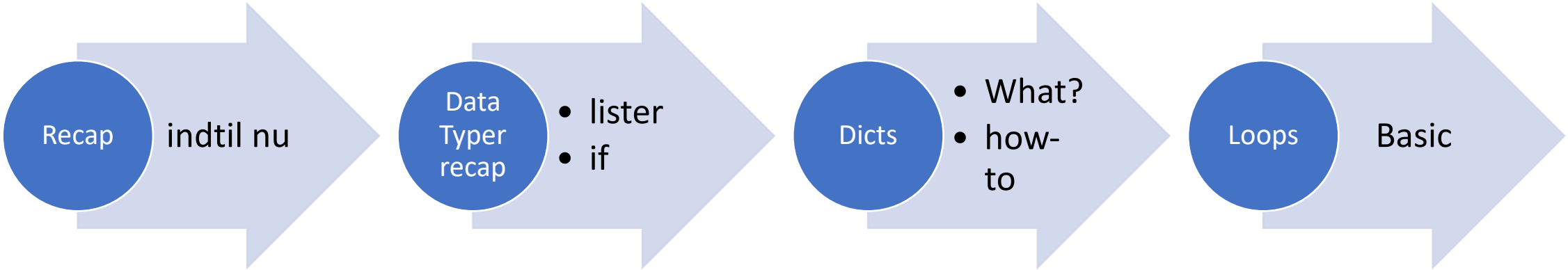
EVU Python LESSON III

Dagens program



Velkomst

kursets forløb

24/8

31/8

07/9

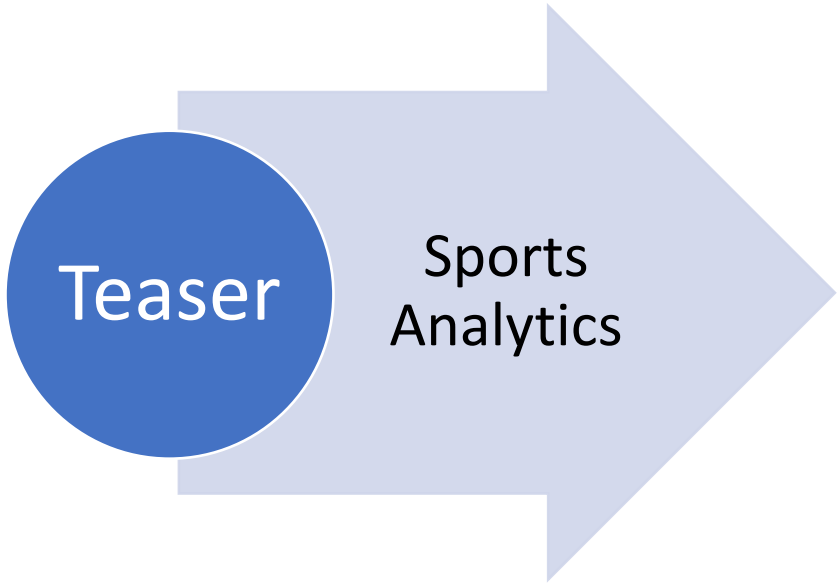
14/9

21/9

28/9

5/10

Her forlader vi bogen

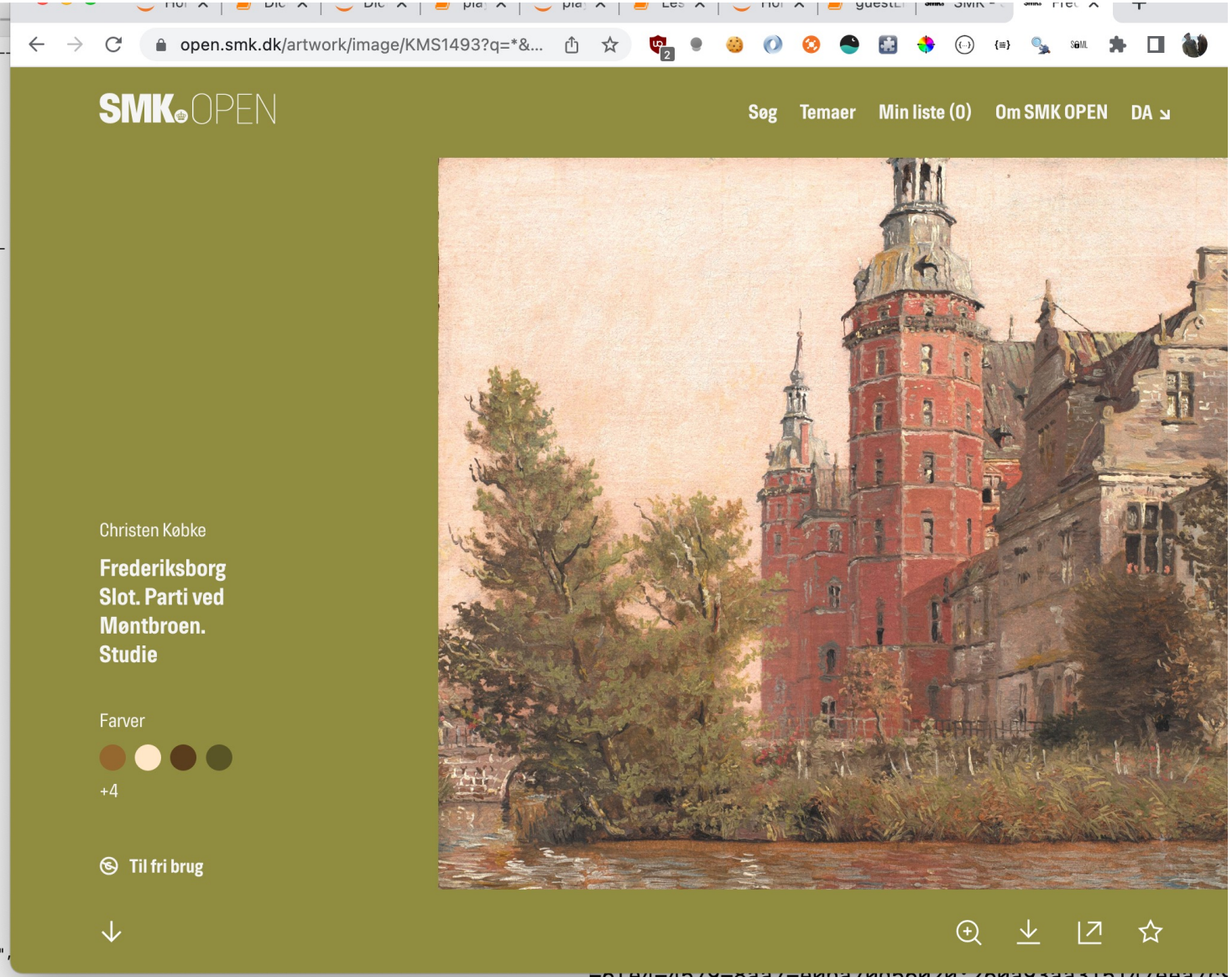



```
{
  "eventId": 10,
  "subEventName": "Shot",
  "tags": [
    {
      "id": 401
    },
    {
      "id": 201
    },
    {
      "id": 1215
    },
    {
      "id": 1802
    }
  ],
  "playerId": 12536,
  "positions": [
    {
      "y": 33,
      "x": 87
    },
    {
      "y": 0,
      "x": 0
    }
  ],
  "matchId": 2499725,
  "eventName": "Shot",
  "teamId": 1613,
  "matchPeriod": "1H",
  "eventSec": 283.438159,
  "subEventId": 100,
  "id": 178442509
},
```

Teaser

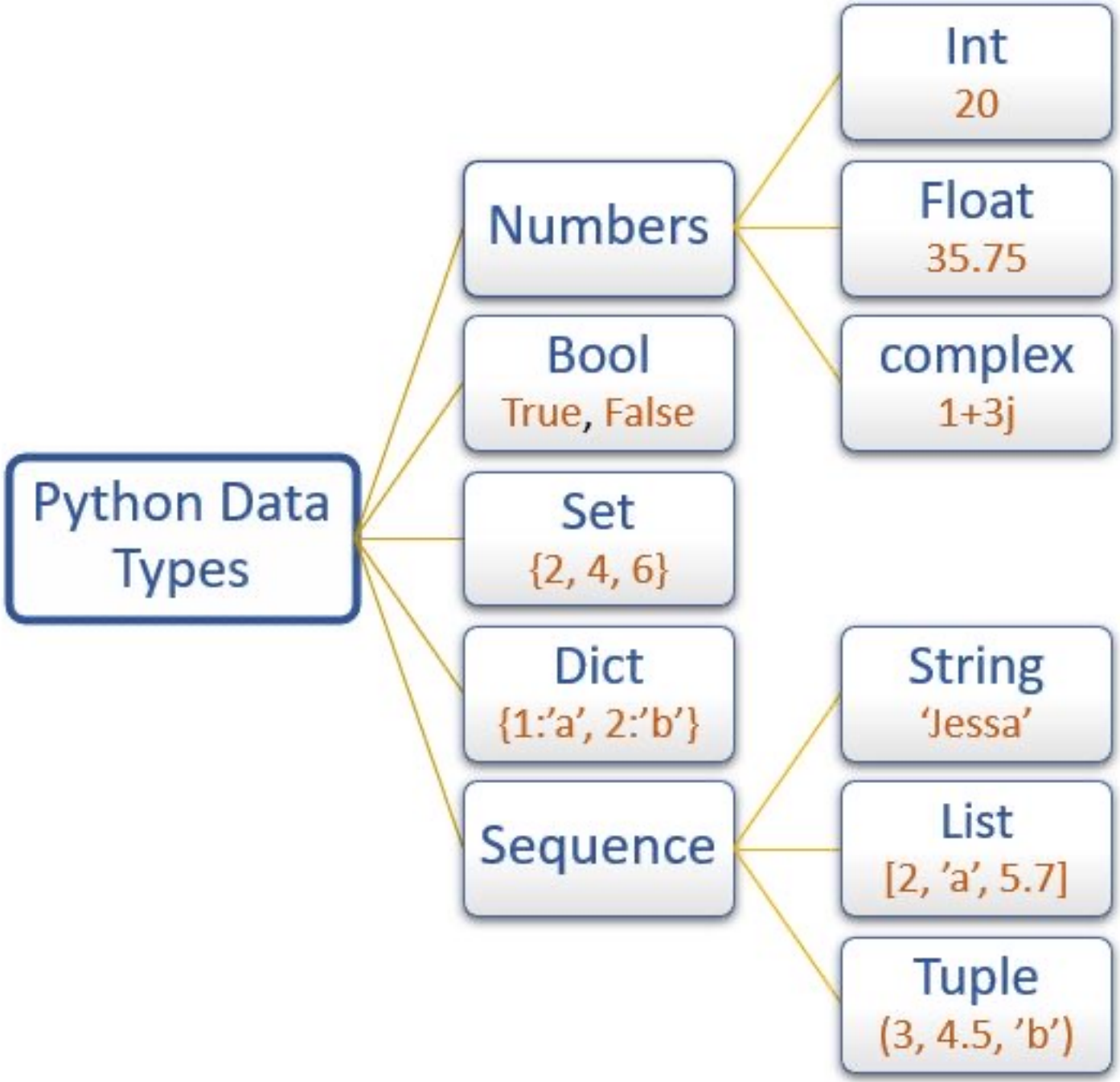
Art

```
| ~ @ 10153-62 (thor)
| => cat out2.json | python -mjson.tool
{
    "offset": 0,
    "rows": 24,
    "found": 1,
    "items": [
        {
            "id": "1170022756_object",
            "created": "2020-03-21T13:41:18Z",
            "modified": "2022-01-31T19:46:27Z",
            "responsible_department": "Den Kongelige Maleri-
            "acquisition_date": "1894-01-01T00:00:00Z",
            "acquisition_date_precision": "1894-12-31",
            "content_subject": [
                "Frederiksborg Slot"
            ],
            "dimensions": [
                {
                    "notes": "240 x 270 mm",
                    "part": "Netto",
                    "type": "h\u00f8jde",
                    "unit": "centimeter",
                    "value": "24"
                },
                {
                    "notes": "240 x 270 mm",
                    "part": "Netto",
                    "type": "bredde",
                    "unit": "centimeter",
                    "value": "27"
                },
                {
                    "part": "Brutto",
                    "type": "h\u00f8jde",
                    "unit": "centimeter",
                    "value": "34.5"
                },
                {
                    "part": "Brutto",
                    "type": "bredde",
                    "unit": "centimeter",
                    "value": "37.5"
                },
                {
                    "part": "Brutto",
                    "type": "dybde",
                    "unit": "centimeter",
                    "value": "5.3"
                }
            ],
            "documentation": [
                {
                    "title": "Christen K\u00f8bke 1810-1848",
                    "author": "unknown",
                    "notes": "heri: Kasper Monrad, \"K\u00f8bke p\u00e5 Frederiksborg i 1835\", p. 198f, fig. 127, kat. 99.",
```



Datatyper

Overview



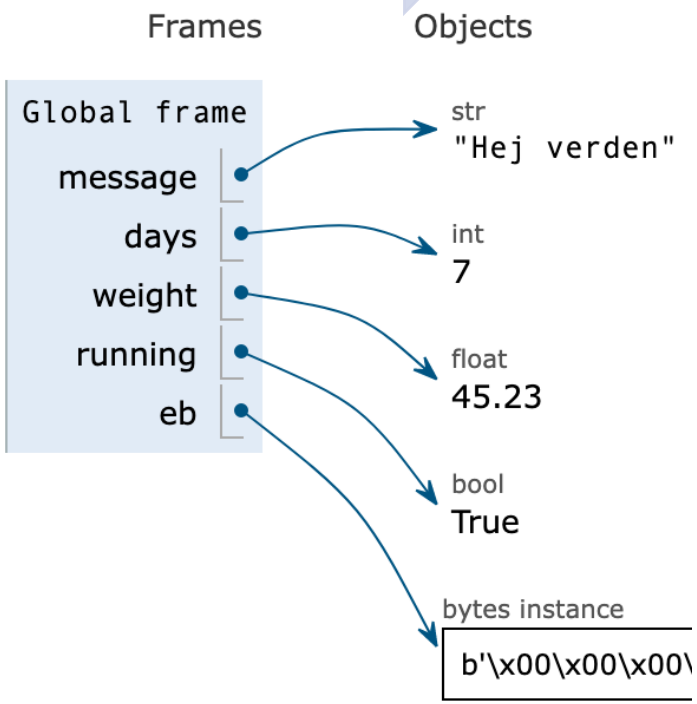
A Python variable is a reserved memory location to store values.

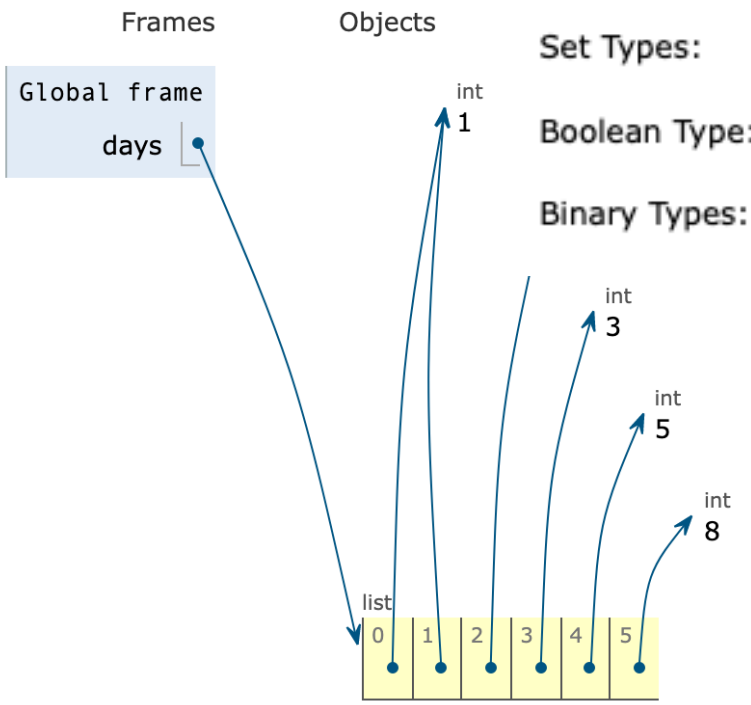
Every value in Python has a datatype.

Datatyper

”Alle”

A Python variable is a reserved memory location to store values.
Every value in Python has a datatype.





Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

Datatyper

Dicts – hvad?

- A dictionary in Python is a collection of key-value pairs.
- Each key is connected to a value.
- You can use a key to access the value associated with that key.
- A key's value can be a number, a string, a list, a dictionary or any object

Simple start

```
alien={'color':'green','points':5}
```

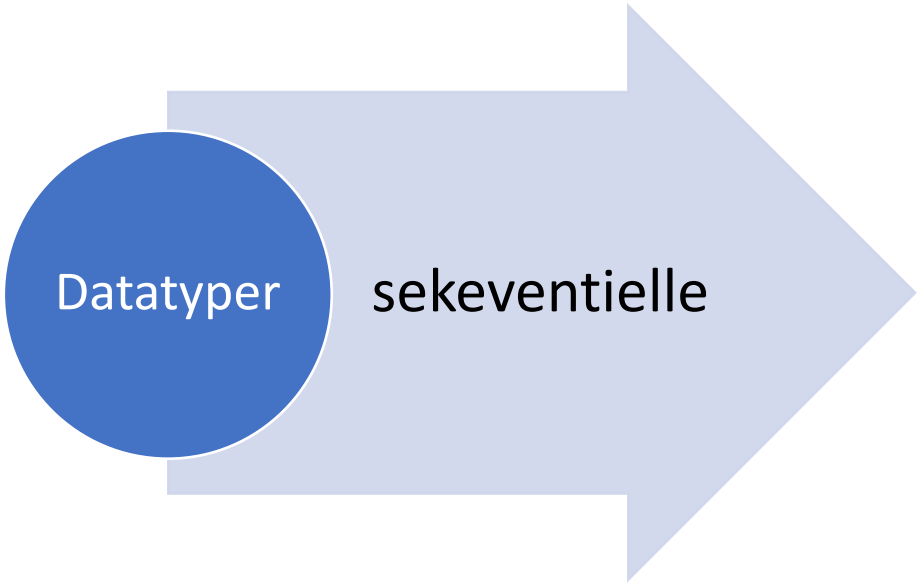
```
In [14]: alien={'alienID':12, 'info':{'color':'green','points':5}}
```

```
In [21]: alien={'alienID':12, 'info': {'color':'green','points':5, 'weapons':['sword','knife']}, 'hist': {'rd1':12, 'rd2':4}}
```

```
In [23]: pp.pprint(alien)

{'alienID': 12,
 'history': {'round1': 12, 'round2': 4},
 'info': {'color': 'green', 'points': 5, 'weapons': ['sword', 'knife']}}
```

"Pretty Print" json,xml



Creating an empty list l=[]	Creating an empty Tuple t=()	Creating a set a=set() b=set(a)	Creating an empty dictionary d={}

List	Tuple	Set	Dictionary
List is a non-homogeneous data structure that stores the elements in single row and multiple rows and columns	Tuple is also a non-homogeneous data structure that stores single row and multiple rows and columns	Set data structure is also non-homogeneous data structure but stores in single row	Dictionary is also a non-homogeneous data structure which stores key value pairs
List can be represented by [ ]	Tuple can be represented by ( )	Set can be represented by { }	Dictionary can be represented by { }
List allows duplicate elements	Tuple allows duplicate elements	Set will not allow duplicate elements	Set will not allow duplicate elements and dictionary doesn't allow duplicate keys.
List can use nested among all	Tuple can use nested among all	Set can use nested among all	Dictionary can use nested among all
List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)

Datatyper

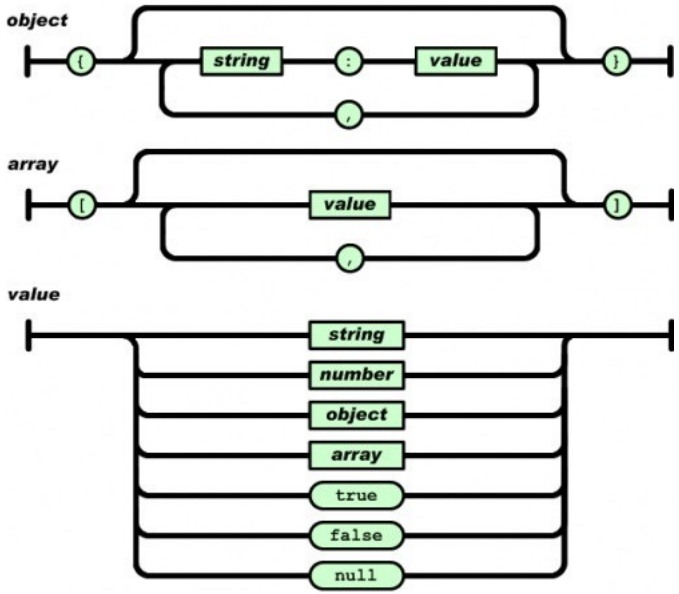
Json og dicts

JSONLint - The JSON Validator


```
{
  "id": "e00fce68c573b4acca2089ce",
  "type": 150,
  "location": 216,
  "latitude": 56.1632767,
  "longitude": 10.2105122,
  "location_name": "Nørrebrogade",
  "city": "Aarhus",
  "country": "Denmark",
  "roles": [
      4
  ],
  "permissions": [],
  "tags": [
      "Randersvej"
  ]
}
```


Cityflow – REST API Data Format

- ”The above command returns JSON structured like this”
- JSON – JavaScript Object Notation



Datatyper

Dicts in action I

- Creating dicts
- Accessing keys and/or values
- Adding items
- Updating items
- Printing (format)
- Complicated dicts
  - Lists in dicts
  - Dicts in dicts
  - List of Dicts

Python Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing the a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Datatyper

Dicts in action I

- Accessing Values in a Dictionary
- Adding New Key-Value Pairs
- Starting with an Empty Dictionary
- Modifying Values in a Dictionary
- Removing Key-Value Pairs
- A Dictionary of Similar Objects
- Looping Through All Key-Value Pairs
- Looping Through All the Keys in a Dictionary
- Looping Through a Dictionary's Keys in Order
- Looping Through All Values in a Dictionary

A List of Dictionaries

A List in a Dictionary

A Dictionary in a Dictionary

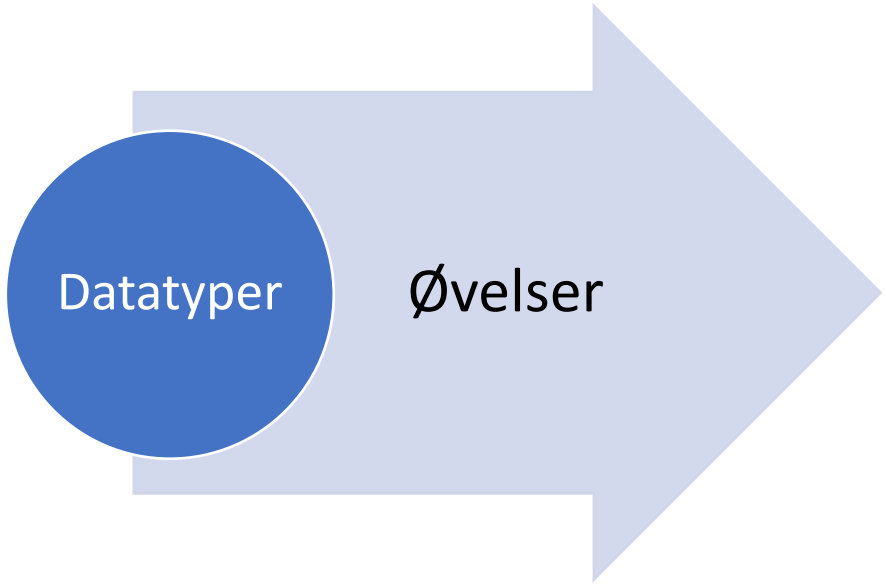
Datatyper

Dicts in action III

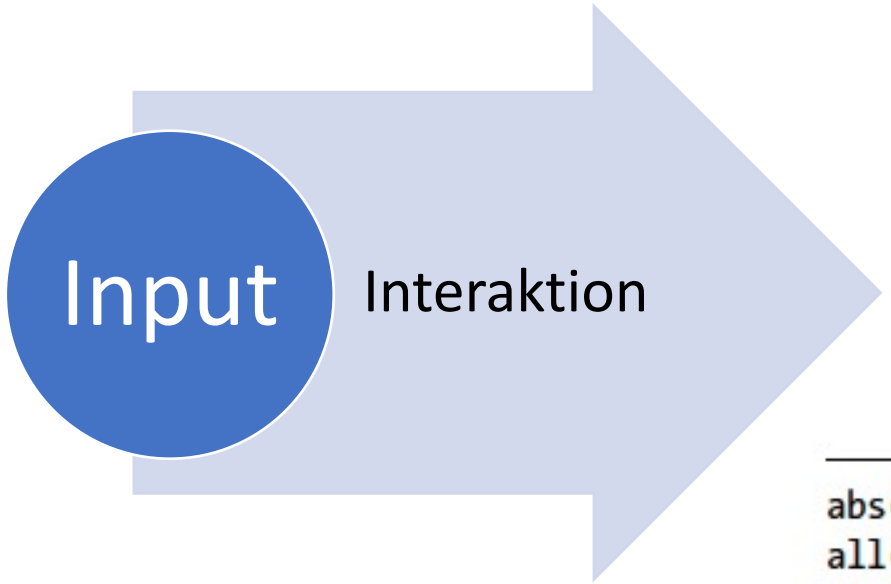
Looping

```
for k,v in myDict.items():
    print(f'{k} -> {v}')
```

```
player_1 -> {'fn': 'Kurtx', 'ln': 'Vernerx', 'bd': '12-04-2000'}
player_2 -> {'fn': 'Ahmed', 'ln': 'Boduz', 'bd': '11-02-2002'}
player_3 -> {'fn': 'Victor', 'ln': 'Hugoo', 'bd': '11-07-2004'}
```

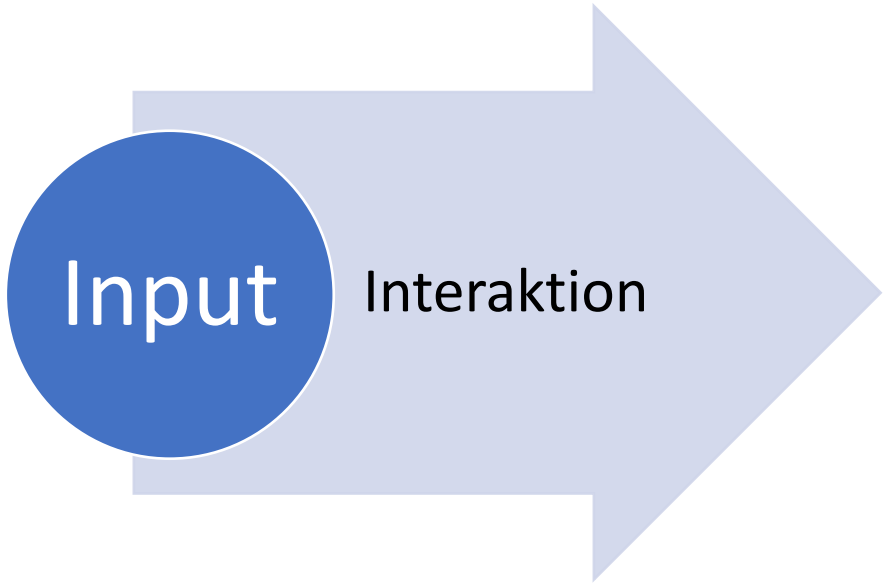


- 6-3 Ordbog med 5 <udtryk> <betydning>
- 6-4 Loop igennem k,v
- 6-8 List of Pets (key: Skully, {kat,”kurt”})



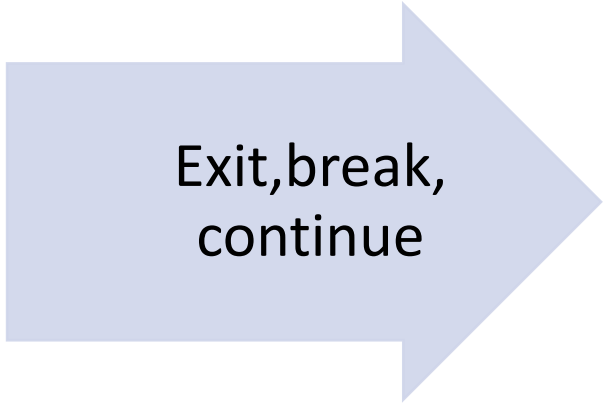
The input() function pauses your program and waits for the user to enter some text.

abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

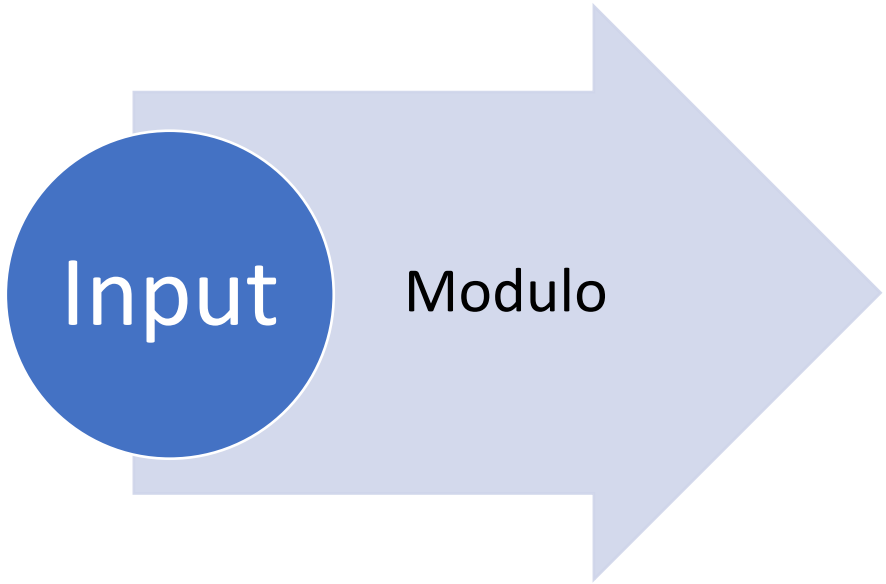



```
running=True
while running:
    choice=input("Whats up?(Q for quit)")
    if choice.lower()=="q":
        running=False
```

```
import re
counter=0
while counter < len(data):
    if (re.search("bmw",data[counter],re.I)):
        print("Got a bmws",data[counter])
        break
    counter +=1
```

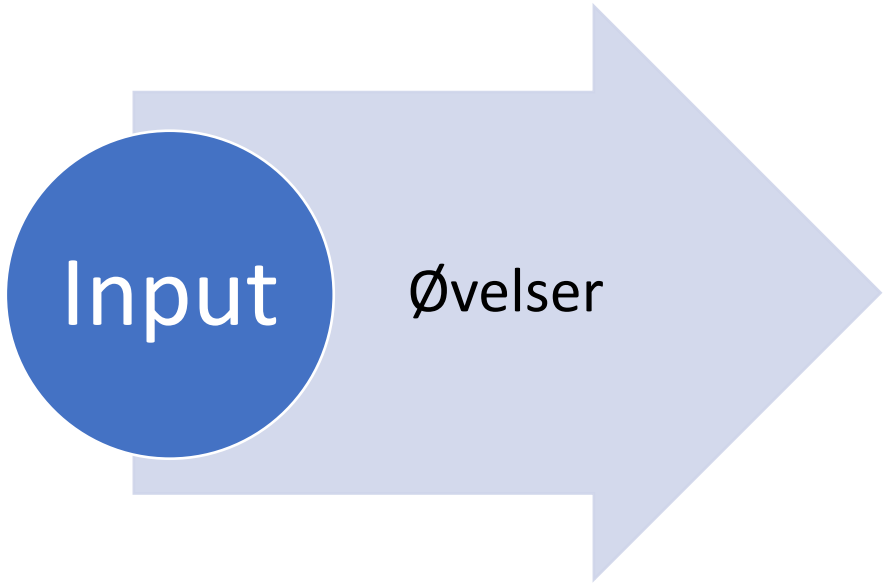



```
while counter < len(data)-1:
    counter +=1
    if not(re.search("bmw",data[counter],re.I)):
        continue
    print("Got a bmws",data[counter])
```




```
In [3]: counter = 0
        while counter < 100:
            counter = counter + 1
            if counter % 4 == 0:
                print("\tDO SOMETHING ELSE")
            else:
                print("DO NORMAL")
```

```
DO NORMAL
DO NORMAL
DO NORMAL
        DO SOMETHING ELSE
DO NORMAL
DO NORMAL
DO NORMAL
        DO SOMETHING ELSE
DO NORMAL
DO NORMAL
DO NORMAL
        DO SOMETHING ELSE
DO NORMAL
DO NORMAL
DO NORMAL
```



- 7-1 Book car
- 7-2 Seats in restaurent (+8 then wait)
- 7-1 Modify
  - create a list of cars (from cars.csv)
  - loop through list and take car to new list
  - present customer with car
  - create outer-while-loop where you ask for name

- 7-5 Ticket-loop (-3 gratis, +3 er 10, +12 er 15)