

Computational Aircraft Prototype Syntheses



Training Session 2

CAPS Geometry

ESP v1.18

Marshall Galbraith

galbramc@mit.edu

Massachusetts Institute of Technology

Bob Haines

haines@mit.edu

John F. Dannenhoffer, III

jfdannen@syr.edu

Syracuse University

- Python Basics
 - Strings, Functions, and Classes
- Loading and viewing geometry via pyCAPS
 - loadCAPS
- Accessing/modifying DESPMTR
 - set/getGeometryVal
 - saveGeometry
- Accessing SET and @values using OUTPMTR
 - getGeometryOutVal
- Directing bodies to AIMs
 - Attribute capsAIM
 - Attribute capsIntent
- Suggested Exercises

- String literals are defined with single or double quotes
- Alternating single/double quotes can be used to print quotes

```
print( "Hello World!" == 'Hello World!' ) #Prints: True
print( 'Hello World!' ) #Prints: "Hello World!"
print( "'Hello World!'" ) #Prints: 'Hello World!'
```

- String concatenated with plus operator

```
hello = "Hello"
world = 'World'
print( '"" + hello + ' ' + world + "!" + '""' ) #Prints: "Hello World!"
```

For more examples: www.w3schools.com/python/python_strings.asp

- Function defined with `def` keyword
- Indentation defines the body of the function
- Function arguments can have default values
- Function arguments can be specified as key value pairs

```
def my_function(thing1 = 1, thing2 = 2):
    print("thing1 =", thing1, "thing2 =", thing2)

my_function()                # Prints: thing1 = 1 thing2 = 2
my_function(2,3)            # Prints: thing1 = 2 thing2 = 3
my_function(thing1 = 4)     # Prints: thing1 = 4 thing2 = 2
my_function(thing2 = 6, thing1 = 5) # Prints: thing1 = 5 thing2 = 6

# Function with a return value
def my_sum(arg1, arg2):
    return arg1 + arg2

print( my_sum(1,2) ) # Prints: 3
print( my_sum("Hello ", "World!") ) # Prints: Hello World!
```

For more examples: www.w3schools.com/python/python_functions.asp

- Python is an object oriented language
 - Objects store data and methods to manipulate that data
- A **class** stores the blueprint to instantiate an object
- An object is instantiated by calling the class constructor
 - The optional `__init__` method instantiates an object

```
class MyClass:  
    def __init__(self, x = 5):  
        self.x = x  
    def printx(self):  
        print(self.x) # Print the value of x to the terminal
```

```
p1 = MyClass()  
p1.x = 10  
p1.printx() # Prints: 10
```

```
p2 = MyClass()  
p2.printx() # Prints: 5  
p1.printx() # Prints: 10
```

For more examples: www.w3schools.com/python/python_classes.asp

- Python Basics
 - Strings, Functions, and Classes
- Loading and viewing geometry via pyCAPS
 - loadCAPS
- Accessing/modifying DESPMTR
 - set/getGeometryVal
 - saveGeometry
- Accessing SET and @values using OUTPMTR
 - getGeometryOutVal
- Directing bodies to AIMs
 - Attribute capsAIM
 - Attribute capsIntent
- Suggested Exercises

session02/f118-A.csm

```
# F-118A Boxster
```

```
# wing design parameters
```

```
DESPMTR  wing:area      4240  # area
DESPMTR  wing:aspect    9.00   # aspect ratio
DESPMTR  wing:thick     0.10   # thickness ratio
DESPMTR  wing:xroot     54.0   # xloc at root LE
DESPMTR  wing:zroot     -5.0   # zloc at root LE
```

```
# horizontal tail design parameters
```

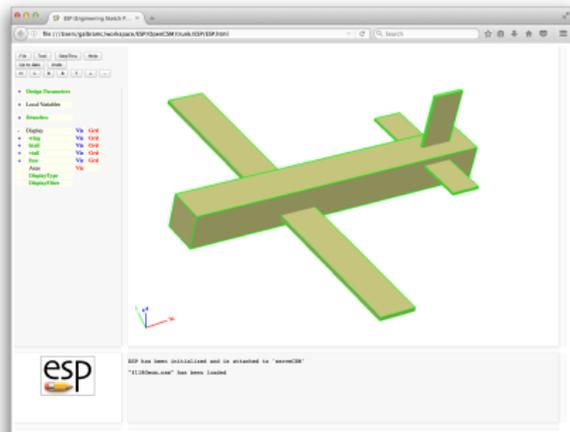
```
DESPMTR  htail:area     1210   # htail area
DESPMTR  htail:aspect   4.15   # htail aspect ratio
DESPMTR  htail:thick    0.08   # htail thickness
DESPMTR  htail:xroot    145    # xloc of root LE
DESPMTR  htail:zroot    5      # zloc of root LE
```

```
# vertical tail design parameters
```

```
DESPMTR  vtail:area     610    # vtail area
DESPMTR  vtail:aspect   1.80   # vtail aspect ratio
DESPMTR  vtail:thick    0.08   # vtail thickness
DESPMTR  vtail:xroot    150    # xloc of root LE
DESPMTR  vtail:zroot    9      # zloc of root LE
```

```
# fuselage design parameters
```

```
DESPMTR  fuse:length    180    # fuselage length
DESPMTR  fuse:width     20     # width of fuselage
DESPMTR  fuse:height    20     # height of mid fuselage
```



session02/f118-A.csm

```

#-----
# set available output parameters
OUTPMTR wing:wet
OUTPMTR wing:volume

OUTPMTR htail:wet
OUTPMTR htail:volume

OUTPMTR vtail:wet
OUTPMTR vtail:volume

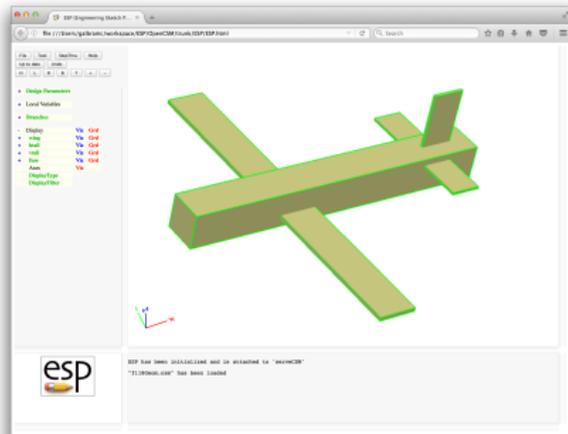
OUTPMTR fuse:wet
OUTPMTR fuse:volume

#=====
# Wing
SET      wing:span      sqrt(wing:aspect*wing:area)
SET      wing:chord     wing:area/wing:span

BOX      wing:xroot     -wing:span/2  wing:zroot  wing:chord  wing:span  wing:chord*wing:thick
SELECT  body
        ATTRIBUTE _name  $Wing

SET      wing:wet      @area
SET      wing:volume   @volume

```



session02/f118_1_Geom.py

```
#-----#  
  
# Allow print statement to be compatible between Python 2 and 3  
from __future__ import print_function  
  
# Import capsProblem class from pyCAPS  
from pyCAPS import capsProblem  
  
#-----#  
  
# Instantiate capsProblem object  
myProblem = capsProblem()
```

- capsProblem provides the context for a CAPS session
 - Multiple capsProblems may be instantiated, but cannot interact

Importing Modules: www.w3schools.com/python/python_modules.asp

- Geometry loaded with loadCAPS
- Returns class `pyCAPS.capsGeometry` instance
- Visualize with `capsViewer` using viewGeometry

session02/f118_1_Geom.py

```
# Instantiate capsProblem object
myProblem = capsProblem()

# Load geometry [.csm] file
# loadCAPS returns a class allowing interaction with bodies on the stack
# The geometry is not built with loadCAPS
filename = "f118-A.csm"
print ('\n==> Loading geometry from file "' + filename + '"...')
f118 = myProblem.loadCAPS(filename)

# The same geometry instance is available via myProblem.geometry
assert(f118 == myProblem.geometry)

# Build and view the geometry with the capsViewer
print ('\n==> Building and viewing geometry...')
f118.viewGeometry()

# Close CAPS (optional)
myProblem.closeCAPS()
```

- Python Basics
 - Strings, Functions, and Classes
- Loading and viewing geometry via pyCAPS
 - loadCAPS
- Accessing/modifying DESPMTR
 - set/getGeometryVal
 - saveGeometry
- Accessing SET and @values using OUTPMTR
 - getGeometryOutVal
- Directing bodies to AIMs
 - Attribute capsAIM
 - Attribute capsIntent
- Suggested Exercises

- DESPMTR are modified/accessed with set/getGeometryVal

session02/f118-A.csm

```
# fuselage design parameters
DESPMTR  fuse:length      180  # fuselage length
DESPMTR  fuse:width      20   # width of fuselage
DESPMTR  fuse:height     20   # height of mid fuselage
```

session02/f118_2_DESPMTR.py

```
# Load geometry [.csm] file
filename = "f118-A.csm"
print ('\n==> Loading geometry from file "' + filename + '"...')
f118 = myProblem.loadCAPS(filename)

# Set wide fuselage
f118.setGeometryVal("fuse:width", 60)
```

- DESPMTR are modified/accessed with set/getGeometryVal

session02/f118-A.csm

```
# horizontal tail design parameters
DESPMTR   htail:area      1210  # htail area
DESPMTR   htail:aspect   4.15  # htail aspect ratio
DESPMTR   htail:thick    0.08  # htail thickness
DESPMTR   htail:xroot    145   # xloc of root LE
DESPMTR   htail:zroot    5     # zloc of root LE
```

session02/f118_2_DESPMTR.py

```
# Double the htail:area
htail_area = f118.getGeometryVal("htail:area")
f118.setGeometryVal("htail:area", htail_area*2)

print ("--> old htail:area = ", htail_area)
print ("--> new htail:area = ", f118.getGeometryVal("htail:area"))

# Build and view the geometry with the capsViewer
print ('\n==> Bulding and viewing geometry...')
f118.viewGeometry()
```

- DESPMTR are modified/accessed with set/getGeometryVal

session02/f118_2_DESPMTR.py

```
# Build the Canard variant

# Reset the fuselage
f118.setGeometryVal("fuse:width", 20)

htail_area = f118.getGeometryVal("htail:area")
wing_area = f118.getGeometryVal("wing:area")

# Swap wing and htail area
f118.setGeometryVal("htail:area", wing_area)
f118.setGeometryVal("wing:area", htail_area/2)

# Rebuild and view geometry
print ('\n=> Bulding and viewing geometry...')
f118.viewGeometry()
```

- Modified geometry can be saved with saveGeometry
 - Available extensions: .egads .stp .step .igs .iges .brep

session02/f118_2_DESPMTR.py

```
# Build and view the geometry with the capsViewer
print ('\n==> Bulding and viewing geometry...')
f118.viewGeometry()
```

session02/f118_3_Save.py

```
# Build and save geometry
print ('\n==> Bulding and saving geometry...')
f118.saveGeometry("f118_3_Save_Wide.egads")
```

- View geometry with:
 - serveCSM f118_3_Save_Wide.egads
 - serveCSM f118_3_Save_Canard.egads

- Python Basics
 - Strings, Functions, and Classes
- Loading and viewing geometry via pyCAPS
 - loadCAPS
- Accessing/modifying DESPMTR
 - set/getGeometryVal
 - saveGeometry
- Accessing SET and @values using OUTPMTR
 - getGeometryOutVal
- Directing bodies to AIMs
 - Attribute capsAIM
 - Attribute capsIntent
- Suggested Exercises

- OUTPMTR values are accessed with `getGeometryOutVal`

session02/f118-A.csm

```
#-----  
# set available output parameters  
OUTPMTR wing:wet  
OUTPMTR wing:volume  
  
BOX wing:xroot -wing:span/2 wing:zroot wing:chord wing:span wing:chord*wing:thick  
SELECT body  
    ATTRIBUTE _name $Wing  
  
SET wing:wet @area  
SET wing:volume @volume
```

session02/f118_4_OUTPMTR.py

```
# Load geometry [.csm] file  
filename = "f118-A.csm"  
print ('\n==> Loading geometry from file "'+filename+'"...')  
f118 = myProblem.loadCAPS(filename)  
  
# Build and print all available output parameters  
print ("--> wing:wet      =", f118.getGeometryOutVal("wing:wet"  ) )  
print ("--> wing:volume   =", f118.getGeometryOutVal("wing:volume" ) )
```

- None returned for any OUTPMTR not SET

session02/f118-A.csm

```
OUTPMTR fuse:wet
OUTPMTR fuse:volume
```

```
BOX 0 -fuse:width/2 -fuse:height/2 fuse:length fuse:width fuse:height
SELECT body
  ATTRIBUTE _name $Fuselage
```

```
# fuse:wet and fuse:volume not set
```

session02/f118_4_OUTPMTR.py

```
# Accessing OUTPMTR that has not been set
print ("--> fuse:wet      =", f118.getGeometryOutVal("fuse:wet"   ) )
print ("--> fuse:volume   =", f118.getGeometryOutVal("fuse:volume" ) )
```

- Accessing non-OUTPMTR gives CAPS_NOTFOUND error

session02/f118-A.csm

```
#-----  
# set available output parameters  
OUTPMTR wing:wet  
OUTPMTR wing:volume  
  
OUTPMTR htail:wet  
OUTPMTR htail:volume  
  
OUTPMTR vtail:wet  
OUTPMTR vtail:volume  
  
OUTPMTR fuse:wet  
OUTPMTR fuse:volume  
  
#=====  
# Wing  
SET      wing:span      sqrt(wing:aspect*wing:area)  
SET      wing:chord     wing:area/wing:span
```

session02/f118_4_OUTPMTR.py

```
# Attempt to get a SET value not defined as OUTPMTR (causes and error)  
print ("--> wing:span      =", f118.getGeometryOutVal("wing:span"  ) )
```

- Python Basics
 - Strings, Functions, and Classes
- Loading and viewing geometry via pyCAPS
 - loadCAPS
- Accessing/modifying DESPMTR
 - set/getGeometryVal
 - saveGeometry
- Accessing SET and @values using OUTPMTR
 - getGeometryOutVal
- Directing bodies to AIMS
 - Attribute **capsAIM**
 - Attribute **capsIntent**
- Suggested Exercises

- capsAIM attribute
 - String semicolon separated AIM names
 - AIMs suitable to use the body
- capsIntent attribute
 - Optional string used to direct bodies to AIM instance
 - String semicolon separated names
 - Multiple bodies may have the same capsIntent

session02/f118-B.csm

```
# Htail
SET      htail:span      sqrt(htail:aspect*htail:area)
SET      htail:chord    htail:area/htail:span

BOX      htail:xroot    -htail:span/2  htail:zroot  htail:chord  htail:span  htail:chord*htail:thick
SELECT  body
ATTRIBUTE capsAIM      $masstranAIM;astrosAIM
ATTRIBUTE capsIntent   $htail;tail
ATTRIBUTE _name        $Htail
```

- Loading AIM with loadAIM
 - aim: The name of the AIM to load
 - analysisDir: Directory to write files if any (must be unique)
 - altName: Alternative name to track multiple instances of the same type of AIM
- Missing capsIntent loads all bodies with matching capsAIM

session02/f118_5_AIM.py

```
# capsAIM == $masstranAIM
masstranAll = myProblem.loadAIM(aim = "masstranAIM",
                                analysisDir="masstranALL", altName="All")

# The AIM instance is also available in the capsProblem.analysis dict
assert(masstranAll == myProblem.analysis["All"])

# Show the geometry used by the AIM
print("=> Geometry used by masstranAll instance with no capsIntent")
masstranAll.viewGeometry()
```

- Loading masstranAIM with bodies
capsAIM == \$masstranAIM and capsIntent == \$wing

session02/f118-B.csm

```
# Wing
SET      wing:span      sqrt(wing:aspect*wing:area)
SET      wing:chord     wing:area/wing:span

BOX      wing:xroot     -wing:span/2  wing:zroot   wing:chord  wing:span  wing:chord*wing:thick
SELECT  body
  ATTRIBUTE capsAIM      $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent   $wing
  ATTRIBUTE _name        $Wing
```

session02/f118_5_AIM.py

```
# capsAIM == $masstranAIM and capsIntent == $wing
myProblem.loadAIM(aim = "masstranAIM", capsIntent="wing",
                  analysisDir="masstranWing", altName="Wing")

# Show the geometry used by the AIM
print("=> Geometry used by Wing instance with capsIntent='wing'")
myProblem.analysis["Wing"].viewGeometry()
```

- Loading masstranAIM with bodies
capsAIM == \$masstranAIM and capsIntent == \$tail

session02/f118-B.csm

```
BOX htail:xroot -htail:span/2 htail:zroot htail:chord htail:span htail:chord*htail:thick
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $htail;tail
  ATTRIBUTE _name $Htail
```

```
BOX vtail:xroot 0 vtail:zroot vtail:chord vtail:chord*vtail:thick vtail:span
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $vtail;tail
  ATTRIBUTE _name $Vtail
```

session02/f118_5_AIM.py

```
# capsAIM == $masstranAIM and capsIntent == $tail
masstranTail = myProblem.loadAIM(aim = "masstranAIM", capsIntent="tail",
                                analysisDir="masstranTail", altName="Tail")
```

- Loading masstranAIM with bodies
capsAIM == \$masstranAIM and
(capsIntent == \$wing or capsIntent == \$fuse)

session02/f118-B.csm

```
BOX wing:xroot -wing:span/2 wing:zroot wing:chord wing:span wing:chord*wing:thick
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $wing
  ATTRIBUTE _name $Wing
```

```
BOX 0 -fuse:width/2 -fuse:height/2 fuse:length fuse:width fuse:height
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $fuse
  ATTRIBUTE _name $Fuselage
```

session02/f118_5_AIM.py

```
# capsAIM == $masstranAIM and (capsIntent == $wing or capsIntent == $fuse)
myProblem.loadAIM(aim = "masstranAIM", capsIntent=["wing","fuse"],
  analysisDir="masstranWingFuse", altName="WingFuse")
```

Fix f118-B.csm

- SET fuse:wet and fuse:volume in session02/f118-B.csm
- Add wing:span as OUTPMTR in session02/f118-B.csm
- Rerun session02/f118_4_OUTPMTR.py

Custom f118-A.csm

- Customize the f118-A.csm with setGeometryVal
 - Start from a copy of session02/f118_2_DESPMTR.py

Custom masstran analysis

- Load wing, htail and fuselage into a masstranAIM
 - Start from a copy of session02/f118_5_AIM.py
- Create your own (optionally share it galbramc@mit.edu)