SD18057
Programming Basics for AEC
Luke Perkins
Stewart – Raleigh NC

Learning Objectives
- Fundamental Programming Concepts
- Languages
- Basics of Software Development
- Programming & AutoCAD

Description
Are you an architect, engineer, drafter, technician, Revit software pro, or AutoCAD Civil 3D software guru? Yes! Do you also have a degree in computer science, or extensive experience as a software developer? No! Great, this class is for you. There's a trend in design software, a by-product of the Internet of Things: designers increasingly asked to understand code, and some even needing to write it themselves. But, how do you do that? You start with the fundamentals of programming—don't worry about which language to start in. Worry about learning the general concepts of programming. Worry about learning how to find the information you need. Learn how to teach yourself coding basics. This class will present the basics of any programming language, how to think programmatically, and the basics of developing software. We'll even introduce some tools for development beyond the code itself. You learn the meaning of terms like: loops, strings, variable, globals, API, and extents testing. This session features AutoCAD and AutoCAD Civil 3D. AIA Approved

Your AU Expert
Luke Perkins is an Engineering Analyst for Stewart in Raleigh, NC. After graduating North Carolina State University in 2014 he has spent the last two and a half years as a civil engineering designer in the land development industry. During his time at Stewart, he has focused on helping the Stewart Land Planning and Design group fully utilize Civil 3D, Infraworks, the integration of civil design and BIM, and enhanced stormwater design. Luke has a passion for technology and has been a part of distinguished design teams on notable projects in North Carolina that are changing the face of Raleigh in the midst of surging growth.
Fundamental Programming Concepts

Functions
Without focusing on a specific programming language and rather what is the ultimate reason for creating a function, we start to understand the importance of combining programming and the AEC. We are terribly behind other industries when it comes to implementing tools to replace repetition. In short, a function is a commonly used routine. Programmatic thinking begins when you start identifying processes that are repetitive and can be replaced by functions. A metaphorical example would be making a cocktail. While there are different ingredients that could be used to make a cocktail the process is the same each time.

Variables are an important component of functions. Local variables are variables contained to the function. Either input parameters of the function, or variables that have been defined within the function that can only be used by the function. Global variables can be used and manipulated outside the function. In our example above, the ingredients could represent an input parameter, or local variables of a function.

Example: Programming a simple function

Define steps to a simple process

""" Functions in Vegas - Fixing up some Cocktails """

    print('Filling shaker with ice')
    print('Pouring in the alcohol')
    print('Adding in the mixer')
    print('Shaking it up')
    print('Pour into tumbler and')
    whiskeySour = 'The perfect drink'

Functionalize the process

""" Functions in Vegas - Whiskey Straight Please """

    def fix_oldFashioned():
        print('Filling shaker with ice')
        print('Pouring in the alcohol')
        print('Adding in the mixer')
        print('Shaking it up')
        print('Pour into tumbler')
        oldFashioned = 'An awesome drink'
        return oldFashioned

Functionalize similar functions
""" Functions in Vegas - Whiskey Straight Please """

def fix_drink():
    print('Filling shaker with ice')
    print('Pouring in the alcohol')
    print('Adding in the mixer')
    print('Shaking it up')
    print('Pour into tumbler')

def fix_oldFashioned():
    fix_drink()
    oldFashioned = 'The Perfect Old Fashioned'
    return oldFashioned

def fix_whiskeySour():
    fix_drink()
    whiskeySour = 'The Perfect Whiskey Sour'
    return whiskeySour

Objects
Object oriented programming (OOP) gives us a way to organize components of code by relating virtual pieces of information to something tangible. Objects in our program are the nouns. When asking yourself if something is an object, try answering these questions: Is it carrying out a specific task or is a task being carried out on it? Does it have attributes? In our previous example, building a cocktail was a function; the cocktail itself is an object. Objects that are essential to our program might be constructed as a class; classes are objects with relevant qualities and methods.

Methods are a vital piece of Objects in OOP. In their simplest form, methods are pieces of code called by name associated with objects. They are “functions” able to operate on data contained within the class (object).

Example: Simple Objects and Methods

""" Casino Objects """

class Casino:
    def __init__(self, name, allowance):
        self.name = name
        self.allowance = allowance

    def gamble(self):
        if self.allowance > 0:
            self.allowance -= 100
            print('rolling the dice at {}'.format(self.name))
        else:
            print('{} took all my money :('.format(self.name))
Inheritance

Classes that are related could contain the same method. Inheritance is the ability to structure classes so that methods can be passed down to multiple associated classes. A more AutoCAD example of this could be a Class called “Areas” that contained methods for creating the internal pattern and coloring of an area. Classes such as “Hatch” or “Polygon” could inherit these methods, because they both require similar methods. In the example below casino objects have been separated into casinos located along The Strip or Fremont. While these objects have unique methods they inherit the initialization method as well as the gamble method. Using Inheritance is a powerful tool that can decrease the amount of coding require to create a program, and define meaningful relationships between various components of code.

Example: Inheritance

```
""" Casino Objects """

class Casino:

    def __init__(self, name, allowance):
        self.name = name
        self.allowance = allowance

    def gamble(self):
        if self.allowance > 0:
            self.allowance -= 100
            print('rolling the dice at {}. Winner winner chicken dinner!'.format(self.name))
        else:
            print('{} took all my money :('.format(self.name))

class TheStrip(Casino):
    
    def show(self):
        print('You just saw an awesome show on the strip!')

class Fremont(Casino):

    def zipline(self):
        print('Zooming down the zip line on Fremont!')
```

Loops and Conditions

Loops and conditional executions are the tools we have at our disposal to handle the logic we process in our program. Loops are pieces of code that perform an iterative operation until certain conditions are being met.

Example: For Loop
""" Casino Objects """

casino = ['The Venetian','The Mirage', 'MGM Grand', 'Treasure Island']

for activity in casino:
    print('I saw an awesome show at {}'.format(activity))

Example: While Loop

""" While Loop - 2016 election"""

Trump = True
Year=2016

while Trump:
    print('Its {} and my girlfriend is unhappy'.format(Year))
    Year = Year+1
    if Year == 2020:
        print('My girlfriend is happy! Its 2020 and Trump is up for re-election!')
        Trump = False

Modules

Packages and modules are sets of tools organized by their functionality that can be imported for use. This means our program doesn't have to replicate any available code. This becomes a huge benefit when programming tools for AutoCAD, because there are many Visual Basic and C# modules that can be imported for use within the Visual Studio programming environment

Programming Languages

Popular Languages in Web Development
HTML – The standard markup language for creating web pages and web applications
JavaScript – Interpreted programming language for web content production
Python – Popular, high level, general purpose, object oriented programming language
PHP – Server-side scripting language used mostly in web development
Ruby – Popular, high level, general purpose, object oriented programming language

Languages in Programming for AutoCAD
Auto LISP – Macro programming language for limited and specific tasks within AutoCAD
Visual Basic .NET – .NET object oriented language designed for rapid application development and shallow learning curve
C# - .NET object oriented language designed to be simple, powerful, and versatile.

Basics of Software Development

There are many thought diagrams and processes for summarizing the progression of software development, but generally there is a communication and planning phase, a design phase, a construction phase, and the deployment and maintenance phase.
An important creation of this process is the minimum viable product (MVP). Startups use this terminology because building something fast in the world of startups is life or death. Something that the rest of the programming community has learned from the startup MVP is the importance of validation. It is important to get to a point where a piece of software can be validated by the user as being worth of further pursuit or if adjustments to the plan and execution need to be made.

Programming & AutoCAD

Macros
Thinking back to functions and thinking programmatically, macros are useful strings of commands and options within commands combined to create a new process. This eliminates repetition and increases efficiency without having to create ‘custom’ code. If macros seem like a good option to enhance your day to day work, take them a step further:
http://au.autodesk.com/au-online/classes-on-demand/class-catalog/2014/autocad/ac4873#chapter=0

API
API is the acronym for Application Programming Interface; this interface is the way a software programmer can communicate with a software product. After software is developed, for additional customization, developers also lay the groundwork for how users can interact with the software by way of the software’s API. It’s essentially the infrastructure between the software and a programmer outside of the development team.
Plug-in
A plugin is a type of program module that adds functionality to a piece of software. It’s essentially a programmer utilizing the software’s API to create additional features for specific uses. Creating your first plug-in is a process that is well documented by Autodesk and it is worth walking through the tutorial to understand the desired method of using the .NET API. Visit the Online Developer Guide to challenge yourself to become a small time developer!

AutoCAD
AutoCAD® software provides a flexible development platform for specialized design and drafting applications. Its open architecture enables you to customize AutoCAD to suit your unique purposes. Examples of customizations include Autodesk’s industry-specific design software and thousands of add-on applications from members of the Autodesk® Developer Network.

AutoCAD provides the ObjectARX® programming environment with the object-oriented C++ programming interfaces, the managed AutoCAD .NET API, the complete Visual LISP® development environment and the ActiveX® interface.

My First AutoCAD Plug-in:

http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=1911627

Additional Resources

https://www.lynda.com/Python-tutorials/Welcome/418249/459105-4.html

http://au.autodesk.com/au-online/classes-on-demand/class-catalog/2014/autocad/ac4873#chapter=0

http://au.autodesk.com/au-online/classes-on-demand/class-catalog/2012/autocad-civil-3d/getting-started-with-autocad-civil-3d-net-programming#chapter=0