Python in Finance

ΤΜΗΜΑ ΛΟΓΙΣΤΙΚΗΣ ΚΑΙ ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΗΣ

Αξία του χρήματος

Η έννοια της αξίας του χρήματος αποτελέι ένα από τα πιο σημαντικά κομμάτια στην ανάλυση τον επενδύσεων

Καθότι η αξία του χρήματος μεταβάλλεται με την πάροδο του χρόνου,η λήψη απόφασης σχετικά με μια επένδυση σήμερα εξαρτάται από την μελλοντική αξία του χρήματος

Γενικά η αξία μειώνεται με την πάροδο του χρόνου. Για πάραδειγμα η αξία 1€ σήμερα είναι μεγαλύτερη από αυτή σε κάποια μελλοντική χρονική στιγμή.

Υπάρχουν 2 κύριοι λόγοι σχετικά με αυτην την αλλαγή στην αξία

Αξία του χρήματος

- Οι ταμειακές ροές σε διαφορες χρονικές περιόδους έχουν διαφορετική αξία:

Όπως αναφέραμε και προηγουμένως η αξία 1€ δεν είναι το ίδιο πολύτιμή με αυτή σε έναν χρόνο. Επίσης θα μπορούσε κάποιος να επενδύσει ένα συγκεκριμένο ποσό σήμερα, και μέσω του τόκου το επόμενο έτος η αξία της επένδυσης του να έχει αυξηθεί. Με λίγα λόγια πρέπει να λαμβάνουμε υπόψιν την αξία του χρήματος για να αξιολογήσουμε τις ταμειακές ροές στην πάροδο του χρόνου

Αξία του χρήματος

- Οι ταμειακές ροές είναι αβέβαιες

Ορισμένες ταμειακές ροές ενδέχεται να μην πραγματοποιηθούν στο μέλλον. Η αβεβαιότητα πηγάζει σχετικά με την πρόβλεψη σχετικά με το χρονοδιάγραμμα και το ποσο των χρηματικών ροών. Με λίγα λογια δεν γνωρίζουμε με βεβαιότητα, τον χρόνο η το ποσό της ταμειακής ροής που θα αποπληρωθεί στο μέλλον. Αυτή η αβεβαιότητα σχετικά με την ταμειακή ροή, πρέπει με κάποιον τρόπο να ληφθεί υπόψιν σχετικά με την αξιολόγηση μια επένδυσης (π.χ Δάνειο)

Αξία του χρήματος

- Προεξόφληση(Discounting)

Με τον όρο προεξόφληση εννοούμε την διαδικασία οπου χρησιμοποιόντας μελλοντικές αξίες(Future Value) υπολογίζουμε την καθαρή παρούσα αξία μιας επένδυσης(Present Value)

- Ανατοκισμός(Compounding)

Αντίστοιχα με την έννοια του ανατοκισμού μετατρέπουμε την παρούσα αξία μιας επέδυσης(Present Value) σε μελλοντική(Future Value)

Αξία του χρήματος

Έστω δανείζουμε ένα ποσό π.χ. 100€ με την συμφωνία αποπληρωμής του ποσού σε 1 μήνα. Αν μας επιστρεφόταν το ποσό των 100€ θα ήταν δίκαιο ; Πιθανόν όχι.

Αρχικά , το γεγονός πως παρέχουμε ένα συγκεκριμένο ποσό σε κάποιον δανειολήπτη αποτελεί το κόστος ευκαιρίας του χρήματος(τόκος).Επίσης ποια είναι η πιθανότητα αθέτησης της αποπληρωμής μετά από 1 μήνα ;

Εκτός από το κόστος ευκαιρίας , πρέπει να λάβουμε υποψιν και την αβεβαιότητα της μη επιστροφής του κεφαλαίου μας την χρονική περίοδο που έχει συμφωνηθεί

Αξία του χρήματος

- Ετσι λοιπόν σε περίπτωση δανείου 100€ έχουμε :

1. Το αρχικο κεφάλαιο 100€ (Principal)
2. Ένα είδος «αποζημίωσης» για το κόστος ευκαιρίας, και την αβεβαιότητα της αποπληρωμής του δανείου στην συμφωνηθέντα χρονική περίοδο(interest rate)

Το ποσό που έχουμε σκοπό να δανείσουμε σήμερα αποτελεί την παρούσα αξία του δανείου και αντίστοιχα το πόσο που αναμένουμε να εισπράξουμε αποτελεί την μέλλουσα αξία

Αξία του χρήματος

Παράδειγμα:

Εστω κάνουμε μια κατάθεση 1000€ στην τράπεζα ‘ΑΒΓ’ με 5% επιτόκιο ανα έτος.

Στο τέλος του έτους το κεφάλαιο μας θα ισούται με 1050€.Ουσιαστικά αποτελείται από το αρχικό μας κεφάλαιο (1000€) συν τον τόκο των 50€.

- Παρούσα αξία της επένδυσης μας 1000€ (PV)
- Μέλλουσα αξία της επένδυσης 1050€ (FV)
- Τόκος 5% επι του αρχικού μας κεφαλαίου

Αξία του χρήματος :Μέλλουσα Αξία

Αν πραγματοποιήσουμε μια κατάθεση σήμερα με ετήσιο επιτόκιο 10%

Ποια είναι η αξία του χρήματος σε 1 ετος από σήμερα ;

PV=100

R=0.10

n=1

FV=PV*(1+R)**n

FV

$$FV = PV(1+R)^n$$

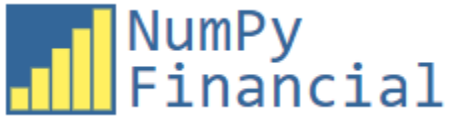
Αξία του χρήματος : Παρούσα Αξία

Αντιστοιχά αν θέλαμε να υπολογίσουμε την παρούσα αξία του χρήματος θα εργαζόμασταν ως εξής :

```
FV=100
R=0.10
n=2
PV=FV/(1+R)**n
PV
```

$$PV = \frac{FV}{(1+R)^n}$$

NumPy-Financial



Το πακέτο NumPy-Financial αποτελεί ένα οικονομικό πακέτο της Python το οποίο μας παρέχει κάποιες βασικές οικονομικές συναρτήσεις. Γενικά αυτές οι συναρτήσεις ήταν διαθέσιμες από την NumPy , όμως από το 2013 δημιουργήθηκε το ξεχωριστό πακέτο NumPy-Financial.

Ένας από τους κύριους λόγους είναι πως αυτές οι συναρτήσεις θεωρήθηκαν ειδικά εξιδεικευμένες για την NumPy

NEP 32 — Remove the financial functions from NumPy — NumPy Enhancement Proposals

Στις επόμενες διαφάνειες θα δούμε εφαρμογές αυτών των συναρτήσεων, για περισσότερες πληροφορίες μπορείτε να ανατρέξετε και στο numpy-financial 1.0.0 — numpy-financial documentation

NumPy-Financial

- Πριν ξεκινήσουμε με τις συναρτήσεις ας ξεκαθαρίσουμε κάτι. Γενικά αν ανατρέξετε στο official site της NumPy-Financial στις συναρτήσεις fv,pv,pmt,rate θα δείτε

```
fv +
pv*(1 + rate)**nper +
pmt*(1 + rate*when)/rate*((1 + rate)**nper - 1) == 0
```

or, when `rate == 0`:

```
fv + pv + pmt * nper == 0
```

Καθώς η παρούσα αξία και οι πληρωμές θεωρούνται κινήσεις κεφαλαίου προς τα έξω, θα τα εισάγουμε με αρνητικό πρόσημο.

NumPy-Financial

pv

`numpy_financial.pv(rate, nper, pmt, fv=0, when='end')`

Compute the present value.

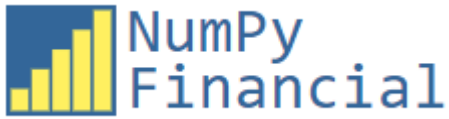
Given:

- a future value, *fv*
- an interest *rate* compounded once per period, of which there are
- *nper* total
- a (fixed) payment, *pmt*, paid either
- at the beginning (*when* = {'begin', 1}) or the end (*when* = {'end', 0}) of each period

Return:

the value now

NumPy-Financial



Με την συνάρτηση pv υπολογίζουμε την παρούσα αξία μιας επένδυσης.

Ποια είναι η παρουσα αξία μιας επένδυσης οπου χρειάζεται $15692.93 μετα από 10 έτη αν αποταμιεύουμε μηνιαία $100 ;Υποθέστε ότι το ετήσιο επιτόκιο ισούται με 5% και ο ανατοκισμός γίνεται μηνιαία.

```
import numpy_financial as npf
npf.pv(0.05/12, 10*12, -100, 15692.93)
```

NumPy-Financial

Η πρώτη συνάρτηση που θα δούμε είναι η fv (future value)

fv

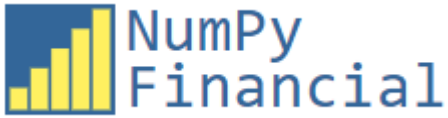
`numpy_financial.fv(rate, nper, pmt, pv, when='end')`

Compute the future value.

Given:

- a present value, *pv*
- an interest *rate* compounded once per period, of which there are
- *nper* total
- a (fixed) payment, *pmt*, paid either
- at the beginning (*when* = {'begin', 1}) or the end (*when* = {'end', 0}) of each period

NumPy-Financial



Αντίστοιχα με το προηγούμενο μας παράδειγμα θα υπολογίσουμε την μέλλουσα αξία της επένδυσης μας.

Οπότε ξεκινάμε με παρούσα αξία στα $100 και αποταμίευση στα $100/μήνα με ετησιο επιτοκιο 5% και μηνιαίο ανατοκισμό.

```
import numpy_financial as npf
npf.fv(0.05/12, 12*10,-100,-100)
```

NumPy-Financial

rate

`numpy_financial.rate(nper, pmt, pv, fv, when='end', guess=None, tol=None, maxiter=100)`

Compute the rate of interest per period.

Parameters:

nper : *array_like*
Number of compounding periods

pmt : *array_like*
Payment

pv : *array_like*
Present value

fv : *array_like*
Future value

when : *{{'begin', 1}, {'end', 0}}, {string, int}, optional*
When payments are due ('begin' (1) or 'end' (0))

guess : *Number, optional*
Starting guess for solving the rate of interest, default 0.1

tol : *Number, optional*
Required tolerance for the solution, default 1e-6

maxiter : *int, optional*
Maximum iterations in finding the solution

NumPy-Financial

Τέλος με την χρήση της Rate θα μπορούσαμε να υπολογίσουμε το επιτόκιο.

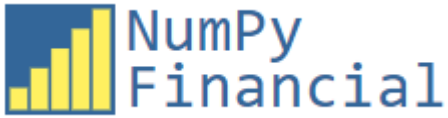
Έχουμε λοιπόν :

Αρχική επένδυση στα $100 με μηνιαίες καταθέσεις επίσης στα $100.Αν θέλουμε να έχουμε μέλλουσα αξία στα $ 15692.93 σε πάροδο 10 ετών με μηνιαίο ανατοκισμό, πόσο ισούται το επιτόκιο ;

```
Import numpy_financial as npf
npf.rate(12*10,-100,-100,15692.9288)
```

Βλέπουμε ότι το αποτέλεσμα πράγματι ισούται με 0.05/12≈0,0041

NumPy-Financial



irr

`numpy_financial.irr(values)`

Return the Internal Rate of Return (IRR).

This is the "average" periodically compounded rate of return that gives a net present value of 0.0; for a more complete explanation, see Notes below.

`decimal.Decimal` type is not supported.

Parameters:

values : *array_like, shape(N,)*

Input cash flows per time period. By convention, net "deposits" are negative and net "withdrawals" are positive. Thus, for example, at least the first element of `values`, which represents the initial investment, will typically be negative.

Returns:

out : *float*

Internal Rate of Return for periodic input values.

NumPy-Financial

Η συνάρτηση irr υπολογίζει τον εσωτερικό βαθμό απόδοσης μιας επένδυσης.

Έστω επένδυση $100 με μελλοντικές ταμειακές ροές(σε σταθερή περίοδο) $39,$59,$55,$20.

Υποθέτοντας ότι η τελική αξία ισούται με το 0 βλέπουμε πως η αρχική επένδυση των $100 μας απέδοσε συνολικά $173 λόγω του ανατοκισμού και των περιοδικών ταμειακών ροών .Το «μέσο» επιτόκιο της επένδυσης ισούται με $\frac{0,73}{4}$

$$-100+\frac{39}{1+r}+\frac{59}{(1+r)^2}+\frac{55}{(1+r)^3}+\frac{20}{(1+r)^4}=0$$

NumPy-Financial

Ουσιαστικά ο εσωτερικός βαθμός απόδοσης (IRR) μετράει την απόδοση μιας μακροχρόνιας απόδοσης εξισώνοντας την παρούσα αξία των μελλοντικών ταμειακών ροων(πλέον τελικής αξίας) με την αγοραία αξία της επένδυσης

- ```
  import numpy_financial as npf
  ```
- ```
  round(npf.irr([-100, 39, 59, 55, 20]), 5)
  ```

Ο εσωτερικός βαθμός απόδοσης ισούται με 28%

Σε επόμενες διαφάνειες θα επανέλθουμε στον εσωτερικό βαθμό απόδοσης μέσω του IRR rule , και θα παρουσιάσουμε μια πιο αλγοριθμική προσέγγιση αυτου του δείκτη.

NumPy-Financial

npv

`numpy_financial.npv(rate, values)`

Returns the NPV (Net Present Value) of a cash flow series.

Parameters:

rate : *scalar*

The discount rate.

values : *array_like, shape(M,)*

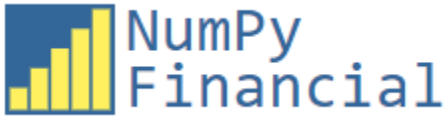
The values of the time series of cash flows. The (fixed) time interval between cash flow "events" must be the same as that for which `rate` is given (i.e., if `rate` is per year, then precisely a year is understood to elapse between each cash flow event). By convention, investments or "deposits" are negative, income or "withdrawals" are positive; `values` must begin with the initial investment, thus `values[0]` will typically be negative.

Returns:

out : *float*

The NPV of the input cash flow series `values` at the discount `rate`.

NumPy-Financial



Θεωρείστε μια πιθάνή επένδυση των $40.000 με χρηματοροές $5.000 $8.000 $12.000 $30.000 στο τέλος κάθε έτους με επιτόκιο 8% ανά έτος. Ποια είναι η καθαρή αξία της επένδυσης ;

- import numpy as np
- import numpy_financial as npf
- rate, cashflows = 0.08, [-40.000, 5.000, 8.000, 12.000, 30.000]
- npf.npv(rate, cashflows).round(5)

NumPy-Financial

Εναλλακτικά θα μπορούσαμε να ορίσουμε ξεχωριστά την αρχική επένδυση , με τις μελλοντικές χρηματοροές. Όπως για παράδειγμα:

```
initial_cashflow = cashflows[0]
cashflows[0] = 0
np.round(npf.npv(rate, cashflows) + initial_cashflow, 5)
```

NumPy-Financial

pmt

`numpy_financial.pmt(rate, nper, pv, fv=0, when='end')`

Compute the payment against loan principal plus interest.

Given:

- a present value, *pv* (e.g., an amount borrowed)
- a future value, *fv* (e.g., 0)
- an interest *rate* compounded once per period, of which there are
- *nper* total
- and (optional) specification of whether payment is made at the beginning (*when* = {'begin', 1}) or the end (*when* = {'end', 0}) of each period

Return:

the (fixed) periodic payment.

NumPy-Financial

Πόση είναι η μηνιαία πληρωμή ώστε να αποπληρώσουμε δάνειο των $200.000 σε 15 έτη με ετήσιο επιτόκιο 7.50% ;

```
import numpy_financial as npf
npf.pmt(0.075/12, 12*15, 200000)
```

Η μηνιαία δόση για την αποπληρωμή του δανείου ισούται με $1,854.024

NumPy-Financial

nper

`numpy_financial.nper(rate, pmt, pv, fv=0, when='end')`

Compute the number of periodic payments.

`decimal.Decimal` type is not supported.

Parameters:

rate : *array_like*
Rate of interest (per period)

pmt : *array_like*
Payment

pv : *array_like*
Present value

fv : *array_like, optional*
Future value

when : *{{'begin', 1}, {'end', 0}}, {string, int}, optional*
When payments are due ('begin' (1) or 'end' (0))

NumPy-Financial

Με την συνάρτηση nper υπολογίζουμε το πλήθος των περιοδικών πληρωμων

Υποθέστε πως διαθέτετε μόνο $150/μήνα για την αποπληρωμή ενός δανείου. Πόσο καιρός απαιτείται ώστε να αποπληρώσετε ένα δάνειο των $8.000 με 7% ετήσιο επιτόκιο;

```
import numpy as np
import numpy_financial as npf
np.round(npf.nper(0.07/12, -150, 8000), 5)
```

Βλέπουμε πως απαιτούνται περίπου 64 μήνες