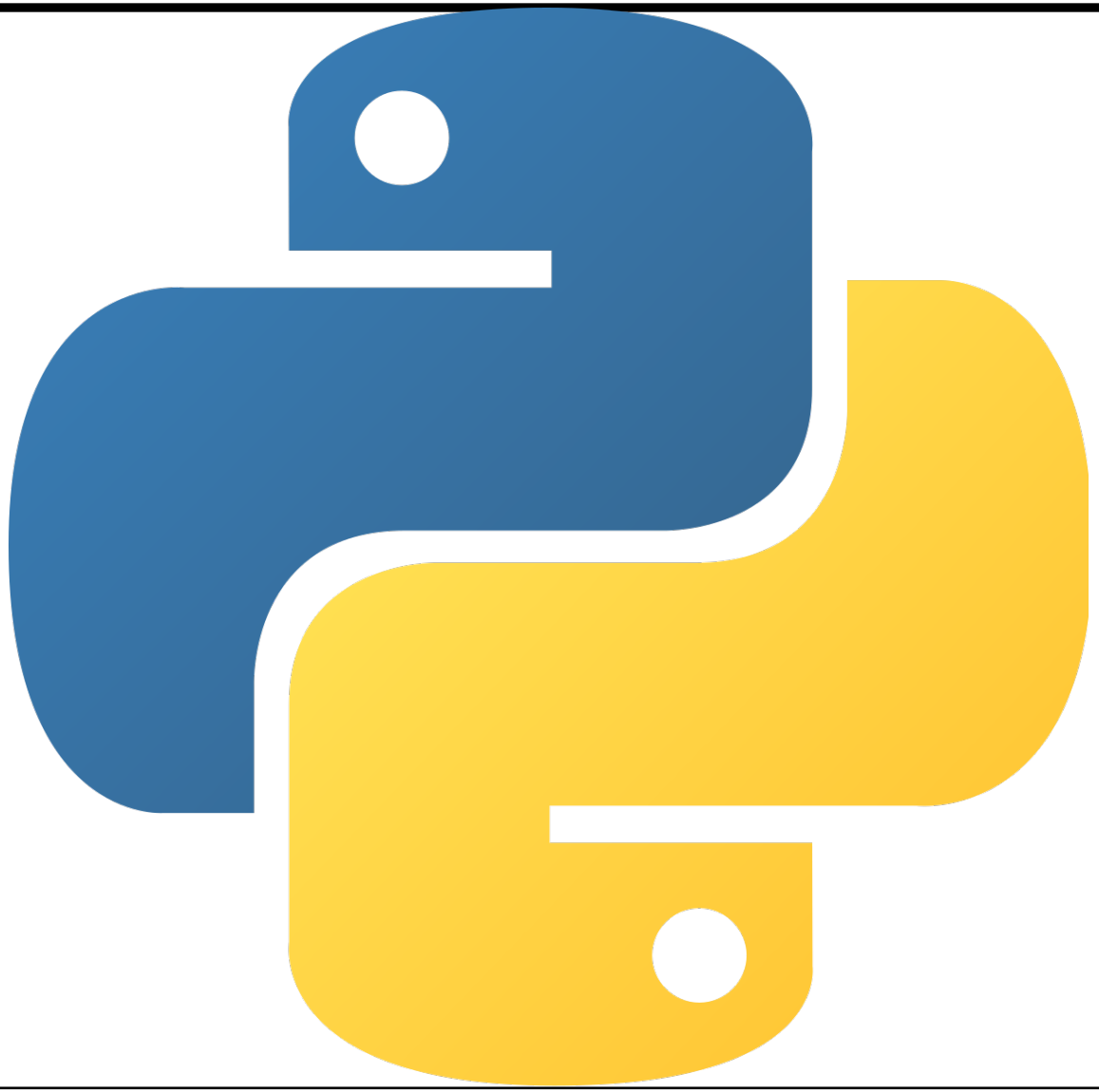
PYTHON PROGRAMMERING



Föreläsning 2

DAGENS AGENDA

- Gästföreläsning AI Act och GDPR
- Installera Jupyter Notebook: kommandon och environment
- Pythons grunder:
  - Variabler
  - Datatyper
  - Operatorer
  - Casting
  - Lister

FÖRRA FÖRELÄSNING

- Introduktion kursen: studieplanering och kursmaterial
- Bekantade med datorens terminal: Powershell
- Installerade Python med Anaconda (minicona)
- Aktiverade Virtual Environments
- Installerade Visual Studio Codes (VSC)
- Aktiverade Python i VSC

JUPYTER NOTEBOOK

- Jupyter Notebook är en webbapplikation som låter en skapa och dela dokument som innehåller live-kod, ekvationer, visualiseringar och berättande text.
- Användningsområden inkluderar data cleaning, dataanalys och transformation, numerisk simulering, statistisk modellering, datavisualisering, maskininlärning och mycket mer.
- Ofta använder vi notebooks för att testa små kodsnuttar innan vi integrerar dem med våra hela skript.
- Ibland utvecklar vi även bara i notebooks när vi vill analysera och visualisera vårt data på olika sätt, då en notebook är mycket förlåtande då den sparar variabler mellan sina ”celler”.
- För att kunna köra notebooks i din browser eller i VSC behöver du ha ett extra paket installerat i din conda environment.

https://www.dataquest.io/blog/jupyter-notebook-tutorial/

CONDA ENVIRONMENTS MED JUPYTER NOTEBOOK

1. **För att automatiskt registrera ett nytt environment** för att använda med notebooks kan man installera ett paket i sitt `base` environment.

- Detta behöver du bara göra en gång!
- I terminalen:

```
conda activate base
```

```
conda install -c conda-forge notebook nb_conda_kernels
```

CONDA ENVIRONMENTS MED JUPYTER NOTEBOOK

2. Nästa gång du vill skapa ett nytt conda environment:

- Skapa ditt environment och installera `python` och `ipykernel`.
- Välj själv namn istället för `NEW_ENV`.
- Eftersom vi installerar `ipykernel` så kommer `nb_conda_kernels` automatiskt att registrera det

```
conda create --name NEW_ENV python=3.9 ipykernel
```

CONDA ENVIRONMENTS MED JUPYTER NOTEBOOK

3. Aktiverat din NYA environment:

- FÖRSTA GÅNGEN du aktiverar din miljö ska du inte glömma att installera alla requirements.
- Se till att du står i samma mapp (folder) som din `requirements.txt` fil finns. Använd `cd` för att ändra mapp du står i.
- Du måste också ha din environment aktiverad. Du aktiverar den med:

```
conda activate NEW_ENV
```

- När du har aktiverat din environment så kan du dubbelkolla att din `requirements.txt` fil finns i mappen genom att titta på listan som dyker upp när du kör:

`ls` eller `dir`

- När du ser att din `requirements.txt`-fil finns så kör du

```
pip install -r requirements.txt
```

CONDA ENVIRONMENTS MED JUPYTER NOTEBOOK

4. Altenativ 1: Skapa en notebook i VSCode

- I VSC, skapa en ny notebook med `ctrl + shift + p`
- Sök på `Jupyter: Create New Blank Notebook` och välj det
- I övre högra hörnet finns en knapp för att välja kernel. Tryck på den och sök på din `NEW_ENV`
- Det kan hände du måste starta om VSC för att se din kernel `NEW_ENV`

4. Alternativ 2: Använda ditt nya environment med jupyter notebooks i webbläsaren

- Du kan nu prova att starta en notebook server från ditt base conda environment (terminalen) med:

```
jupyter notebook
```

- Om du skapar en ny notebook så kommer du kunna välja en kernel med namnet Python `[conda env: NEW_ENV]`

JUPYTER NOTEBOOK KOMMANDON

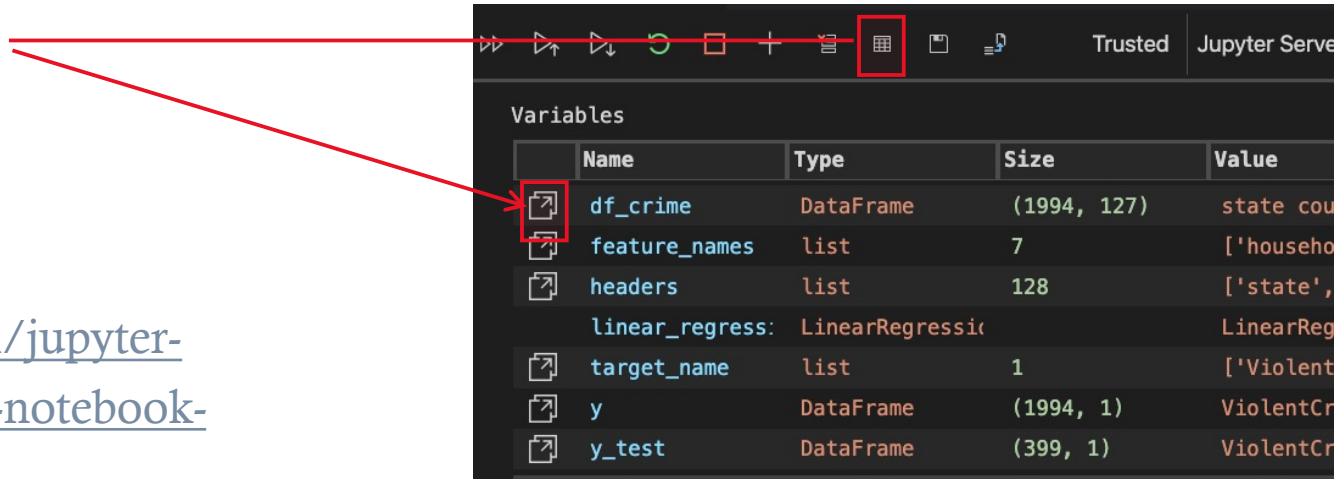
- Enter – redigera läge
- Ecs – kommando läge
- Shift+enter – run cell
- Alt+enter – run cell och infoga ny cell under
- Gömma output - dubbeltklicka vänster sida av output eller använd ;

När man är i kommando läge (Esc):

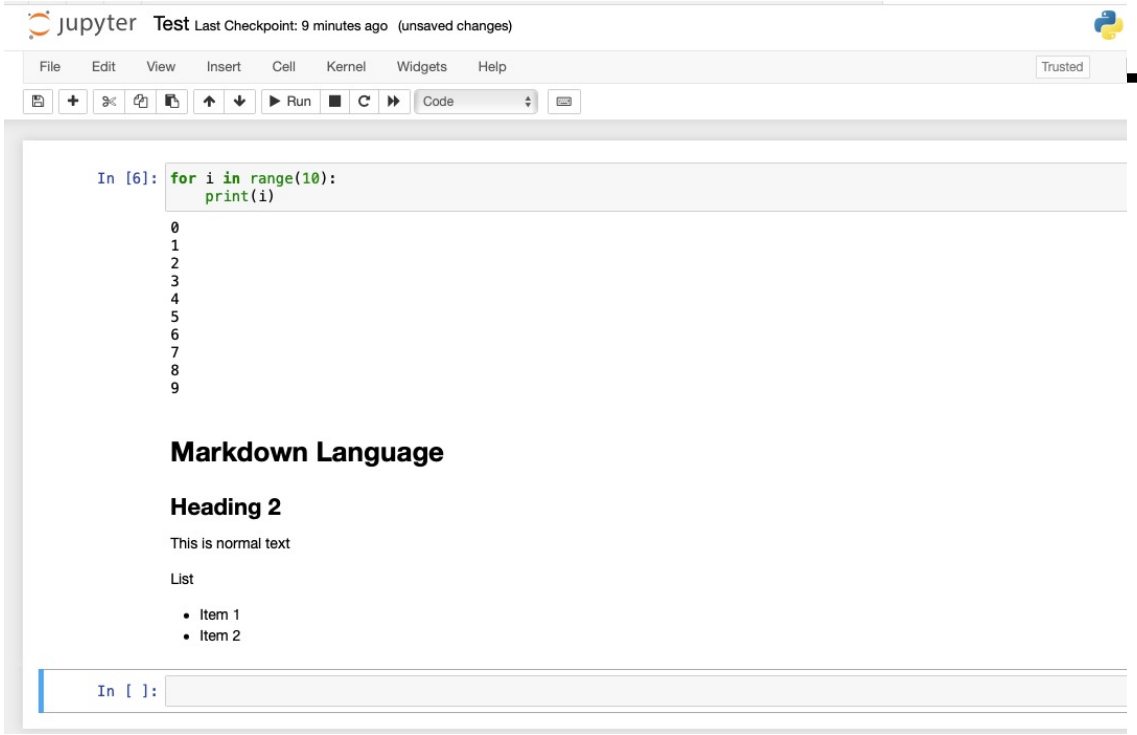
- a – ny cell över
- b – ny cell under
- x – klipp ut vald cell
- c – kopiera vald cell
- v - klistra in cell
- d+d – radera cell
- z – ångra raderad cell
- s – spara
- m – ändra cell till Markdown
- y – ändra cell till kod
- o – göm output/vis output

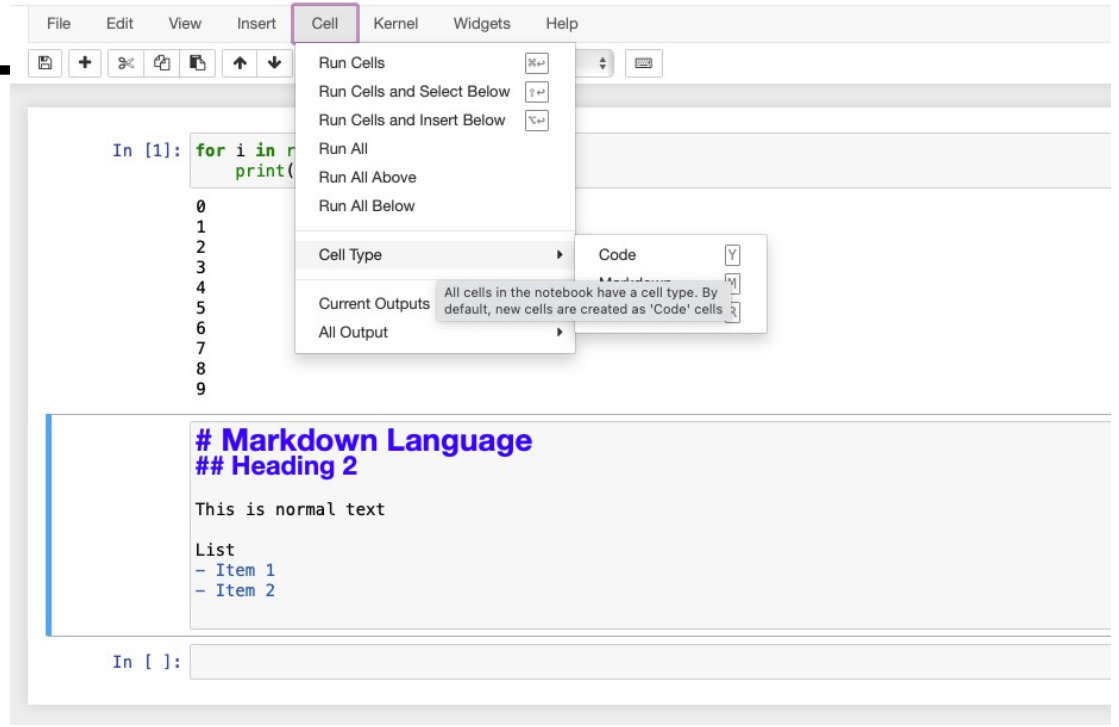
JUPYTER NOTEBOOK

- För att se alla variabler man har skapat



- Hjälp om Notebook i VSC:
- https://code.visualstudio.com/docs/python/jupyter-support#_intellisense-support-in-the-jupyter-notebook-editor



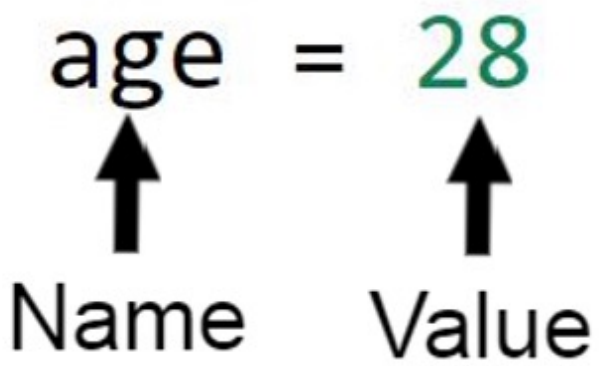


MARKDOWN LANGUAGE CHEAT SHEET

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

VARIABLER

- En variabel är en behållare som har en typ och ett namn
- Används för att lagra data som kan nås senare
- Grundsten i programmering
- Vi refererar till variabler efter namnet
- I Python blir variabelns typ bestämd när variabeln får ett värde. Kallas dynamic type checking. Samma som i R, men inte vanligt i andra programmeringsspråk, där anges ofta typ.
- Vi använder print för att skriva ut värdet på en variabel



NAMNGIVNING VARIABLER

- Vanligast att använda små bokstäver och underscore
- Ge variablerna beskrivande och meningsfulla namn

```
average_age = 26 #correct
Average_age = 26 #wrong
averageAge = 26 #wrong
average_Age = 26 #wrong
```

OPERATORER

- Tilldelningsoperatoren =
  - Tilldelar högersidan till variabeln på vänstersidan. Evalueras till sist.
- Matematiska operatorer +, -, *, /, %, ()
  - fungerar som man förväntar och följer normal evalueringsordning.
- Tilldelning och matematisk operator +=, -=, *=, /=, %=
  - Utför matematisk operator med variabeln på högersidan och variabeln på vänstersidan

```
one = 1
two = 1 + 1
three = 1 + 1 * 2
four = (1 + 1) * 2
two += 2 # equals 4
```

DATATYPER – NUMBERS

- Tre tal-typer:
  - Heltal – int
  - Floating-point number – float
  - Complex (ingår inte i kursen)
- Tal med med decimal . blir automatisk float
- Python hanterar ändring mellan int och float automatisk

```
four = 4
print(type(four)) # prints <class 'int'>
price = 9.99
print(type(price)) # prints <class 'float'>
quarter = 3/four # quarter is a float
print(quarter) # prints 0.75
```

DATATYPER – STRING

- Strings för att behandla text
- Encoded med UTF-8
- Skapar string med äntingen enkla eller dubbla citat-tecken: ’string’, ”string”
- Vissa matematiska operatorer fungerar ihop med strings som +. Betyder concat på slutet av string

```
hello = "Hello"
print(type(hello)) # prints <class 'str'>
world = "World"
hello_world = hello + world # concatenation
print(hello_world) # prints HelloWorld
```

DATATYPER - BOOLEAN

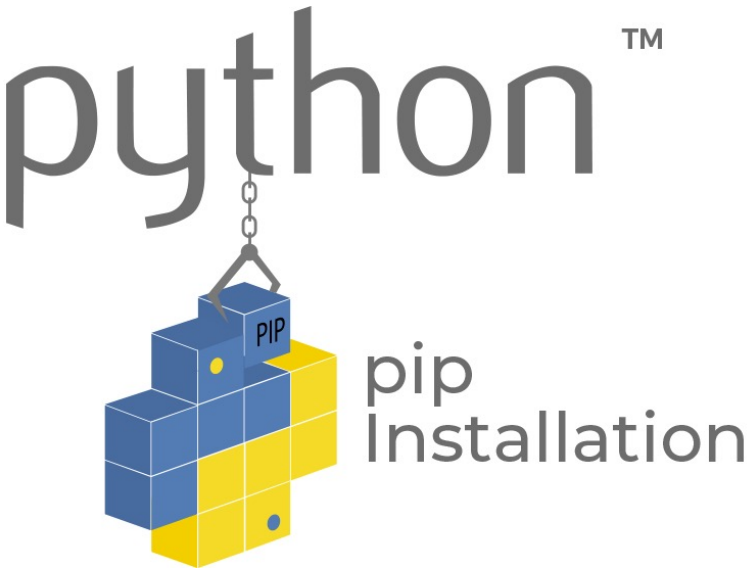
- Boolean: för att veta om ett uttryck är True eller False
- Relationell operator == returnerar boolean True om det stämmer och False annars
- Som i R har True också värdet 1 och False värdet 0

```
is_true = True
is_false = 1 == 2
# is false is a bool with value False
print(1 == 2) # prints False
```

CASTING

- Casting för att konvertera datatyper
- Används med
  - `str()`
  - `int()`
  - `float()`

```
one = "1"
two = 1 + int("1") # two is a int with value 2
twelve = one + str(two)
# twelve is a str with value "12"
```



PIP

- `pip` är ett pakethanteringssystem skrivet i Python som används för att installera och hantera programvarupaket.
- Det installeras automatiskt med Python, som gör installationen av externa bibliotek till Python mycket enkelt.
- `pip` är något du kommer använda mycket som pythonanvändare.

RECAP

- Gästföreläsning AI Act och GDPR
- Installera Jupyter Notebook: kommandon och environment
- Pythons grunder:
  - Variabler
  - Datatyper
  - Operatorer
  - Casting
  - Lister

NÄSTA LEKTION

- Control flow statements (if-else)
- Konventioner: Main-metod, scripting, funktioner typing, konventioner, docstrings
- Läsa WTP s. 41-52
- https://www.python.org/dev/peps/pep-0008/