1. Basics of Financial Math

1.1 Time Value of Money

r = annual interest rate

FV = future value , PV = present value

$FV = PV(1+r)^n$, n = # of years

$PV = FV(1+r)^{-n}$

if interest is compounded m times per year: $FV = PV \cdot \left(1+\frac{r}{m}\right)^{n \cdot m}$

terminology:
- BEY (bond-equivalent yield)
  annualized yield, compounded semi-annually
- MEY (mortgage-equivalent yield)
  annualized yield, compounded monthly

effective annual interest rate: $(1+r_{eff})^n = \left(1+\frac{r}{m}\right)^{nm}$

$$\Rightarrow r_{eff} = \left(1+\frac{r}{m}\right)^m - 1$$

↳ allows to compare instruments with different compounding standards

Example: $r = 10\%$, compounded twice a year

$$r_{eff} = \left(1+\frac{0.1}{2}\right)^2 - 1 = 0.1025 = 10.25\%$$

Is r_{eff} always bigger than r? $(r>0)$

$$r_{eff} = \left(1+\frac{r}{m}\right)^m - 1 = 1 + m\frac{r}{m} + \underbrace{\text{rest}}_{>0} - 1 > r \quad \text{Yes}$$

continuous compounding: $\lim_{m\to\infty}$

$$FV = PV \lim_{m\to\infty} \left(1+\frac{r}{m}\right)^{m\cdot n} = PV \cdot e^{rn}$$ as opposed to periodic compounding above

$$\left(1+\frac{r}{m}\right)^m = e^r$$

1.2 General Cash Flows

n years, y is the yearly interest rate

at end of year there is cash flows $C_1, C_2, \ldots, C_n$

$$PV = \frac{C_1}{(1+y)} + \frac{C_2}{(1+y)^2} + \ldots + \frac{C_n}{(1+y)^n} = \sum_{i=1}^{n} \frac{C_i}{(1+y)^i}$$

with $x = \frac{1}{1+y}$ => have to evaluate polynomials $\sum_{i=1}^{n} C_i x^i$

Implementation in python:

- explicit loop:

```
x = ...
C = ...
for i in arange(0, len(C))
    ...
...
```

(bad and slow)

- Horner's scheme:

$$PV = \left(\left(\left(\left(C_n x + C_{n-1} \right) x + C_{n-2} \right) x + \ldots + C_1 \right) x \right.$$

- polyval (fct. from SciPy), uses optimized Horner's scheme
  ↳ look up documentation

- best: use vectorized operations (here: NumPy arrays)

  i = arange(1, n+1) , C = array([..., ..., ...])

  use dot product for vectors: $PV = \text{dot}(\underset{\text{vector}}{C}, \underset{\text{vector}}{(1+y)**(-i)})$

for HW: Python fct.s:

```
def PV(C,y):
    ...
    ...
    return ...
...
```

indent

<u>Annuity</u>: $C_1 = C_2 = \ldots = C_n = C$

ordinary annuity (usually assumed): pays C at end of the year

$$FV = \sum_{i=0}^{n-1} C(1+r)^i = C \sum_{i=0}^{n-1} (1+r)^i$$

geometric series: $x\sum_{i=0}^{n} x^i = \sum_{i=1}^{n+1} x^i = x^{n+1} - 1 + \sum_{i=0}^{n} x^i$

$$(x-1)\sum_{i=0}^{n} x^i = x^{n+1} - 1 \Rightarrow \sum_{i=0}^{n} x^i = \frac{1-x^{n+1}}{1-x}$$

$$FV = C\left(\frac{(1+r)^n - 1}{r}\right)$$

$\boxed{r \equiv y}$

also: annuity due : pays C at beginning of year

$$FV = \sum_{i=1}^{n} C(1+r)^i = C(1+r)\sum_{i=0}^{n-1}(1+r)^i = C(1+r)\left(\frac{(1+r)^n - 1}{r}\right)$$

general annuity: m payments per year

↳ ordinary: $FV = \sum_{i=0}^{nm-1} C\left(1+\frac{r}{m}\right)^i = C\left(\frac{\left(1+\frac{r}{m}\right)^{nm} - 1}{\frac{r}{m}}\right)$

What is PV?

$$PV = \sum_{i=1}^{m\cdot n} C\left(1+\frac{r}{m}\right)^{-i} = C\sum_{i=1}^{m\cdot n}\left(\frac{1}{1+\frac{r}{m}}\right)^i$$

$$= C\left(\frac{1}{1+\frac{r}{m}}\right)\left(\frac{\left(1+\frac{r}{m}\right)^{-nm} - 1}{\left(\frac{1}{1+\frac{r}{m}}\right) - 1}\right)$$

$$= C\left(\frac{1-\left(1+\frac{r}{m}\right)^{-nm}}{\frac{r}{m}}\right)$$

$n\to\infty$: perpetual annuity

↳ $PV = \lim_{n\to\infty} C\left(\frac{1-\left(1+\frac{r}{m}\right)^{-n\cdot m}}{\frac{r}{m}}\right) = C\frac{m}{r}$

Amortization:

→ repay loan with regular payments

↳ you pay principal + interest

traditional mortgage = equal regular payments

$$C = PV\left(\frac{\frac{r}{m}}{1-\left(1+\frac{r}{m}\right)^{-n\cdot m}}\right)$$

remaining principal after k payments: $\sum_{i=1}^{m\cdot n-k} C\left(1+\frac{r}{m}\right)^{-i}$

↳ HW: create an amortization schedule

Internal Rate of Return (IRR):

IRR = the period interest rate y, given cash flows $C_1, \dots, C_n$,

for a financial instrument sold at price P now (= present value)

solve $PV(y) = \sum_{i=1}^{n} \frac{c_i}{(1+y)^i} = P$ for y

net-present value (NPV): $NPV(y) = PV(y) - P$

at the IRR, the NPV is zero.

↳ HW: use python fct. brentq (look up documentation)