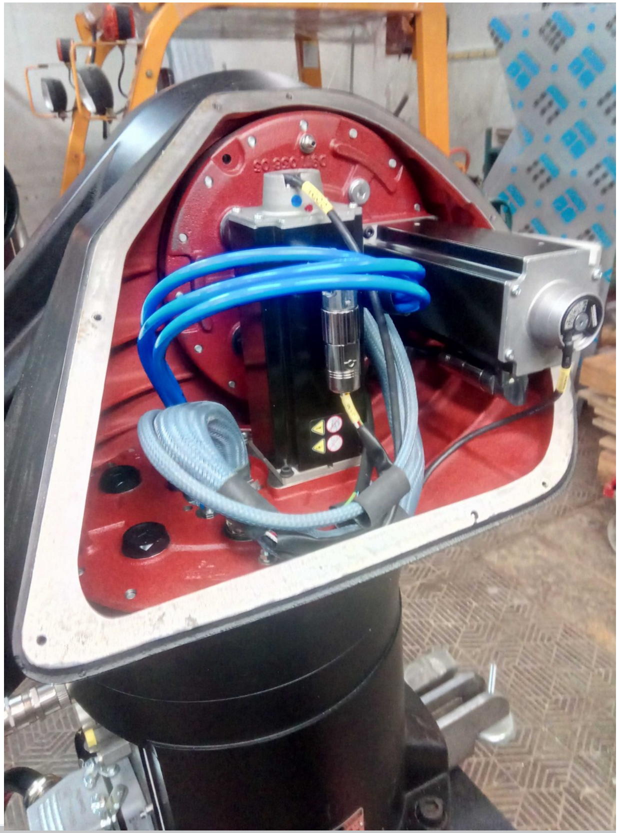
Staubli RX160 industrial robot arm with Python

```
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
RDK = Robolink()
#RDK.setRunMode(RUNMODE_MAKE_ROBOTPROG)

import csv

PROGRAM_NAME = "test"


# Get the robot item by name:
robot = RDK.Item('Staubli-RX160', ITEM_TYPE_ROBOT)
RDK.ProgramStart(PROGRAM_NAME, "", "", robot)
robot.setSpeedJoints(200)
robot.setSpeed(200)
#robot.ConnectSafe(robot_ip='192.168.0.254', max_attempts=5, wait_connection=4, callback_ab
# state = robot.Connect()
# print(state)
robot.setJoints([0,0,0,0,0,0])
```


STÄUBLI

Staubli RX160

MODEL	RX160
Maximum payload	30 kg, 66.1 lb (34 kg, 74.9 lb under conditions)
Nominal payload	20 kg, 44 lb
Reach (between axis 1 and 6)	1710 mm, 67,3 in
Number of degrees of freedom	6
Repetability – ISO 9283	± 0.05 mm
Stäubli series controller	CS8C
Weight	248 kg, 546.7 lb
MAXIMUM SPEED	
Axis 1	200°/s
Axis 2	200°/s
Axis 3	255°/s
Axis 4	315°/s
Axis 5	360°/s
Axis 6	870°/s
Maximum speed at load gravity center	10.3 m/s
Maximum inertia axis 5	4 $kg.m^2$
Maximum intertia Axis 6	1 $kg.m^2$
Brakes	



FOREARM CONNECTIONS	
Pneumatic	2 solenoid valves 5/2-way (compressed air) monostable 1 direct line between the base and the forearm or 2 solenoid valves 3/2-way (vacuum). 1 direct line between the base and the forearm.
Electrical	1 female 19-contact socket (7 twisted pairs including 2 shielded, 3 power contacts)
Cleanroom standard ISO 14644-1	5
Protection class according to EN 60529	IP65 / IP67

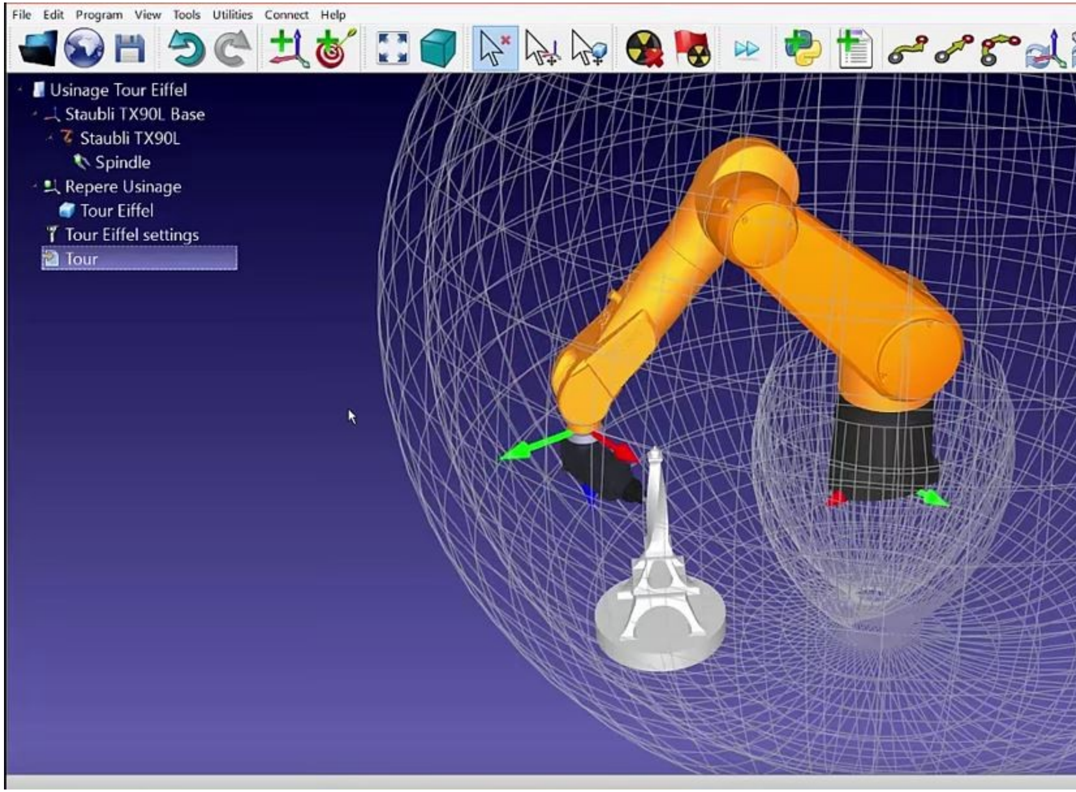
- Staubli – very precise, very fast, mechanics and connectors made in-house – but their toolchain and documentation is less accesible
- Kuka, ABB etc – heavier payload, less precise, more open specifications and tools
- No common robot control/programming standard: **ABB** has its **RAPID** programming language. **Kuka** has **KRL** (Kuka Robot Language). **Comau** uses **PDL2**, **Yaskawa** uses **INFORM** and **Kawasaki** uses AS. **Fanuc** robots use **Karel**, **Stäubli** robots use **VAL3** and **Universal Robots** use **URScript**.
- (but: **ROS-Industrial** is a BSD-licensed software development program designed to create a Unified Robot Description Format (URDF) for industrial robots)

How robots are controlled in practice

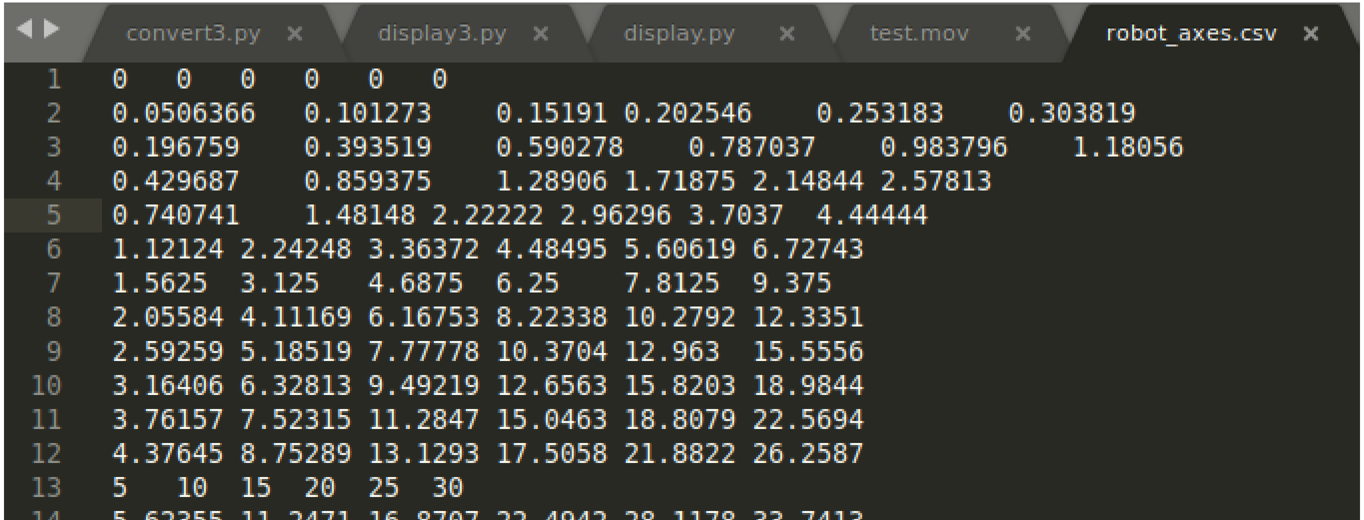
- Teach pendant – movements are set up in place
- Robotics simulators – virtual robot is programmed off-line (Gazebo, RoboDK, Delmia, Octopuz, Robotmaster etc.)
- Robot machining (CAD to motion)
- Industrial software – $$$

RoboDK

- Simulator
- Post-processor
- API
- Machining/3D printing
- Live control (on Win)
- 3k $
- 30 day evaluation



- Our (suddenly urgent) use case: stop motion animation movie w/ Dragonframe and Maya
- Camera path → robot joint coordinates



0	0	0	0	0	0
0.0506366	0.101273	0.15191	0.202546	0.253183	0.303819
0.196759	0.393519	0.590278	0.787037	0.983796	1.18056
0.429687	0.859375	1.28906	1.71875	2.14844	2.57813
0.740741	1.48148	2.22222	2.96296	3.7037	4.44444
1.12124	2.24248	3.36372	4.48495	5.60619	6.72743
1.5625	3.125	4.6875	6.25	7.8125	9.375
2.05584	4.11169	6.16753	8.22338	10.2792	12.3351
2.59259	5.18519	7.77778	10.3704	12.963	15.5556
3.16406	6.32813	9.49219	12.6563	15.8203	18.9844
3.76157	7.52315	11.2847	15.0463	18.8079	22.5694
4.37645	8.75289	13.1293	17.5058	21.8822	26.2587
5	10	15	20	25	30

RoboDK Python API modules

- robolink – bridges RoboDK and Python (item manipulation)
  https://robodk.com/doc/en/PythonAPI/robolink.html
- robodk – robotics toolbox (pose transforms, etc.)
  https://github.com/RoboDK/RoboDK-API/tree/master/Python

- RoboDK then post-processes the programs into VAL 3 code, which can be ftp’d to the controller.

```python
# https://robodk.com/doc/en/RoboDK-API.html
from robolink import *    # API to communicate with RoboDK
from robodk import *      # robodk robotics toolbox

# Any interaction with RoboDK must be done through RDK:
RDK = Robolink()

# Select a robot (popup is displayed if more than one robot
robot = RDK.ItemUserPick('Select a robot', ITEM_TYPE_ROBOT)
if not robot.Valid():
    raise Exception('No robot selected or available')


# get the current position of the TCP with respect to the re
# (4x4 matrix representing position and orientation)
target_ref = robot.Pose()
pos_ref = target_ref.Pos()
print("Drawing a polygon around the target: ")
print(Pose_2_TxyzRxyz(target_ref))


# move the robot to the first point:
robot.MoveJ(target_ref)
```

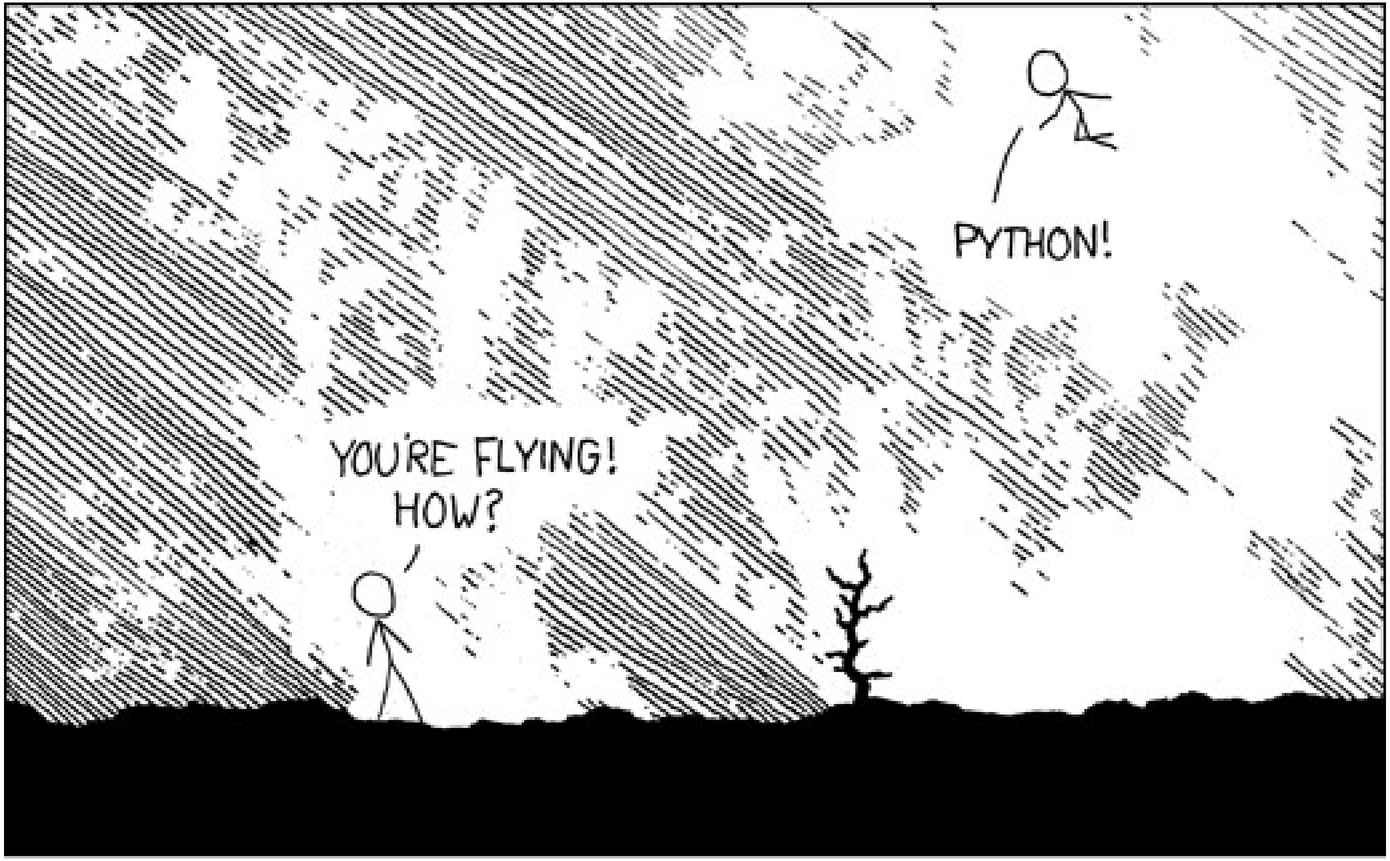
```python
# It is important to provide the reference frame and the tool fram
robot.setPoseFrame(robot.PoseFrame())
robot.setPoseTool(robot.PoseTool())
robot.setZoneData(10) # Set the rounding parameter (Also known as:
robot.setSpeed(200) # Set linear speed in mm/s

# Set the number of sides of the polygon:
n_sides = 6
R = 100

# make a hexagon around reference target:
for i in range(n_sides+1):
    ang = i*2*pi/n_sides #angle: 0, 60, 120, ...

    #-----------------------------
    # Movement relative to the reference frame
    # Create a copy of the target
    target_i = Mat(target_ref)
    pos_i = target_i.Pos()
    pos_i[0] = pos_i[0] + R*cos(ang)
    pos_i[1] = pos_i[1] + R*sin(ang)
    target_i.setPos(pos_i)
    print("Moving to target %i: angle %.1f" % (i, ang*180/pi))
    print(str(Pose_2_TxyzRxyz(target_i)))
    robot.MoveL(target_i)
```

In our case

```python
# Get the robot item by name:
robot = RDK.Item('Staubli-RX160', ITEM_TYPE_ROBOT)
RDK.ProgramStart(PROGRAM_NAME, "", "", robot)
robot.setSpeedJoints(200)
robot.setSpeed(200)
#robot.ConnectSafe(robot_ip='192.168.0.254', max_attempts=5, wait_connection=4, callback_abort=None)
# state = robot.Connect()
# print(state)
robot.setJoints([0,0,0,0,0,0])

with open("robot_axes.csv", "r") as f:
    reader = csv.reader(f, delimiter="\t")
    for i, line in enumerate(reader):
        robot.MoveJ([float(joint) for joint in line])
```

Some caveats..

- Not every ftp program works
- Not every cable is OK
- RoboDK’s post-processor for Staubli contains errors, a patch is here: https://robodk.com/forum/Thread-Staubli-CS8-connection-issue


YOU'RE FLYING!
HOW?
PYTHON!