Last class

In [7]:

# This is a function consisting of:
#   1. a header (def ...): "def" is a keyword
#   2. a body (print ...): the body is indented using <tab>

def say_introduction():
    print("My name is Inigo Montoya.")

def threaten_vengeance():
    print("You killed my father.")
    print("Prepare to die.")

print("Hello.")

Hello.
Last class

```python
# We can use functions that someone else wrote
from simplefunctions import print_date_and_time  # make a function available to you

print_date_and_time()  # call the function

2019-06-08 10:06:47.104904
```
# draw a simple house

```python
def draw_house(x, y):
    # drawing code here
```

```
import drawSvg as draw

x = -100  # shift house's horizontal position by this amount

d = draw.Drawing(200, 200, origin='center')  # create canvas

d.append(d.set(stroke='black', fill='gray').rect(100, 100, 100, 100))  # body of house

d.append(d.set(stroke='black', fill='green').polyline([(0, 0), (100, 0), (50, 50), (0, 50)]))  # roof

d.append(d.set(stroke='black', fill='red').rect(40, 20, 50, 100))  # door

d.append(d.set(stroke='black', fill='white').rect(15, -30, 15, 15))  # left window

d.append(d.set(stroke='black', fill='white').rect(70, -30, 15, 15))  # right window

d.set(stroke='black', fill='black').rect(0, 0, 100, 100)  # background
```

```
Out[56]:
```
print (style v. substance)

print( "threaten vengeance" )
print ( "threaten vengeance" )
print ("threaten vengeance")
print("threaten vengeance")
# draw five rectangles of the same size 50 x 50

```python
import drawSvg as draw

d = draw.Drawing(200, 200, origin='center')

d.append(draw.Rectangle(0,0,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(-20,35,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(5,45,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(-50,50,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(-100,10,50,50, fill='red', stroke='black'))

d
```
# draw five rectangles of the same size 50 x 50

```python
import drawSvg as draw

d = draw.Drawing(200, 200, origin='center')

d.append(draw.Rectangle(0,0,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(-20,35,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(5,45,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(-50,50,50,50, fill='red', stroke='black'))
d.append(draw.Rectangle(-100,10,50,50, fill='red', stroke='black'))

d
```

Variables
# draw five rectangles of the same size 50 x 50
import drawSvg as draw

d = draw.Drawing(200, 200, origin='center')
r = 50  # assignment, not “equals”

d.append(draw.Rectangle(0, 0, r, r, fill='red', stroke='black'))
d.append(draw.Rectangle(-20, 35, r, r, fill='red', stroke='black'))
d.append(draw.Rectangle(5, 45, r, r, fill='red', stroke='black'))
d.append(draw.Rectangle(-50, 50, r, r, fill='red', stroke='black'))
d.append(draw.Rectangle(-100, 10, r, r, fill='red', stroke='black'))

d
# draw five rectangles of the same size 50 x 50
import drawSvg as draw

d = draw.Drawing(200, 200, origin='center')
r = 25 # assignment, not “equals”

d.append(draw.Rectangle(0,0,r,r, fill='red', stroke='black'))
d.append(draw.Rectangle(-20,35,r,r, fill='red', stroke='black'))
d.append(draw.Rectangle(5,45,r,r, fill='red', stroke='black'))
d.append(draw.Rectangle(-50,50,r,r, fill='red', stroke='black'))
d.append(draw.Rectangle(-100,10,r,r, fill='red', stroke='black'))

d
Variables: int

meaning_of_life = 42
Variables: int

meaning_of_life = 42

print( meaning_of_life )
Variables: int

meaning_of_life = 42

print( meaning_of_life )

output: 42
Variables: int

```
meaning_of_life = 42
```

computer memory
Variable Names

meaning_of_life
Variable Names

meaning_of_life

meaningOfLife
Variable Names

meaning_of_life

meaningOfLife

whatever
Variable Names

meaning_of_life ≠ Meaning_of_life

42_is_the_meaning_of_life

meaning of life = 42
Variables: int, long int

int : -2,147,483,648 ... 2,147,483,647

int : $-2^{31}$ ... $2^{31}-1$

long int : anything larger than an int
Variables: floating-point

\[ a = 6 \]

\[ a = 6.02 \]

\[ a = 6.02e23 \quad \# \quad 6.02 \times 10^{23} \]
Expressions and Operators

print( 18 + 24 )
Expressions and Operators

print( 18 + 24 )

output: 42
Expressions and Operators

\[
\text{print}(18 + 24)
\]

expression
Expressions and Operators

```python
print( 18 + 24 )
```

operator
Expressions and Operators

print( 18 + 24 )

operands
Expressions and Operators

addition +

subtraction -

multiplication *

division /

modulus (mod) %  

9 % 4 -> 1
Expressions and Operators

print( 24 + 3 * 6 ) 42

print( (24 + 3) * 6 ) 162

print( 100 / 5 * 2 ) 40.0

print( 100 / (5 * 2) ) 10.0
Typed Operators

print( (3 * 10) / 4 )  7.5
Typed Operators

print( (3 * 10) / 4 )

print( 30.0 / 4.0 )

7.5
# ----- DRILL -----
# Write a function `draw_bullseye` consisting of
# an inner red circle superimposed atop
# a green circle superimposed atop
# an outer blue circle
# (you should therefore see a red circle, green ring and blue ring)
# define a single variable `s` that scales the size of the bullseye
# (i.e. the radius of each circle)
# ----- DRILL -----
# Write a function `draw_bullseye` consisting of
# an inner red circle superimposed atop
# a green circle superimposed atop
# an outer blue circle (you should therefore see a red circle, green ring and blue ring)
# define a single variable `s` that scales the size of the bullseye

```python
import drawSvg as draw

d = draw.Drawing(200, 200, origin='center')

s = 0.5  # scale bullseye by this amount

d.append(draw.Circle(0, 0, s*100, fill='blue'))  # outer part
d.append(draw.Circle(0, 0, s*75, fill='green'))  # central part
d.append(draw.Circle(0, 0, s*50, fill='red'))   # inner part

d
```
Assignment Operator

\[ x = 5 \]
Assignment Operator

\[ x = 5 \]

\[ y = 6 \]
Assignment Operator

\[ x = 5 \]
\[ y = 6 \]
\[ x = 4 \]
Assignment Operator

```python
x = 5
y = 6
x = 4
z = x + 1
print(z)
```
Assignment Operator

x = 5
y = 6
x = 4
z = x + 1

print(z)  # Output: 5
Assignment Operator

```python
x = 5
y = 6
x = 4
z = x + 1
print(z)  # 5
x = 5
print(z)
```
Assignment Operator

x = 5
y = 6
x = 4
z = x + 1

print( z ) 5

x = 5
print( z ) 5
Assignment Operator

x = 5
y = 6
x = 4
x = x + 1

print(x)
Assignment Operator

x = 5
y = 6
x = 4
x = x + 1
print(x)  5
Variables: string

last_letter = "z"

print( last_letter )

output: z
Variables: string

print( "hello" )

hello = 5

print( hello )
Variables: string

print( "4 + 7" )

print( 4 + 7 )
print( 4 + 7 )

output: 11

print( "hello " + "my name" )

output: hello my name
Variables: convert between types

```python
print( float(4) )            # 4.0
print( int(3.14) )           # 3
print( str(4) + str(2) )     # 42
print( int("4") + int("2") ) # 6
```
Variables: boolean

x = True
print(x)
output: True

y = False
print(y)
output: False
Summary

• Variables
  • store information in computer memory
  • int, long int, float, string, booleans, functions
  • naming

• Expressions and Operators
  • arithmetic
  • just like functions
  • assignment