TRABAJO FIN DE GRADO. ANEXOS

GRADO EN ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS

CURSO ACADÉMICO 2020/2021

CONVOCATORIA JUNIO

TÍTULO: APLICACIONES FINANCIERAS PROGRAMADAS EN PYTHON

APELLIDOS/NOMBRE ESTUDIANTE: GARRIDO RODRÍGUEZ, REBECA

DNI: 53460043Q

GRADO/DOBLE GRADO QUE CURSA: GRADO EN ADMINISTRACIÓN Y DIRECCIÓN EMPRESAS (SEMIPRESENCIAL)

APELLIDOS/NOMBRE TUTOR:

Aparicio, Adolfo

Fecha:27 de mayo de 2021

Contenido

ANEXOS....3
Interés Simple....3
Interés Compuesto....3
Uso de VF, VA, Tasa y Nper....4
Rentas....7
Depósitos....8
VAN & TIR....8
Letras de cambio, letras del tesoro y bonos....12
Préstamos....13
Beta de compañías....14
Automatización de informes....14
1. Características del Ibex35....17

ANEXOS

Para elaborar este trabajo ha sido necesario la elaboración de diversos códigos que nos permitieran crear paso a paso las funciones necesarias, si bien se han incluido en la parte del trabajo las que se han considerado más importantes que apoyaran las argumentaciones del trabajo, son muchos códigos los que no se han podido incluir pero que pueden resultar interesantes si presenta más inquietud por conocer el código empleado. Se han clasificado en función del capítulo que ocupan para mantener el mismo orden.

Interés Simple

Código para gráfica en interés simple

```
fig, ax = plt.subplots()
ax.bar(tiempo, intereses, color = "darkblue")
plt.title("Intereses en I. Simple", fontsize = 20)
plt.xlabel("Periodos", fontsize = 15)
plt.ylabel("Intereses en €", fontsize = 15)
plt.xticks(np.arange(tiempo[0], tiempo[-1]+1, 1))
plt.yticks(np.arange(0,intereses[-1]+1,100))
plt.show()
```

Interés Compuesto

Código de calculadoras financieras para interés compuesto:

```
def CalculaFinanciera():

    mes = ['m', 'mes', 'mensual', 'uno']
    año = ['a', 'anual' , 'año']
    trimestral = ['t' , 'trimestral', 'trimestre', 'tres']
    semestral = ['s', 'semestre', 'semestral', 'seis']
    try:
        tipo = input('''Indique que tipo de periodo:
                        m = mensual
                        a = anual
                        t = trimestral
                        s = semestral ''').lower()
        capitalInicial = float(input('Importe a invertir: '))
        tantoNominal = float(input('Introduzca el tipo de interés Nominal anual (Ejemplo 8%: 0.08) : '))
        if tipo in mes:
                periodoTemporal = 'Meses'
                frecuencia = 12
        elif tipo in año:
                periodoTemporal = 'Años'
                frecuencia = 1
        elif tipo in trimestral:
                periodoTemporal = 'Trimestres'
                frecuencia = 4
        elif tipo in semestral:
                periodoTemporal = 'Semestres'
                frecuencia = 2
        periodos = float(input('Número de periodos (en {}): '.format(periodoTemporal)))
        vf = capitalInicial *(1 + tantoNominal)**(periodos / frecuencia)
        print("Su inversión tendrá un valor futuro de: {}".format(round(vf,2)))
    except ValueError:
        print('Error!, solo se admiten numeros')




CalculaFinanciera()
```

```python
def calculo_compuesta(C0, i, n):
    Cn=C0*(1+i)**n
    return round(Cn,2)

print("""          El tipo de interés (i) y el número de periodos (n) que introduzca deben tener la misma unidad temporal
          Si el tipo de interés es anual, entonces el número de periodos han de ser años.
          Si se trabaja con meses, entonces el tipo de interés introducido ha de ser un tipo mensual.

          Por ejmplo, si el tipo de interés es del 12% anual y la operación de inversión a interés simple
          dura 18 meses, indicaremos que el tipo de interés es i=0,12 y el número de años es n=1,5.
          """)

C0 = float(input('Importe a invertir: '))
i = float(input('Introduzca el tipo de interés (Ejemplo 8%: 0.08) : '))
n = float(input('Número de periodos: '))

print('')
print('                          El capital final obtenido es', calculo_simple(C0, i, n))
```

```
          El tipo de interés (i) y el número de periodos (n) que introduzca deben tener la misma unidad temporal.
          Si el tipo de interés es anual, entonces el número de periodos han de ser años.
          Si se trabaja con meses, entonces el tipo de interés introducido ha de ser un tipo mensual.

          Por ejmplo, si el tipo de interés es del 12% anual y la operación de inversión a interés simple
          dura 18 meses, indicaremos que el tipo de interés es i=0,12 y el número de años es n=1,5.

Importe a invertir: 10000
Introduzca el tipo de interés (Ejemplo 8%: 0.08) : 0.10
Número de periodos: 4

                          El capital final obtenido es 14641.0
```

Código para gráfica en interés compuesto

```python
fig, ax = plt.subplots()
ax.bar(tiempo, intereses, color = "Darkblue")
plt.title("Intereses en I. Compuesto", fontsize = 20)
plt.xlabel("Periodos", fontsize = 15)
plt.ylabel("Intereses en €", fontsize = 15)
plt.xticks(np.arange(tiempo[0], tiempo[-1]+1, 1))
plt.yticks(np.arange(0,intereses[-1],100))
plt.show()
```

Código para gráfica comparativa interés simple vs compuesto

```python
fig, ax = plt.subplots()
plt.plot(intcomp, color = "blue", marker = '*', markersize = 4, label = "Intereses Compuestos")
plt.plot(intsim, color = "orange", marker = '*', markersize = 4, label = "Intereses Simples")
plt.title("Comparación Intereses Simples vs Compuestos", fontsize = 20)
plt.xlabel("Periodos", fontsize = 15)
plt.ylabel("Intereses en €", fontsize = 15)
plt.xticks(np.arange(tiempo[0], tiempo[-1], 1))
plt.yticks(np.arange(intcomp[0], intcomp[-1], 100))
plt.legend(loc = "best")
plt.show()
```

Uso de VF, VA, Tasa y Nper

Fórmula VF

Capital Final

```python
i = float(input("Interes: "))
n = float(input("Plazo: "))
C0 = float(input("Capital inicial: "))

fv = npf.fv(rate=i, nper=n, pmt= 0,pv=-C0, when='end')
print("El capital final en esta operación es {}".format(round(fv,2)))
```

```
Interes: 0.10
Plazo: 4
Capital inicial: 10000
El capital final en esta operación es 14641.0
```

```python
c0 = float(input("Capital inicial: "))
n = float(input("Plazo: "))
Cn = float(input("Capital final: "))

i = ((Cn/c0)**(1/n))-1
print("El interés de esta operación es {} %".format(round(i,2)*100))
```

```
Capital inicial: 10000
Plazo: 4
Capital final: 14641
El interés de esta operación es 10.0 %
```

Fórmula Tasa

Interés

Equivalencia con fórmua Excel **TASA** (Español) **RATE** (Inglés)

Funcionan igual, requiere los mismos parámetros (plazo, pago, capital inicial, capital final y al final o principio del periodo).

```
Cn = float(input("Capital final: "))
n = float(input("Plazo: "))
C0 = float(input("Capital inicial: "))

rate = npf.rate(nper=n, pmt=0, pv=C0, fv=-Cn, when='end')
print("El tipo de interés de esta operación es {}%".format(round(rate*100,2)))

```

```
Capital final: 14641
Plazo: 4
Capital inicial: 10000
El tipo de interés de esta operación es 10.0 %
```

```
c0 = float(input("Capital inicial: "))
i = float(input("Interes: "))
n = float(input("Plazo: "))

Cn = c0*(1+i)**n

print("El Valor Final es {}".format(round(Cn,2)))
```

```
Capital inicial: 10000
Interes: 0.10
Plazo: 4
El Valor Final es 14641.0
```

Fórmual VA

Capital Inicial

Equivalencia con fórmula Excel **VA** (Español) **PV** (Inglés)

Igualmente nos solicita exactamente los mismos parámetros (tasa, plazo, pago, capital final, final o principio del periodo)

```
Cn = float(input("Capital final: "))
n = float(input("Plazo: "))
i = float(input("Interés: "))

C0 = npf.pv(rate=i, nper=n, pmt=0, fv=-Cn, when='end')
print("El Capital Inicial de esta operación es {}".format(round(C0,2)))
```

```
Capital final: 14641
Plazo: 4
Interés: 0.10
El Capital Inicial de esta operación es 10000.0
```

```
i = float(input("Interes: "))
n = float(input("Plazo: "))
Cn = float(input("Capital final: "))

c0 = Cn*(1+i)**(-n)
print("El capital inicial de esta operación es: {}".format(round(c0,0)))

```

```
Interes: 0.10
Plazo: 4
Capital final: 14641
El capital inicial de esta operación es: 10000.0
```

Fórmula Nper

Tiempo

Equivalencia con fórmula Excel **NPER** (Español e Inglés)

También nos requieren los mismos parámetros que en Excel, tasa, pago, capital inicial, capital final y principio o final del periodo.

```
Cn = float(input("Capital final: "))
C0 = float(input("Capital inicial: "))
i = float(input("Interés: "))

n = npf.nper(rate=i, pmt=0, pv=C0, fv=-Cn, when='end')
print("El Plazo de esta operación es {}".format(n))
```

```
Capital final: 14641
Capital inicial: 10000
Interés: 0.10
El Plazo de esta operación es 3.9999999999999964
```

```
c0 = float(input("Capital inicial: "))
i = float(input("Interes: "))
Cn = float(input("Capital final: "))

n = (math.log(Cn/c0))/(math.log(1+i))
print("El plazo de esta inversión es {} periodos".format(round(n,0)))
```

```
Capital inicial: 10000
Interes: 0.10
Capital final: 14641
El plazo de esta inversión es 4.0 periodos
```

Ejercicios prácticos:

5. Ana tiene otra preocupación. Cuando se jubile dentro de 25 años, quiere haber ahorrado 100.000 euros para poder retirarse tranquilamente en su pueblo. Ana quiere saber cuánto tiene que ahorrar anualmente para llegar a dicha cantidad, si invierte en un producto financiero que le asegura una rentabilidad del 3%

```
fv = 100000
n=25
#pmt = ?
i = 0.03


pmt = npf.pmt(rate=i, nper=n, pv=0, fv=-fv, when='end')
print("Ana tendrá que ahorrar {:,.2f} euros anuales".format(pmt))
```

```
Ana tendrá que ahorrar 2,742.79 euros anuales
```

6. Eva ha decidido ahorrar cuando cumpla 30 años, haciendo aportaciones en un plan de pensiones que tiene una rentabilidad del 4%. Con lo ahorrado, cuando cumpla 65 años quiere obtener una mensualidad de 2.500 € al inicio de cada mes, y el importe restante se mantiene a un 3%, hasta que cumpla 90 años momento en el que quiere cobrar 100.000 €. ¿Cuánto tendrá que aportar anualmente Eva a su plan de pensiones?

```
#Primera fase del ejercicio: Calcular el capital inicial que debemos tener cuando cumpla Eva 65 años.

fv = 100000
n = 25       #12 meses * 25 años
i = 0.03     #3% / 12 meses
pmt = 2500
m = 12

pv = npf.pv(rate=i/m, nper=n*m, pmt=-pmt, fv=-fv, when='begin')
print("Eva necesita tener {:,.2f} euros cuando cumpla 65 años".format(pv))

#Segunda fase del ejercicio: Calcular las aportaciones al plan de pensiones anualmente.

fv2 = pv
n2 = 35       #12 meses * 35 años
i2 = 0.04     #4% / 12 meses
pv1 = 0
m = 12

pmt = npf.pmt(rate = i2/m, nper = n2*m, pv= pv1, fv=-fv2)
print("Eva necesita ahorrar {:,.2f} euros al mes para cumplir su objetivo".format(pmt))
```

```
Eva necesita tener 575,790.00 euros cuando cumpla 65 años
Eva necesita ahorrar 630.15 euros al mes para cumplir su objetivo
```

Rentas

Código para cálculo de rentas.

```
n = 120
pmt = 1000
i = 0.07
m = 12
im = (1+i)**(1/m)-1

#Método 1. Mediante fórmula de fv

fv = npf.fv(rate=im, nper=n, pmt=-pmt, pv=0, when='begin')
print("Resultado método 1: {} euros.".format(round(fv,2)))

#Método 2. Mediante fórmula Cf = pmt*(1+im)*(1+im)^(n)-1/im

Cf = pmt*(1+im)*((1+im)**n-1)/im
print("Resultado método 2: {} euros".format(round(Cf,2)))

```

```
Resultado método 1: 172018.88 euros.
Resultado método 2: 172018.88 euros
```

Ejemplo práctico

3. Un agricultor desea adquirir un tractor dentro de 4 años por importe de 60.000 €. Para ello decide ahorrar efectuando aportaciones trimestrales de 3.000 €, comenzando hoy mismo la primera. El depósito se remunera al 6% nominal anual. Es consciente de que a pesar de las 16 aportaciones realizadas al fondo, no llegará al montante necesario y tendrá que pagar, dentro de 4 años, un importe X para poder adquirir el tractor. Calcular X.

```
n = 4
m = n*4
fv = 60000
pmt = 3000
i = 0.06
itr = i/4
# X = ?

fv1 = npf.fv(rate=itr, nper=m, pmt=-pmt, pv=0, when='begin')
print("Al final del periodo, el agricultor ha ahorrado {} euros".format(round(fv1,2)))

x = fv - fv1
print("El agricultor tendrá que abonar {} euros dentro de 4 años".format(round(x,2)))
```

```
Al final del periodo, el agricultor ha ahorrado 54604.07 euros
El agricultor tendrá que abonar 5395.93 euros dentro de 4 años
```

```

fig, ax = plt.subplots()
ax.annotate('100 €', xy=(4, 1), xycoords='data',
            xytext=(1, 1), textcoords='data',
            arrowprops=dict(facecolor='black', shrink= 1),
            horizontalalignment='right', verticalalignment='top')
ax.annotate("133,1 €", [4, 1])
ax.annotate('100 €', xy=(4, 2), xycoords='data',
            xytext=(2, 2), textcoords='data',
            arrowprops=dict(facecolor='blue', shrink= 1),
            horizontalalignment='right', verticalalignment='top')
ax.annotate("121 €", [4, 2])
ax.annotate('100 ', xy=(4, 3), xycoords='data',
            xytext=(3, 3), textcoords='data',
            arrowprops=dict(facecolor='green', shrink= 1),
            horizontalalignment='right', verticalalignment='top')
ax.annotate("110 €", [4, 3])
ax.annotate('100 €', xy=(4, 4), xycoords='data',
            xytext=(4, 4), textcoords='data',
            arrowprops=dict(facecolor='darkblue', shrink= 1),
            horizontalalignment='right', verticalalignment='top')

plt.title("Valor final de las Rentas = 464,1", fontsize = 20)
plt.xlabel("Periodos", fontsize = 15)
plt.ylabel("Rentas recibidas", fontsize = 15)
plt.xlim(0, 5)
plt.xticks(np.arange(0, 5, 1))
plt.yticks((1, 5))
plt.show()
```

Depósitos

Código gráfico de depósitos

```
p = ['ING', 'BANKIA', 'BSCH', 'BANKINTER', 'CAIXA']
plt.figure(figsize = (10, 5))
plt.barh(p, Intereses1 , height=0.8, align='center')
plt.title("Comparativa Depósitos", fontsize = 16)
plt.xlabel("Intereses generados", fontsize = 12)
plt.ylabel("Entidad Bancaria", fontsize=12)
plt.show()
```

VAN & TIR

Código para gráfico VAN - TIR

```
cf=[-1000, 200, 300, 1000, 500]
r = 0.05

VAN = 0
for i in range(len(cf)):
    VAN += cf[i]/(1+r)**(i)    #Generamos el VAN de nuestro proyecto

L = list(range(0,351))         #Vamos a generar una lista de supuestos tipos de interés, va poder ver como cambia nuestr

rs = []
for i in L:
    rs.append(i/1000)

vans = []                      #Calculamos el VAN para cada uno de los intereses generados en la anterior lista.
for i in rs:
    van = 0
    for j in range(len(cf)):
        van += cf[j]/(1+i)**(j)
    vans.append(van)

```

```
plt.figure(figsize = (8, 6))
plt.scatter(x = r, y = VAN, s = 300, c="blue", marker = '*', label = "VAN @ r")
plt.plot(rs, vans, color = "red", linewidth = 2, linestyle='-', label = "VAN(r)")
plt.grid()
plt.hlines(y=0, xmin = rs[0], xmax=rs[-1], linestyle="dashed", color="blue", label="VAN = 0")
plt.title("VAN & TIR", fontsize=15)
plt.xlabel("Interés", fontsize=12)
plt.ylabel("VAN", fontsize=12)
plt.annotate("TIR", xy = (0.280, 0), xytext=(0.280, 100), arrowprops = {'color':'black'}, fontsize=15)
plt.legend(loc="best", fontsize=15)
plt.show()

```

Ejemplos prácticos VAN & TIR

2. Aplicando una tasa de 10% efectivo anual, calcular el VAN de una operación de inversión cuyo desembolso es de 70.000 € y consta de 8 recuperaciones semestrales de 12.000 € cada una. En este caso, dado que la renta es constate podemos aplicar tanto la fórmula de VNA o VA, vamos a verlo aplicado.

```
#Fórmula VNA

cf = [-70000, 12000, 12000, 12000, 12000, 12000, 12000, 12000, 12000]
i = 0.10
m = 2
i2 = (1+i)**(1/m)-1 # i2 es el tanto semestral efectivo

VAN = npf.npv(rate = i2, values = cf)
print("{:,.2f} €".format(VAN))

#Fórmula VA

pago = 70000
pmt = 12000
n = 8

VA = npf.pv(rate = i2, nper = n, pmt=-pmt, fv=0, when='end')-pago #tenemos que sacar fuera el pago inicial
print("{:,.2f} €".format(VA))

```

```
7,933.38 €
7,933.38 €
```

5. Calcular la TIR de una operación de inversión con las siguientes características: Desembolso de 100.000 €. Siendo las recuperaciones de 10.000 € mensuales durante un año, salvo el 3º mes que no se recupera nada y el mes 9º donde se perciben 20.000 €.

```
cf = [-100000, 10000, 10000, 0, 10000, 10000, 10000, 10000, 10000, 20000, 10000, 10000, 10000]
m = 12 # frecuencia

r12 = npf.irr(cf)     #r12 es la TIR mensual
print('Rentabilidad mensual:{}% tanto efectivo mensual'.format(r12*100))
r = (1+r12)**m-1
print('TIR = {}% tanto efectivo anual'.format(r*100))
```

```
Rentabilidad mensual:2.6973355461856086% tanto efectivo mensual
TIR = 37.62905031900152% tanto efectivo anual
```

Código gráfica TIR Múltiple

```
plt.figure(figsize = (8, 8))

plt.plot(r, VAN, color = "red", linewidth = 2, linestyle='-', label = "VAN(r)")
plt.grid()
plt.hlines(y=0, xmin = r[0], xmax=r[-1], linestyle="dashed", color="blue", label="VAN = 0")
plt.title("TIR MÚLTIPLE", fontsize=15)
plt.xlabel("Interés", fontsize=12)
plt.ylabel("VAN", fontsize=12)
plt.annotate("TIR 1", xy = (0.020, 0), xytext=(0.020, 50), arrowprops = {'color':'black'}, fontsize=15)
plt.annotate("TIR 2", xy = (0.100, 0), xytext=(0.100, 50), arrowprops = {'color':'black'}, fontsize=15)
plt.annotate("TIR 3", xy = (0.360, 0), xytext=(0.360, 50), arrowprops = {'color':'black'}, fontsize=15)
plt.legend(loc="best", fontsize=15)
plt.show()
```

Class de Proyectos. Código de funciones previas

```python
#Damos forma a determinadas funciones que después vamos a incorporar en nuestra class

def VAN(rate, values):     #para su calculo necesitamos una tasa de descuento y unos flujos de caja
    VAN = 0
    for i in range(len(values)):
        VAN += values[i]/(1+rate)**(i)
    return VAN

def TIR(values):           #Para el cálculo de la TIR solo necesitamos los flujos de caja
    TIR = npf.irr(values)
    return TIR

def PAYBACK(values):       #Igualmente, para el Payback solo necesitamos los flujos de caja

    Acum_FC = 0

    for i in range(len(FC)):
        Acum_FC += FC[i]

        if Acum_FC > 0:
            return i
            break

        elif Acum_FC <= 0 and i == len(FC)-1:
            print("El proyecto no recupera su inversión")


def IR(rate, pv, values):    #Para el índice de rentabilidad necesitamos varios datos.
    values = values[1:]

    VA = npf.npv(rate, values)
    IR = VA / pv
    return IR

```

Propiedades y atributos. Init de Class

```python


#Definimos class Proyecto, empezando por las propiedades de las variables, evitando valores nulos.

class Proyecto:

    @property
    def nper(self): return self._nper #Todo proyecto debe tener un periodo superior a 0
    @nper.setter
    def nper(self, nper):
        if type(nper) == int and nper > 0:
            self._nper = nper
        else: print("La duración del periodo debe ser > 0")

    @property
    def pv(self): return self._pv #Todo proyecto debe tener una inversión inicial superior a 0
    @pv.setter
    def pv(self, pv):
        if (type(pv) == int or type(pv) == float) and pv > 0:
            self._pv = pv
        else: print("La inversión inicial debe ser > 0")

    @property
    def rate(self): return self._rate #En este caso, vamos a pedir una tasa de interés, más adelante veremos calcular
    @rate.setter
    def rate(self, rate):
        if type(rate) == float and rate > 0:
            self._rate = rate
        else: print("El interés (8% = 0.08) de ser > 0")
```

```
    def __init__(self, rate, nper, pv, values):  #Definimos qué datos necesitamos para analizar nuestro proyecto
        self.rate = rate            #tasa de descuento
        self.nper = nper            #duración del proyecto
        self.pv = pv                #inversión inicial
        self.values = values        #flujos de caja del proyecto (más adelante veremos si podemos calcularlos)

    def VAN(self):
        return VAN(self.rate, self.values)   # definimos la función para todos los objetos de Proyecto.

    def TIR(self):
        return TIR(self.values)              # definimos la función para todos los objetos de Proyecto

    def PAYBACK(self):
        return PAYBACK(self.values)          # definimos la función de Payback para Proyecto

    def IR(self):
        return IR(self.rate, self.pv, self.values) # definimos la función de Indice Rentabilidad para Proyecto

    def ANALISIS(self):
        return "Este proyecto tiene un VAN de {}, una TIR de {}, su PayBack es de {} años, y su indice de retorno es de
```

Aplicación

```
Proyecto1.PAYBACK()
```

```
El Payback del Proyecto son 3 años
```

```
Proyecto1.IR()
```

```
'El Indice de Rentabilidad es 1.8246625634380738.'
```

```
Proyecto1.ANALISIS()
```

```
'Este proyecto tiene un VAN de 737.77, una TIR de 0.28, su PayBack es de 3 años, y su indice de retorno es de 1.82'
```

ualmente, podemos tener tantos proyectos como queramos, y analizarlos todos

```
Proyecto2 = Proyecto(0.05,3,1000,[-1000, 500, 500, 500])
Proyecto3 = Proyecto(0.03,4,500,[-500, 200, 200, 200, 200])
Proyecto4 = Proyecto(0.02,2,200,[-200, 500, 500])
Proyecto5 = Proyecto(0.08,5,1500,[-1500, 400, 400, 400, 400, 400])
Proyecto6 = Proyecto(0.03,3,400,[-400, 500, 500, 500])
Proyecto7 = Proyecto(0.07,2,800,[-800, 600, 600])
Proyecto8 = Proyecto(0.05,4,700,[-700, 400, 400, 400, 400])
Proyecto9 = Proyecto(0.06,3,900,[-900, 800, 800, 800])
```

```
Proyecto2.ANALISIS()
```

```
'Este proyecto tiene un VAN de 361.62, una TIR de 0.23, su PayBack es de 3 años, y su indice de retorno es de 1.43'
```

```
Proyecto3.ANALISIS()
```

```
'Este proyecto tiene un VAN de 243.42, una TIR de 0.22, su PayBack es de 3 años, y su indice de retorno es de 1.53'
```

Letras de cambio, letras del tesoro y bonos.

Ejemplos

```
#desarrollo para introducir datos manualmente

datos_vto = input('Introduzca la fecha de vencimiento del activo (forma YYYY-MM-DD) ')
año, mes, dia = map(int, datos_vto.split('-'))
vencimiento = date(año, mes, dia)
datos_liq = input('Introduzca la fecha de liquidación del activo (forma YYYY-MM-DD) ')
año, mes, dia = map(int, datos_liq.split('-'))
liquidacion = date(año, mes, dia)
base = float(input('''Introduzca:
                      1 para base de 365 días
                      2 para base de 360 días
                      3 para meses de 30 dias y año 360 '''))
Amortizacion = float(input('Introduzca el valor de reembolso del activo '))
pr = float(input('Introduzca el precio del activo '))

RENDTO_DESC(liquidacion,vencimiento,pr,amortizacion,base)
```

```
Introduzca la fecha de vencimiento del activo (forma YYYY-MM-DD)2013-03-07
Introduzca la fecha de liquidación del activo (forma YYYY-MM-DD)2013-01-07
Introduzca:
                    1 para base de 365 días
                    2 para base de 360 días
                    3 para meses de 30 dias y año 3603
Introduzca el valor de reembolso del activo2421.38
Introduzca el precio del activo2458
```

```
0.08939
```

```
#Introducción datos manuales


par = float(input('Introduzca el valor nominal del Bono '))
datos_emi = input('Introduzca la fecha de emisión del Bono (forma YYYY-MM-DD)')
año, mes, dia = map(int, datos_emi.split('-'))
f_emi = date(año, mes, dia)
datos_liq = input('Introduzca la fecha de liquidación del Bono (forma YYYY-MM-DD)')
año, mes, dia = map(int, datos_liq.split('-'))
f_liq = date(año, mes, dia)
tasa = float(input('Introduzca el tipo de interés (Ejemplo 8%: 0.08) : '))
frecuencia = float(input('''Introduzca la frecuencia:
                                Para años = 1
                                Para semestres = 2
                                Para trimestres = 4 '''))
base = float(input("Intoduzca 1 para base 365 días, 2 para base 360 días"))

INT_ACUM(f_emi, f_liq, tasa, par, frecuencia, base)
```

```
Introduzca el valor nominal del Bono 1000
Introduzca la fecha de emisión del Bono (forma YYYY-MM-DD)2014-03-15
Introduzca la fecha de liquidación del Bono (forma YYYY-MM-DD)2016-06-19
Introduzca el tipo de interés (Ejemplo 8%: 0.08) : 0.0325
Introduzca la frecuencia:
                              Para años = 1
                              Para semestres = 2
                              Para trimestres = 4 1
Intoduzca 1 para base 365 días, 2 para base 360 días1
```

```
73.548
```

Préstamos

Código para elaborar cuadros de amortización

```python
#Hacemos un cuadro de amortización para ver los datos con la librería tabulate
datos =[]
saldo = C0
saldo2 = 0
linea1 = [0,0,0,0,C0,0]
datos.append(linea1)

Anualidad = npf.pmt(rate=tasa, nper=n, pv=-C0, fv=0, when='end')

for i in range(1, n+1):
    pago_capital = npf.ppmt(rate=tasa, per=i, nper=n, pv=-C0, fv=0, when='end')
    pago_int = Anualidad - pago_capital
    saldo -= pago_capital
    saldo2 += pago_capital

    linea = [i, format(Anualidad, '.0f'),  format(pago_int, '.0f'),
             format(pago_capital, '.0f'), format(saldo, '.0f'), format(saldo2, '.0f')]

    datos.append(linea)

print(tab.tabulate(datos, headers= ['Periodo', 'Anualidad', 'Intereses',
                                    'Amortización','Capital Vivo','Capital Amortizado'],
                   tablefmt = 'psql'))

```

Código para gráficos

```python
plt.figure(figsize = (8, 4))
plt.bar(range(0, n+1 ), Amortz, label = 'amortizacion')
plt.bar(range(0, n+1 ), inter, bottom= Amortz, label='intereses')
plt.legend(fontsize = 10)
plt.title("Método Italiano", fontsize = 15)
plt.xlabel("Periodos en años", fontsize = 12)
plt.ylabel("Pagos en euros", fontsize = 12)
plt.plot()
```

Código para comparativa de prestamos

```python
Capital = float(input("Importe del Préstamo: "))
tasa = float(input("Tipo de interés del Préstamo (Ej. 8% = 0.08): "))
años = int(input("Periodo del Préstamo (años): "))
meses = int(input("Introduzca 0 para prestamos anuales o 1 para prestamos mensuales"))
tipo = input("Tipo de Préstamo (Francés, Americano, Italiano): ").lower()

frances = ['frances', 'f']
americano = ['americano', 'a']
italiano = ['italiano', 'i']

if tipo in frances:
    PFrances(Capital, tasa, años, meses)

elif tipo in americano:
    PAmericano(Capital, tasa, años, meses)

elif tipo in italiano:
    PItaliano(Capital, tasa, años, meses)


```

Beta de compañías

Código para gráficos

```
data['Rentabilidad Valor'].cumsum().plot(subplots=True,figsize=(16,7))
```

```
def plot_regression(renta, asset):
    plt.figure(figsize=(10,5))
    sns.scatterplot('^IBEX', asset, data=renta,
                    hue=renta.index.year, alpha=0.6, legend=False)
    sns.regplot('^IBEX', asset, data=renta, scatter=False, color='green')

asset = 'AMS.MC'
plot_regression(renta, asset)
```

Código para automatizar el cálculo de Beta

```
def beta(valor,periodo,mercado):       #asignamos las variables
    simbol = [valor, mercado]          #creamos una lista para después extraer los datos de la página de yahoo
    data = yf.download(simbol, period=periodo, auto_adjust=True) #descargamos los datos que necesitamos
    renta = np.log(1 + data.loc[:,'Close'].pct_change()).dropna(how = 'all') #calculamos la rentabilidad
    var = renta[(mercado)].var() #calculamos la varianza del mercado
    cov = renta.cov()            #calculamos la covarianza del valor
    beta = renta.cov()/var       #calculamos la Beta de la compañía
    return  beta[mercado].head(1).to_frame('Beta').T  #Finalmente mostramos el dato que necesitamos.
```

```
valor = 'IAG.MC'  #recordemos que debemos indicar el simbolo de la empresa.
periodo = '5y'    #podemos selecionar varios periodos. (1d,5d,1mo,3mo,6mo,1y,2y,5y,10y,ytd,max)
mercado = '^IBEX'

beta(valor, periodo, mercado)
```

```
[**********100%**********]  2 of 2 completed
```

	IAG.MC
Beta	1.664964

Automatización de informes

Código para gráficos

```
plt.figure(figsize=(6,10))
sector.plot(kind='pie',  subplots=True, fontsize=12, ylabel='', colormap ='mako')
plt.title("Sectores mas influyentes en el IBEX35", fontsize=20)

plt.savefig("Ibex.jpg", bbox_inches='tight') #nos guarda la imagen

plt.show()
```

```
listaEmpresas = unido.groupby('Sector').Empresa.count().sort_values(ascending=False)
listaEmpresas.plot(kind='barh', figsize=(6,6), fontsize=13, color="darkblue")
plt.title("Número de empresas por Sectores en Ibex35")

plt.savefig("NumeroEmpresas.jpg", bbox_inches='tight')

plt.show()
```

```
lugar = unido.groupby('Sede').Empresa.count().nlargest(5)
lugar.plot(kind='pie',  subplots=True, figsize=(6,10), fontsize=13, ylabel='', colormap='mako')
plt.title('Ciudades con más empresas del Ibex', fontsize=20)

plt.savefig("Ciudades.jpg", bbox_inches='tight') #es importante guardar la imagen antes de visualizarla,

plt.show()

```

```
entrada = unido.groupby('Entrada').Empresa.count()
entrada.plot(kind='bar',  subplots=True, figsize=(9,3), fontsize=10, ylabel='', color='darkblue')
plt.title("Nº de Empresa que han entrado en el Ibex por año", fontsize=20)
plt.ylabel('Año Entrada')
plt.xlabel('Nº de Empresas')

plt.savefig("Empresa.jpg", bbox_inches='tight')


plt.show()

```

```
evoluci['^IBEX'].plot(figsize=(10,5));
plt.title('Evolución Ibex35 en los últimos meses', fontsize=15)
plt.ylabel('Precio de Cierre')
plt.xlabel('Fechas')
plt.yticks(range(7800,8700,100));

plt.savefig("Evolucion.jpg", bbox_inches='tight')

plt.show()
```

Código completo automatización de informes

```
document = Document()
document.add_heading("ANÁLISIS DE IBEX_35", level=0)
document.add_heading("1. Características del Ibex35", level=1)
document.add_paragraph("El Ibex se compone de las 35 empresas con más liquidez que cotizan en el sistema bursátil español que está formado por las bolsas de Madrid, Valencia, Barcelona y Bilbao. Por ello se utiliza como referencia para conocer la situación de la Bolsa española.")
document.add_paragraph("Si comprobamos la distribución por sectores, podemos comprobar que Electricidad y Gas es el sector más representativo, ya que supone un 22% del Ibex, seguido de la Banca, Textil y calzado, Telecomunicaciones y Minerales-transformación.")

document.add_picture("Ibex.jpg", width=Cm(11))

document.add_paragraph("El gráfico nos muestra los 5 sectores más representativos, siendo un total de 17 sectores los que están representados en el Ibex.35")
document.add_paragraph("En la siguiente tabla comprobamos la distribución por sectores del Ibex")

table = document.add_table(rows=1, cols=2, style='Colorful Shading Accent 1')
#font = row_cells[0].text
table.height = Pt(8)

table.rows[0].cells[0].text = 'Sectores'
table.rows[0].cells[1].text = 'Ponderación'


for i,p  in zip(sectores, ponderaciones):
    row_cells = table.add_row().cells
    row_cells[0].text = i
    row_cells[1].text = str(round(p,2)) + '%'
```

```
document.add_paragraph("")
document.add_paragraph("En el siguiente gráfico podemos comprobar el número de empresas totales por sector")
document.add_picture("NumeroEmpresas.jpg", width=Cm(10), height=Cm(8))

document.add_paragraph('')
document.add_paragraph('En el siguiente gráfico vemos las 5 ciudades donde más empresas tienen sus sedes.')
document.add_picture("Ciudades.jpg", width=Cm(7))
document.add_paragraph("""Madrid es la ciudad donde más empresas tienen su sede, seguido de Alcobendas y Bilbao. Después le siguen Barcelona y Alicante.
Posiblemente podemos encontrar distintos motivos por los que la empresas deciden establecer su sede en un punto geográfico y no otro, así las infraestructuras, la innovación tecnológica o su capital humano son factores que influyen a la hora de establecer la sede de una empresa.""")

document.add_paragraph('El Ibex se creó en 1.992, por ello no es de extrañar que nos encontremos con el año en el que más entradas se produjeron en el Ibex.')
document.add_paragraph('En el siguiente gráfico podemos ver el número de entradas por año.')
document.add_picture('Empresa.jpg', width=Cm(12))
document.add_paragraph('Podemos ver una entrada en el año 1969, que probablemente sea un error publicado en la fuente de origen de los datos, Wikipedia, puesto que la creación del Ibex fue en 1.992, y por tanto el año 1969 debe ser el año de creación de la propia empresa.')


document.add_heading("2. Análisis evolución Ibex 35", level=1)
document.add_paragraph("En el siguiente gráfico podemos ver la evolución del Ibex 35 de los últimos 3 meses")
document.add_picture("Evolucion.jpg", width=Cm(12))
document.add_paragraph("La tendencia, después de meses marcados por pérdidas debido al coronavirus, es alcista. Parece que las previsiones económicas en 2021 establecen recuperaciones económicas tanto para Europa como para España, lo que inyecta confianza en los inversores.")
document.add_paragraph("Esto se ve reflejado en las subidas experimentadas en el Ibex a lo largo de estos meses.")


document.add_paragraph("En la siguiente tabla podemos comprobar el precio de medio de los últimos 3 meses")
table = document.add_table(rows=1, cols=2, style='Colorful Shading Accent 1')
table.height = Pt(8)

table.rows[0].cells[0].text = 'Empresas'
table.rows[0].cells[1].text = 'Precio medio 3 meses'


for s,m  in zip(simbolos, medias):
    row_cells = table.add_row().cells
    row_cells[0].text = s
    row_cells[1].text = str(round(m,2)) + '€'


document.add_paragraph('')

document.add_paragraph("No todos los valores sufren las mismas variaciones. Para saber si los valores oscilan mucho en precios, podemos calcular la desviación típica que nos dará una idea de cómo han variado en precio durante los últimos 3 meses.")


document.add_paragraph('')
document.add_paragraph('')


document.add_paragraph("En la siguiente tabla, podemos ver la variación del Ibex en primer lugar, seguido de los 4 valores que más variaciones han sufrido en estos meses:")


table = document.add_table(rows=1, cols=2, style='Colorful Shading Accent 1')
```

```
table.height = Pt(8)


table.rows[0].cells[0].text = 'Empresas'
table.rows[0].cells[1].text = 'Variacion típica en 3 meses'


for s,m  in zip(empr, mayor):
    row_cells = table.add_row().cells
    row_cells[0].text = s
    row_cells[1].text = str(round(m,2))


document.save('Ibex.docx')
```

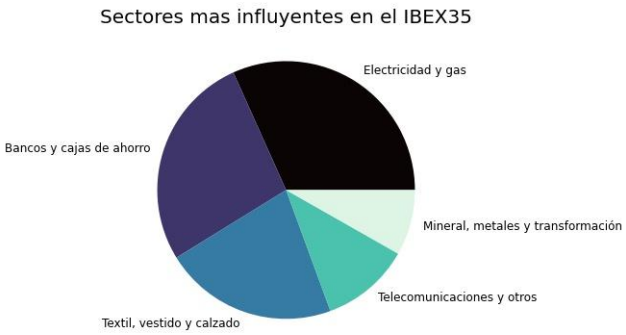
Parte del Informe automatizado (2 páginas), copiado y pegado para que puedan ver cómo quedaría. El informe completo lo pueden encontrar en los archivos comprimidos incluidos en los anexos.

ANÁLISIS DE IBEX_35

1. Características del Ibex35

El Ibex se compone de las 35 empresas con más liquidez que cotizan en el sistema bursátil español que está formado por las bolsas de Madrid, Valencia, Barcelona y Bilbao. Por ello se utiliza como referencia para conocer la situación de la Bolsa española.

Si comprobamos la distribución por sectores, podemos comprobar que Electricidad y Gas es el sector más representativo, ya que supone un 22% del Ibex, seguido de la Banca, Textil y calzado, Telecomunicaciones y Minerales-transformación.



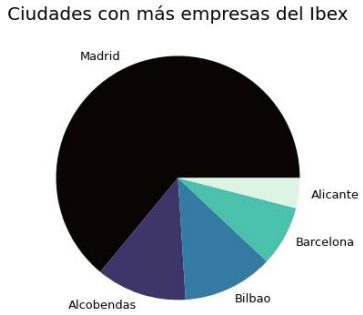
El gráfico nos muestra los 5 sectores más representativos, siendo un total de 17 sectores los que están representados en el Ibex.35

En la siguiente tabla comprobamos la distribución por sectores del Ibex

Sectores	Ponderación
Electricidad y gas	22.19%
Bancos y cajas de ahorro	18.95%
Textil, vestido y calzado	15.25%
Telecomunicaciones y otros	7.83%
Mineral, metales y transformación	5.75%
Construcción	5.74%
Transporte y distribución	5.63%
Electrónica y software	4.97%
Fabricación y montaje de bienes de equipo	3.91%
Petróleo	2.99%
Productos farmacéuticos y biotecnología	2.38%
SOCIMI	1.44%
Seguros	0.95%
Equipamiento deportivo	0.83%
Alimentación y bebidas	0.48%
Energías renovables	0.39%
Ocio, turismo y hostelería	0.24%

En el siguiente gráfico podemos comprobar el número de empresas totales por sector

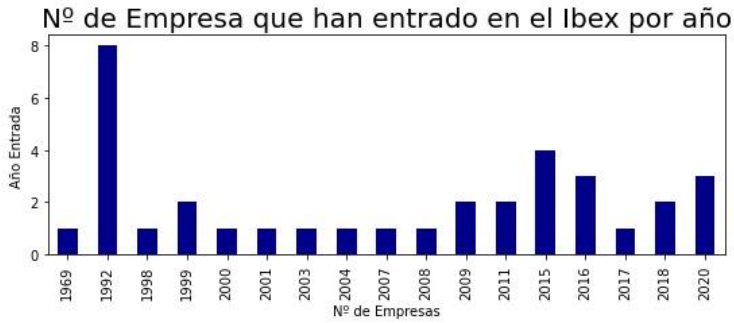
En el siguiente gráfico vemos las 5 ciudades donde más empresas tienen sus sedes.



Madrid es la ciudad donde más empresas tienen su sede, seguido de Alcobendas y Bilbao. Después le siguen Barcelona y Alicante.
Posiblemente podemos encontrar distintos motivos por los que las empresas deciden establecer su sede en un punto geográfico y no otro, así las infraestructuras, la innovación tecnológica o su capital humano son factores que influyen a la hora de establecer la sede de una empresa.

El Ibex se creó en 1.992, por ello no es de extrañar que nos encontremos con el año en el que más entradas se produjeron en el Ibex.

En el siguiente gráfico podemos ver el número de entradas por año.



Podemos ver una entrada en el año 1969, que probablemente sea un error publicado en la fuente de origen de los datos, Wikipedia, puesto que la creación del Ibex fue en 1.992, y por tanto el año 1969 debe ser el año de creación de la propia empresa.

En cualquier caso, el código completo se encuentra en el repositorio de GitHub donde se puede descargar de forma libre. Igualmente se incluye en la plataforma un archivo zip con el código descargado.