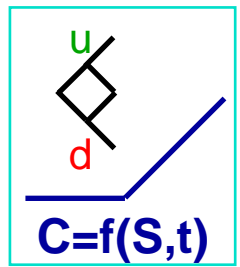


# 1 Introduction

**Jens Carsten Jackwerth**

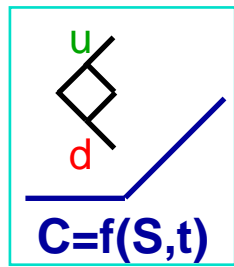
**University of Konstanz**

**[jens.jackwerth@uni-konstanz.de](mailto:jens.jackwerth@uni-konstanz.de)**



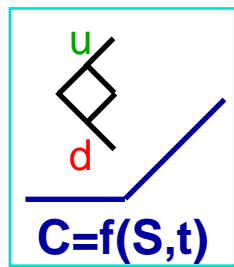
# Lecturer

- **Eva Isakeit**
- **Email: [eva.isakeit@uni-konstanz.de](mailto:eva.isakeit@uni-konstanz.de)**
- **Appointments via Zoom**
- **Master's student in Mathematical Finance**



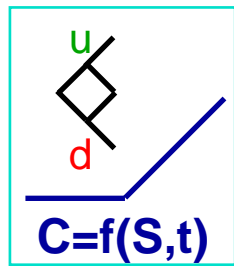
# This Course

- **English**
- **Keep a journal. It is a prerequisite to pass this course**
  - Document your progress, programs, questions, learning experience
  - Helps you whenever you come back to topics covered in this course
  - ~10 pages
- **Slides and coding examples are available on homepage. The password will be announced in the first meeting**
- **Class**
  - Interactive online meeting: Q&A and revision of assignment
  - Weekly upload of new material on the homepage (substantive video + assignment)



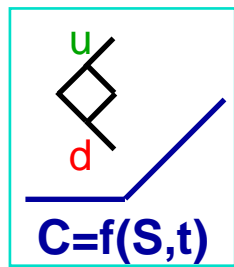
# How to get the 3 ECTS

- Rules to pass the course
- Weekly Assignments
  - Up to 10 points per homework
  - You need to have on average 5 points
  - Assignment not handed in = 0 points
  - Homework you cannot present in class = 0 points
- Send to [eva.isakeit@uni-konstanz.de](mailto:eva.isakeit@uni-konstanz.de) until Saturday 8pm
- Journal
  - to be handed in at semester end



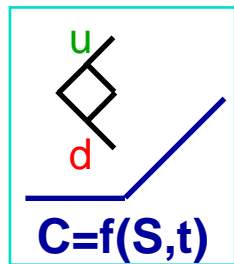
# Table of Contents

1. **Introduction**
2. **Script Files**
3. **Functions**
4. **Data Import and Data Manipulation**
5. **Graphical Illustrations**
6. **CAPM**
7. **Logical Expressions**
8. **Loops**
9. **European Option Pricing**
10. **American Option Pricing**
11. **Optimization**
12. **NPV Optimization**



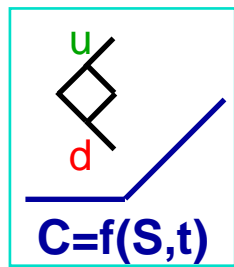
## Literature

- ❑ Sweigart, A., 2019. Automate The Boring Stuff With Python, 2Nd Edition. No Starch Press, Incorporated.
- ❑ Liang, Y., 2013. Introduction To Programming Using Python. Boston: Pearson.
- ❑ Downey, A., 2015. Think Python: How To Think Like A Computer Scientist, 2Nd Edition. O'Reilly Media.



# Why Python for Finance

- What is Python?
- Why do we need it?
- Comparison to other languages
- Shift from MATLAB to Python



## Motivation

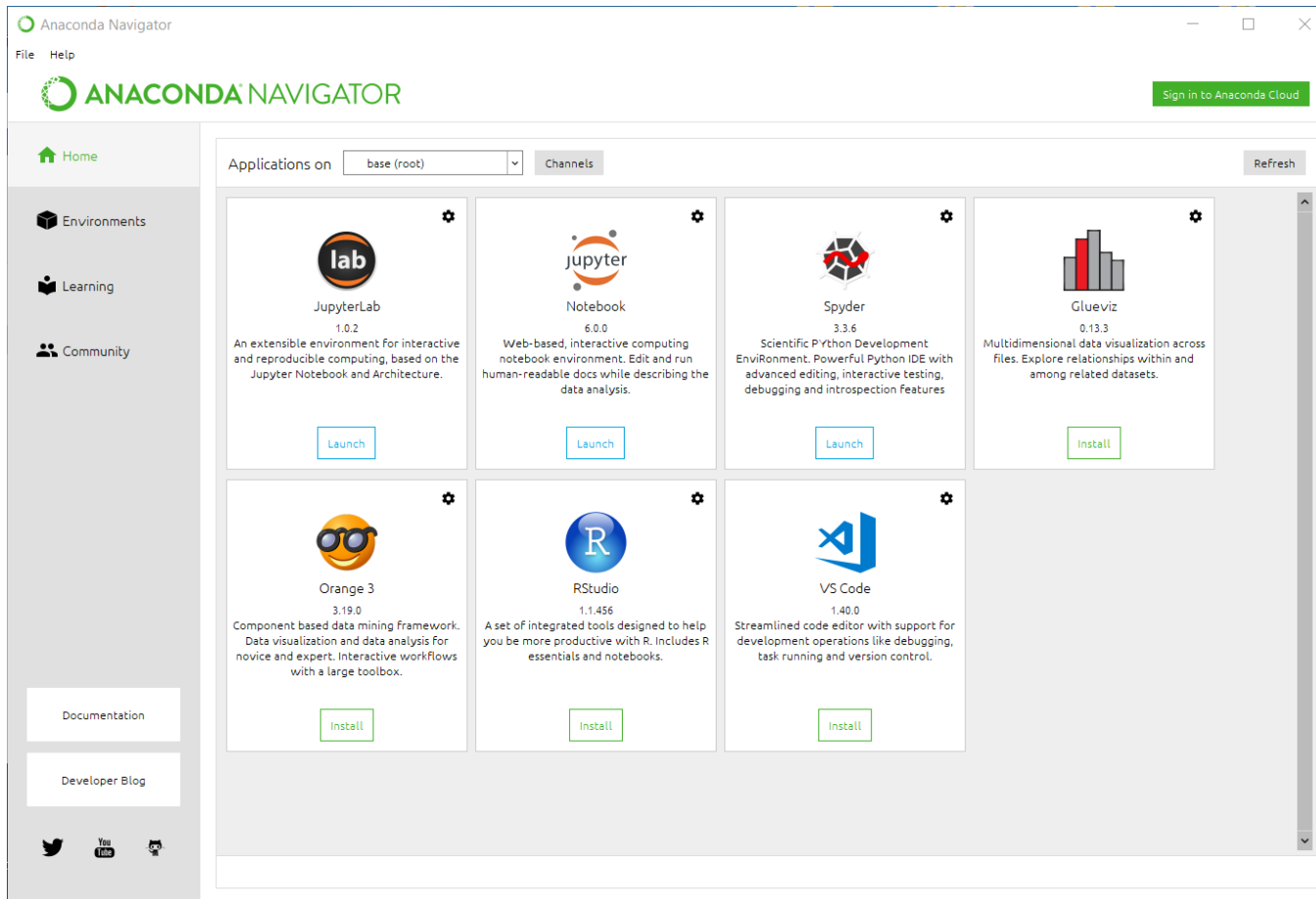
- “Python is a general-purpose, versatile and popular programming language. It is great as a first programming language because it is concise and easy to read and it is also a good language to have in any programmers stack as it can be used for everything from web development to software development and scientific applications”
- Python is becoming de facto standard in finance industry
- Easy and high-level introduction to programming
- Major feature is its ecosystem, e.g., libraries and tools
- Might be useful for Bachelor thesis



# Starting Python


$$C=f(S,t)$$

<https://www.anaconda.com/>



Anaconda Navigator

File Help

ANACONDA NAVIGATOR [Sign in to Anaconda Cloud](#)

Home

Environments

Learning








Community

Documentation

Developer Blog

Twitter YouTube GitHub

Applications on  Channels [Refresh](#)

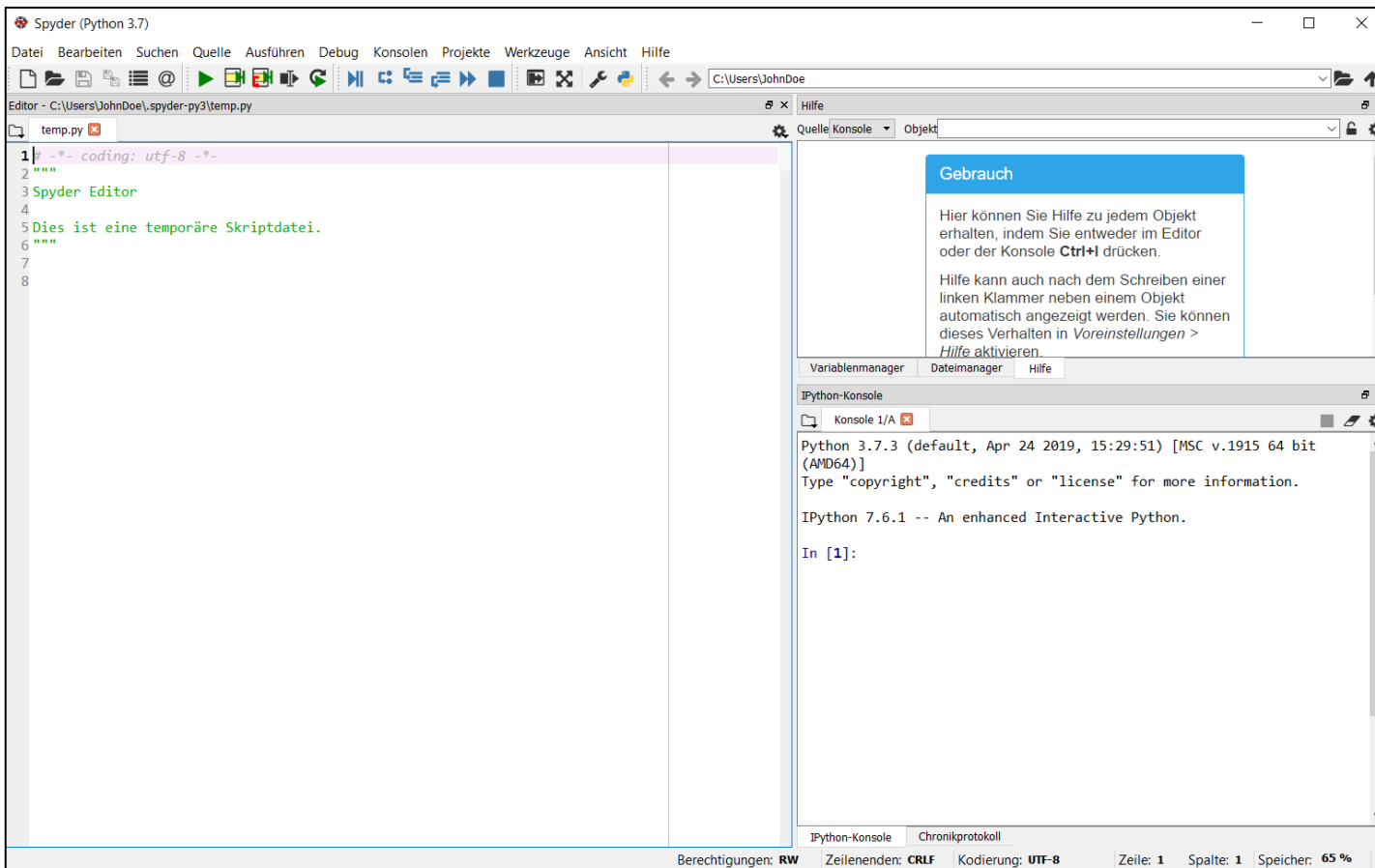
 <b>JupyterLab</b> 1.0.2 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. <a href="#">Launch</a>	 <b>Jupyter Notebook</b> 6.0.0 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. <a href="#">Launch</a>	 <b>Spyder</b> 3.3.6 Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. <a href="#">Launch</a>	 <b>Glueviz</b> 0.13.3 Multidimensional data visualization across files. Explore relationships within and among related datasets. <a href="#">Install</a>
 <b>Orange 3</b> 3.19.0 Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. <a href="#">Install</a>	 <b>RStudio</b> 1.1.456 A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. <a href="#">Install</a>	 <b>VS Code</b> 1.40.0 Streamlined code editor with support for development operations like debugging, task running and version control. <a href="#">Install</a>	

# Starting Python II



$$C=f(S,t)$$

## Launch Spyder



Spyder (Python 3.7)

Datei Bearbeiten Suchen Quelle Ausführen Debug Konsolen Projekte Werkzeuge Ansicht Hilfe

Editor - C:\Users\JohnDoe\spyder-py3\temp.py

```

1 #-*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 Dies ist eine temporäre Skriptdatei.
6 """
7
8

```

Hilfe

Quelle: Konsole Objekt

**Gebrauch**

Hier können Sie Hilfe zu jedem Objekt erhalten, indem Sie entweder im Editor oder der Konsole **Ctrl+I** drücken.

Hilfe kann auch nach dem Schreiben einer linken Klammer neben einem Objekt automatisch angezeigt werden. Sie können dieses Verhalten in *Voreinstellungen* > *Hilfe aktivieren*.

Variablenmanager Dateimanager Hilfe

IPython-Konsole

Konsole 1/A

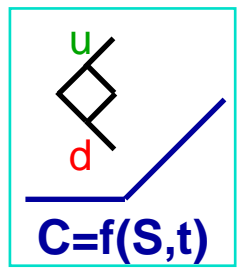
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]:

IPython-Konsole Chronikprotokoll

Berechtigungen: RW Zeilenenden: CRLF Kodierung: UTF-8 Zeile: 1 Spalte: 1 Speicher: 65 %



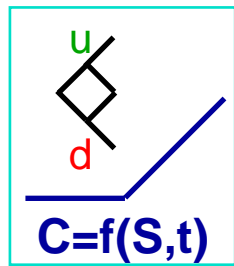
# Calculation of NPV in Python I

- A “Great Deal“: Assume the following cashflows:

year	0	1	2	3
cashflow	-100	-50	30	200

- Assume that  $r=0.1$
- The NPV formula is given by

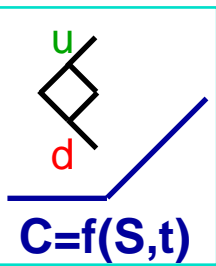
$$NPV = C_0 + \frac{C_1}{1+r} + \frac{C_2}{(1+r)^2} + \frac{C_3}{(1+r)^3}$$



# Operations

- **The order of operations is given as:**
  1. **Terms inside parentheses ( ) or brackets [ ]**
  2. **Functions in Python**
  3. **Exponents and roots**
  4. **Multiplication and division**
  5. **Addition and subtraction**
  
- **Attention:  $a^b \rightarrow a^{**}b$       or      `pow(a, b)`**

# Calculation of NPV in Python II



Spyder (Python 3.7)

Datei Bearbeiten Suchen Quelle Ausführen Debug Konsolen Projekte Werkzeuge Ansicht Hilfe

Editor - C:\Users\JohnDoe\Downloads\PyFi01.py

```

1 """NPV-calculation"""
2
3 npv = -100-50/(1+0.1)+30/(1+0.1)**2+200/(1+0.1)**3
4
5 print(npv)
6
7
8
9
10
11
12
13
14
15
16

```

Name	Typ	Größe	Wert
npv	float	1	29.601803155522134

IPython-Konsole

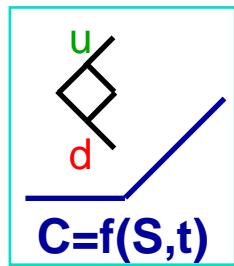
```

In [2]: runfile('C:/Users/JohnDoe/Downloads/PyFi01.py', wdir='C:/Users/JohnDoe/Downloads')
29.601803155522134

In [3]:

```

Berechtigungen: RW Zeilenenden: CRLF Kodierung: UTF-8 Zeile: 121 Spalte: 4 Speicher: 79 %



## Calculation of NPV in Python III

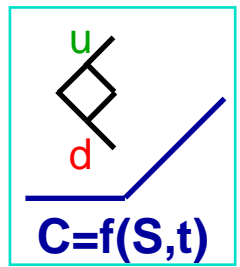
- **Example:** Now assume that you can sell your machine in year 4 for 100. How does the NPV change?

```
npv = -100 - 50/(1+0.1) + 30/(1+0.1)**2 + 200/(1+0.1)**3
```

```
npv += 100/(1+0.1)**4
```

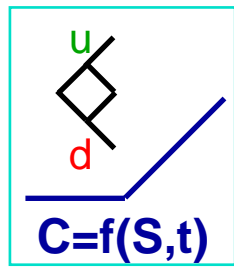
```
# npv = npv + 100/(1+0.1)**4
```

```
= 97.90314869202919
```



# Assignments

- **The new NPV is 97.903 (in comparison to the old one: 29.601)**
- **The variable npv will be re-defined and the value of 29.601 will be lost**



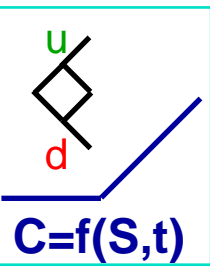
# Numpy

- **numpy is the main library for scientific computing with Python**

```
import numpy as np
```

- **Use it as the main library for any calculations with vectors and matrices**





# Defining Variables I

```
#Variables you want to define always stand on the left-hand side of the equal sign, the following  
# commands save the values of cashflows in a numpy-array (works here like a row vector)
```

```
cashflows = np.array([-100, -50, 30, 200])
```

```
print(cashflows)
```

```
#To create a column vector we use the option of a two-dimensional array
```

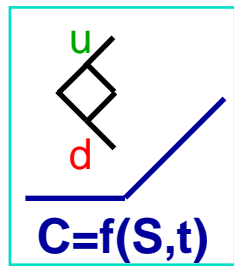
```
cashflows_column = np.array([[-100], [-50], [30], [200]])
```

```
print(cashflows_column)
```

```
#you will recognize that every new row will be build with new brackets: "[ ]"
```

```
dim2 = np.array([[1, 2], [3, 4]])
```

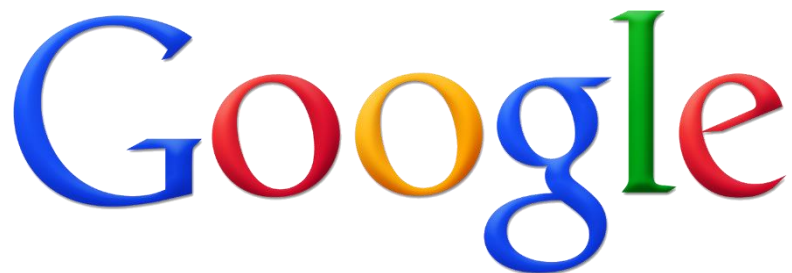
```
print(dim2)
```



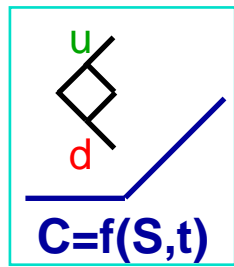
## Getting Help

```
#Another useful command is help  
help(np.array)
```

### Other resources:

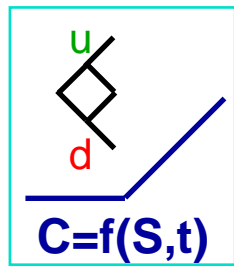


 stackoverflow



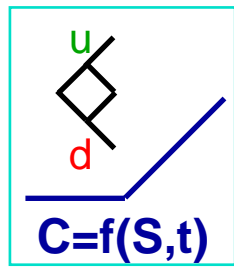
## Defining Variables II

- ❑ To extend a row vector with one entry, use a comma:  
`np.array([a, b])`
- ❑ To extend a column vector with one entry, use [brackets]:  
`np.array([a], [b])`
- ❑ Define a matrix with `np.array([[a, b], [c, d]])`. Watch out:  
Dimensions must agree for a matrix!



## Defining Variables III

- ❑ Variable names can consist of letters, numbers and “\_”, but should not start with a number. Do not use names which already exist in Python. Variable names are case sensitive
- ❑ Variables store values that can be re-used in a different part of the program
- ❑ All written after a #-sign in the same line will not be considered from Python
- ❑ Alternatively you can use `““““...””””` to make comments for more than one line



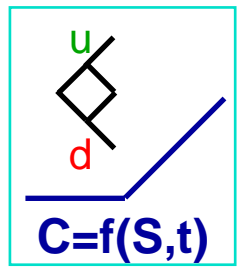
# Calling Variables for Calculations

- The variable `cashflows` is a row vector (remember that we have redefined it). Therefore `cashflows @ discount_r` is a vector multiplication

```
cashflows= np.array([-100, -50, 30, 200])
r = 0.1
discount_r = np.array([1, 1/(1+r)**1, 1/(1+r)**2, 1/(1+r)**3])

#Define the variable NPV to store the value of the computation:
npv = cashflows @ discount_r

print(npv)
```



## In-Class Exercise

1. Create the following matrices in Python:

$$A = \begin{pmatrix} 2 & 5 & 7 \\ 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 3 & 3 \\ -1 & 0 \\ 2 & 4 \end{pmatrix}$$

2. What is the result of the matrix multiplication:

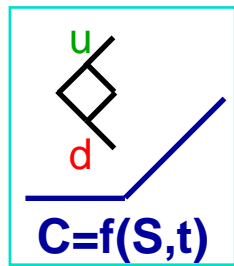
$$X = A @ B$$

3. What is the result for the following command:

$$X[1, 0]$$

4. What is the result of the matrix multiplication?

$$X = A * B.T$$

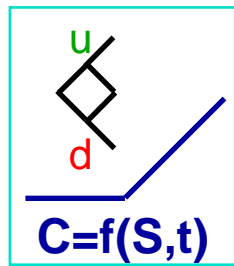


## Exercise I

1. **Open Python and examine all windows. Make sure everything is in the correct order as explained above. Define a folder for your lecture examples and exercises in a new folder. Change the current directory of Python to the folder you defined for the lecture examples**
2. **Calculate the NPV of a bond which matures at  $t=4$ , pays an annual coupon of 10 beginning in year 1, and you will receive in year  $t=4$  the last coupon and an additional payment (value at maturity) of 100. Assume that  $r=0.1$**
3. **What is the value of the bond if  $r=0.08$ ?**

$$\text{Bond price } P = \frac{CP}{1+r} + \frac{CP}{(1+r)^2} + \frac{CP}{(1+r)^3} + \frac{CP + \text{value at maturity}}{(1+r)^4}$$

*You should define variables for these calculations as described in the lecture*



## Exercise II

4. Now assume there is uncertainty and you estimate that you only receive the last payment (value at maturity) with a probability of 0.6. What is the value of the bond now?
5. What might be the advantage of using variables?
6. Create the following matrix in Python:

$$p = \begin{pmatrix} 1 & -0.4 & 0.6 \\ -0.4 & 1 & 0 \\ 0.6 & 0 & 1 \end{pmatrix}$$