{B}\left(-\left(\sum_{i=1}^{m} t_i \frac{C_i}{(1+y)^{t_i+1}} + t_m \frac{P}{(1+y)^{t_m+1}}\right)\right)$$

$$\frac{(1+y)}{\sum_{i=1}^{m} \frac{C_i}{(1+y)^{t_i}} + \frac{P}{(1+y_m)^{t_m}}}\left(\sum_{i=1}^{m} t_i \frac{C_i}{(1+y)^{t_i+1}} + t_m \frac{P}{(1+y)^{t_m+1}}\right)$$

$$\frac{\sum_{i=1}^{m} t_i \frac{C_i}{(1+y)^{t_i}} + t_m \frac{P}{(1+y)^{t_m}}}{\sum_{i=1}^{m} \frac{C_i}{(1+y)^{t_i}} + \frac{P}{(1+y_m)^{t_m}}}$$

La fórmula de convexidad:

$$\text{Convexidad} = \frac{1}{B}\frac{d^2B}{dy^2}$$

Calculamos la segunda derivada:

$$\frac{d^2B}{dy^2} = \frac{d}{dy}\left[-\left(\sum_{i=1}^{m} t_i \frac{C_i}{(1+y)^{t_i+1}} + t_m \frac{P}{(1+y)^{t_m+1}}\right)\right]$$

$$\frac{d^2B}{dy^2} = \sum_{i=1}^{m}\left[t_i(t_i+1)\frac{C_i}{(1+y)^{t_i+2}}\right] + t_m(t_m+1)\frac{P}{(1+y)^{t_m+2}}$$

Por lo que la convexidad de B es:

$$\frac{1}{\sum_{i=1}^{m}\frac{C_i}{(1+y)^{t_i}} + \frac{P}{(1+y_m)^{t_m}}}\left(\sum_{i=1}^{m}\left[t_i(t_i+1)\frac{C_i}{(1+y)^{t_i+2}}\right] + t_m(t_m+1)\frac{P}{(1+y)^{t_m+2}}\right)$$

O también:

$$\frac{1}{B}\left(\sum_{i=1}^{m}\left[\frac{t_i(t_i+1)}{(1+y)^2}\frac{C_i}{(1+y)^{t_i}}\right] + \frac{t_m(t_m+1)}{(1+y)^2}\frac{P}{(1+y)^{t_m}}\right)$$

En el caso de capitalización continua:

$$B = \sum_{i=1}^{m} C_i e^{-yt_i} + Pe^{-yt_m}$$

La derivada en B para la duración:

$$\frac{dB}{dy} = -\left(\sum_{i=1}^{m} C_i t_i e^{-yt_i} + Pt_m e^{-yt_m}\right)$$

Y la duración es:

$$-\frac{(1+y)}{B}\frac{dB}{dy} = \frac{(1+y)}{\sum_{i=1}^{m} C_i e^{-yt_i} + Pe^{-yt_m}}\left(\sum_{i=1}^{m} C_i t_i e^{-yt_i} + Pt_m e^{-yt_m}\right)$$

La segunda derivada de B es:

$$\frac{d^2B}{dy^2} = \sum_{i=1}^{m} C_i t_i^2 e^{-yt_i} + Pt_m^2 e^{-yt_m}$$

Y la convexidad es:

$$\frac{1}{B}\frac{d^2B}{dy^2} = \frac{1}{\sum_{i=1}^{m} C_i e^{-yt_i} + Pe^{-yt_m}} \left(\sum_{i=1}^{m} C_i t_i^2 e^{-yt_i} + Pt_m^2 e^{-yt_m} \right)$$

18.

Para capitalización compuesta:

Habiendo obtenido B en el ejercicio anterior:

B = 709.3259514028572 Euros

Utilizando Wolfram Alpha, obtenemos la TIR del bono:

y = 0.0422372

Entonces sustituyendo en nuestra fórmula:

$$\frac{\sum_{i=1}^{m} t_i \frac{C_i}{(1+y)^{t_i}} + t_m \frac{P}{(1+y)^{t_m}}}{B}$$

Calculado con el código siguiente:

```
C = 6; y = 0.0422372
P = 1000; Dur = 0;
n = 10; B = 709.3259514028572

for k in range(1,n+1):
    Dur += (k*(C/((1+y)**k)))
print((Dur+(n*(P/((1+y)**k))))/B)
```

Y obtenemos:

Duración = 9.671641898333466

Para la convexidad de manera similar, sustituimos en la siguiente fórmula:

$$\frac{1}{B}\left(\sum_{i=1}^{m}\left[\frac{t_i(t_i+1)}{(1+y)^2}\frac{C_i}{(1+y)^{t_i}}\right]+\frac{t_m(t_m+1)}{(1+y)^2}\frac{P}{(1+y)^{t_m}}\right)$$

Calculado con el código siguiente:

```
C = 6; y = 0.0422372
P = 1000; Conv = 0;
n = 10

for k in range(1,n+1):
    Conv += (k*(k+1)*(C/((1+y)**(k+2))))
print((Conv+(n*(k+1)*(P/((1+y)**(k+2)))))/B)
```

Y obtenemos:

Convexidad = 96.89084788743776

Para capitalización continua:

Utilizando el valor de B calculado en el ejercicio anterior:

B = 703.3992242922072 Euros

La TIR es:

y = 0.0422377

Utilizando las fórmulas anteriormente presentadas, y el algoritmo siguiente:

```
C = 6; y = 0.0422377
P = 1000; Dur = 0; Conv = 0;
n = 10; B = 703.3992242922072;
from numpy import exp

for k in range(1,n+1):
    Dur += (k*C*exp(-y*k))
print(((1+y)/B)*(Dur+(k*(P*exp(-y*k)))))

for k in range(1,n+1):
    Conv += ((k**2)*C*exp(-y*k))
print((1/B)*(Conv+((P*(k**2)*exp(-y*k)))))
```

Obtenemos para la duración y convexidad:

Duración = 10.078251075277544

Convexidad = 95.55381235461157