Media Computing Project

Python and Fusion 360 API

Prof. Dr. Jan Borchers
M.Sc. René Schäfer

```
#Author-Autodesk Inc.
#Description-Etract BOM information from active design.

import adsk.core, adsk.fusion, traceback

def spacePadRight(value, length):
    pad = ''
    if type(value) is str:
        paddingLength = length - len(value) + 1
    else:
        paddingLength = length - value + 1
    while paddingLength > 0:
        pad += ' '
        paddingLength -= 1

    return str(value) + pad

def walkThrough(bom):
    mStr = ''
    for item in bom:
        mStr += spacePadRight(item['name'], 25) + str(spacePadRight(item['instances'], 15)) + str(item['volume']) + '\n'
    return mStr

def run(context):
    [illegible] = [illegible]one
    [illegible]y:
        [illegible]p = [illegible]d[illegible]co[illegible].A[illegible]li[illegible]ion.get()
        [illegible] = [illegible]p[illegible]se[illegible]nt[illegible]fa[illegible]

        product = app.activeProduct
        design = adsk.fusion.Design.cast(product)
        title = 'Extract BOM'
        if not design:
            ui.messageBox('No active design', title)
            return

        # Get all occurrences in the root component of the active design
        root = design.rootComponent
        occs = root.allOccurrences

        # Gather information about each unique component
        bom = []
        for occ in occs:
            comp = occ.component
            jj = 0
            for bomI in bom:
                if bomI['component'] == comp:
                    # Increment the instance count of the existing row.
                    bomI['instances'] += 1
                    break
                jj += 1
```

BASICS

Python

Basics

- Introduced in 1991
- Made for beginners
    - Easy to read (resembles English)
    - Simple syntax
- Interpreted language
- Large community
    - Many libraries / modules

Basics

- Indent-based coding
  - Code blocks are created by evenly indenting
- Case sensitive
  - True ≠ true
  - do_something() ≠ Do_something()
- Python 2.X and 3.X are incompatible

Variables

- Type is not defined in the code and can change during runtime

```
x = 3

x = 3.7

x = ”3”

x = True (not true as python is case sensitive)
```

If Statements

```
if condition1 and condition2:
    do_something()

if condition1 or condition2:
    do_something()

if not condition1:
    do_something()

if condition1:
    do_something()
elif condition2:
    do_something_different()
else:
    do_something_else()
```

Lists and Slices

```
nothing = [] # empty list

names = [“Adrian”, “Oliver”, “Marcel”, “Anke”]

names[1]   # -> “Oliver”

names[0:2] # -> [“Adrian”, “Oliver”]

names[1:]  # -> [“Oliver”, “Marcel”, “Anke”]

names[-1]  # -> “Anke”
```

Loops

```
names = [“Adrian”, “Oliver”, “Marcel”, “Anke”]

for name in names:
    print(name)

for index in range(len(names)):
    print(names[index])

index = 0
while index < len(names):
    print(names[index])
    index += 1
```

Try-Except

- Keeps your code alive if something unexpected happens

```
try:
    some_dangerous_code()
except ValueError:
    some_error_handling()
else:
    no_errors_occured()
finally:
    do_some_cleanup()
```

- Optional: ValueError, else & finally

Methods

```
def longest_name(names):
    if len(names) < 1:
        return “”

    result = “”
    for name in names:
        if len(name) > len(result):
            result = name

    return result

def my_function(*args, **kwargs):
    for arg in args:
        print(arg)
    print(“kwargs: ”, kwargs)

my_function(“MCP”, “Python”, arg1=“Fusion”, arg2=“Duck”)
# MCP
# Python
# kwargs: {‘arg1’: ‘Fusion’, ‘arg2’: ‘Duck’}
```

Standard Library

- Usually distributed with python
- Contains many modules
  - Can be included using **import**
- Documentation:
  - https://docs.python.org/3.7/library/index.html

Built-in Functions

- Functions which are always available
- Casting
    - int(), float(), str(), …
- Checking types
    - type(), isinstance(), …
- …

Imports

- Include other modules and packages
- Can be renamed locally
- Examples:

```
import math

import numpy as np

from Modules import MyFile
```

Guidelines

- Documentation:
  - https://www.python.org/dev/peps/pep-0008/

Guidelines

“Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live”

- John Woods

SOFTWARE

Fusion 360 API

Python within Fusion 360

- Fusion has python included
- Currently version 3.7.6
  - **NOT** 3.9.0

API - Good to Know

- Lengths are usually in cm
- Angles are usually in radians
- Use try-except blocks to see possible error messages
- Try to avoid modules which are note pure python
- Always pay attention on case sensitive problems
    - The API contains some in their documentation as well

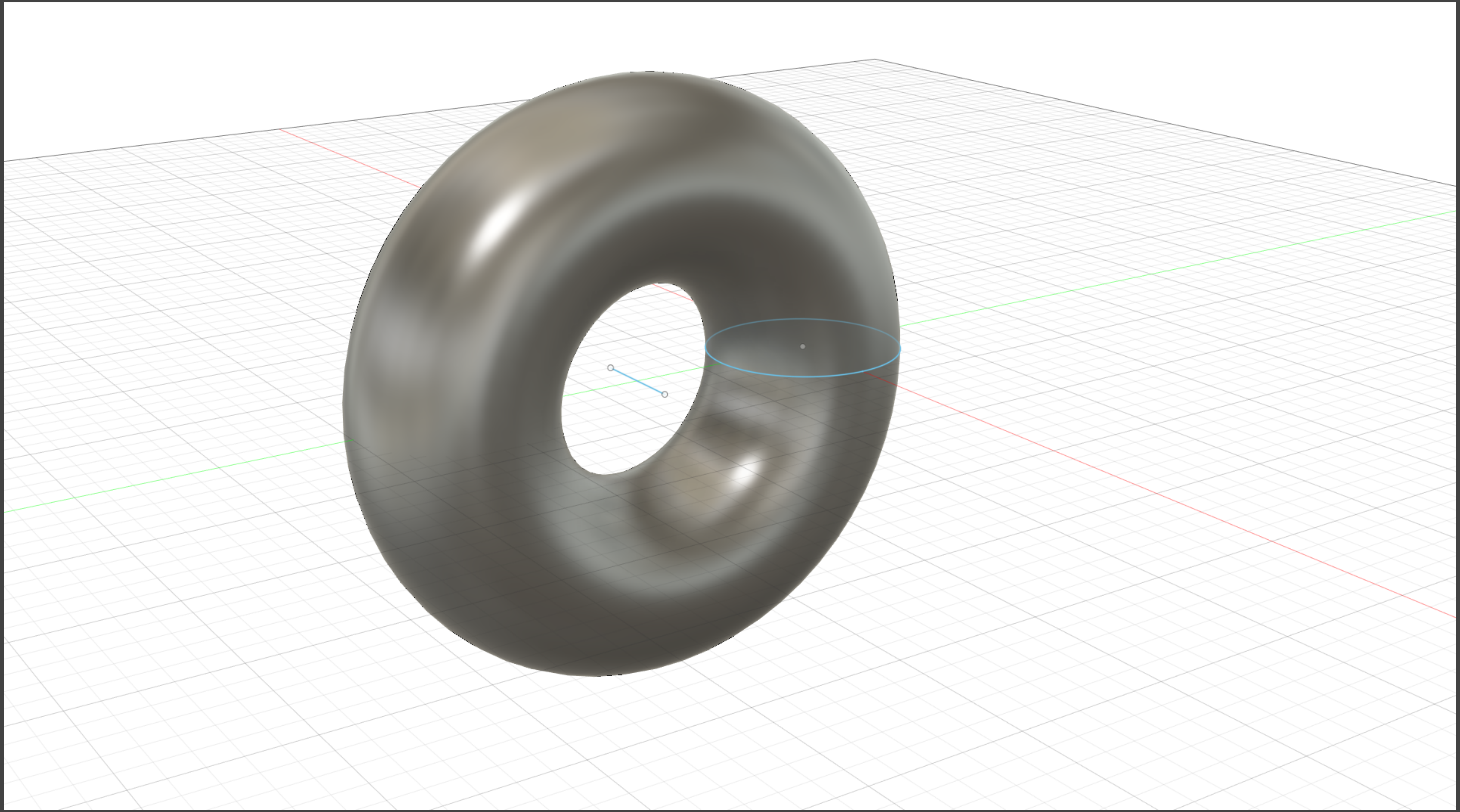
VS Code

- Used to code and debug
- Print commands are displayed here

IN-CLASS EXERCISE

Fusion 360 API User's Manual

In-Class Exercise



Exercise

Follow the Fusion 360 API User's Manual

Create this circle

ASSIGNMENT

Tasks for next week

Tasks for next week

- Modify the script to create a random number of circles
  - 5 - 10 circles
  - The circle for the first revolve has 1/n * 360 degrees
    - The second circle has 2/n * 360 degrees
    - The last circle is complete with 360 degrees
  - Optional:
    - Assign colours to the circles within your script
  - Hint:
    - Profiles inside a sketch may not be sorted by creation order

Tasks for next week