Procedural (C) vs OOP (C++/Java/Python)

- Large software can be built in C. e.g: Linux Kernel
- OOP: Easier to design, reuse components and build large software
- SDE: should know OOP as they will be working in teams building large software
- ML & DS roles: Should know the basics to use various modules and extend them when needed.
- The true power of OOP and OOD can be undestood when deisgning a large piece of software like a library for plotting

Pre-requisites: previous live-sessions in this series.

In [139]:

```
import sys
print (sys.version)
```

```
3.7.3 (default, Mar 27 2019, 09:23:15)
[Clang 10.0.1 (clang-1001.0.46.3)]
```

In [150]:

```
# NON-OOP (a.k.a Procedural) based implementation of a ShortURL System.

import random
import string

d = dict();

# given a long URL, get a short URL
def getShortURL(longURL):
    # length = random value in 6-10
    l = random.randint(6,10);


    # generate random characters into a string of length l
    chars = string.ascii_lowercase
    shortURL = ''.join(random.choice(chars) for i in range(l))


    # check if this string is already present in dict d
    if shortURL in d:
        return getShortURL(longURL);
    else:
        d[shortURL] = longURL;


    r = "https://www.shortURL.com/"+shortURL
    return r;

def getLongURL(shortURL):

    # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
    k = shortURL[25:];


    if k in d:
        return d[k];
    else:
        return None;
```

Class: datatype

Object: variable

In [50]:

```
# Class: group all variables/attributes and functions/methods into a single logi
cal unit

class ShortURL:

    def __init__(self): # constructor; not must, but, good to have;  initialize
 all attributes here
        self.d=dict();

    # given a long URL, get a short URL
    def getShortURL(self, longURL): # first argument to all methods is "self" =>
this object
        # length = random value in 6-10
        l = random.randint(6,10);


        # generate random characters into a string of length l
        chars = string.ascii_lowercase
        shortURL = ''.join(random.choice(chars) for i in range(l))


        # check if this string is already present in dict d
        if shortURL in self.d:
            return getShortURL(longURL);
        else:
            self.d[shortURL] = longURL;


        r = "https://www.shortURL.com/"+shortURL
        return r;

    def getLongURL(self, shortURL):

       # print(self.d); # print statemnt for debugging

        # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
        k = shortURL[25:];


        if k in self.d:
            return self.d[k];
        else:
            return None;
```

In [51]:

```
# Class: datatype/DS & Object: variable of a class
s = ShortURL() # constructor being called; memory allocated.

print(type(s))
```

```
<class '__main__.ShortURL'>
```

What does this "__main__" mean?

The script invoked directly is considered to be in the **main** module. It can be imported and accessed the same way as any other module.

Modules:

[https://www.w3schools.com/python/python_modules.asp (https://www.w3schools.com/python/python_modules.asp)]

Consider a module to be the same as a code library. A file containing a set of functions you want to include in your application.

In [52]:

```
print(s.shortURL("appliedaicourse"))
print(s.shortURL("gate.appliedcourse.com"))
```

```
---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-52-84023b7d1bbb> in <module>
----> 1 print(s.shortURL("appliedaicourse"))
      2 print(s.shortURL("gate.appliedcourse.com"))

AttributeError: 'ShortURL' object has no attribute 'shortURL'
```

In [53]:

```
print(s.getShortURL("appliedaicourse.com"))
print(s.getShortURL("gate.appliedcourse.com"))
```

```
https://www.shortURL.com/alrpoy
https://www.shortURL.com/dlfjxuxhff
```

In [54]:

```
print(s.getLongURL("https://www.shortURL.com/alrpoy"))
```

```
appliedaicourse.com
```

In [55]:

```
print(s.d)
```

```
{'alrpoy': 'appliedaicourse.com', 'dlfjxuxhff': 'gate.appliedaicours
e.com'}
```

In [59]:

```
s.d["interviewprep.appliedcourse.com"] = "abcdefgh";

print(s.d)
```

```
{'alrpoy': 'appliedaicourse.com', 'dlfjxuxhff': 'gate.appliedaicours
e.com', 'interviewprep.appliedcourse.com': 'abcdefgh'}
```

In [151]:

```
# No need to have any atributes or methods
class EmptyClass:
    pass

e = EmptyClass();
print(type(e))
```

```
<class '__main__.EmptyClass'>
```

There are lots of internals and boundary cases for which reading documentation is a good idea.

- https://docs.python.org/3/tutorial/classes.html (https://docs.python.org/3/tutorial/classes.html)

We will focus more on applied aspects in the context of ML/AI.

In [77]:

```
# Class variables shared by all objects

# Class: group all variables/attributes and functions/methods into a soingle log
ical unit

class ShortURL1:

    URLPrefix = "https://www.shortURL.com/"; # class variable shared by all obje
cts

    def __init__(self): # constructor; not must, but, good to have;  initialize
 all attributes here
        self.d=dict();

    # given a long URL, get a short URL
    def getShortURL(self, longURL): # first argument to all methods is "self" =>
this object
        # length = random value in 6-10
        l = random.randint(6,10);


        # generate random characters into a string of length l
        chars = string.ascii_lowercase
        shortURL = ''.join(random.choice(chars) for i in range(l))


        # check if this string is already present in dict d
        if shortURL in self.d:
            return getShortURL(longURL);
        else:
            self.d[shortURL] = longURL;


        r = self.URLPrefix + shortURL
        return r;

    def getLongURL(self, shortURL):

       # print(self.d); # print statemnt for debugging

        # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
        k = shortURL[25:];


        if k in self.d:
            return self.d[k];
        else:
            return None;
```

In [78]:

```
s1 = ShortURL1();

print(s1.getShortURL("appliedaicourse.com"))
print(s1.getShortURL("gate.appliedaicourse.com"))
```

```
https://www.shortURL.com/bqrkdpum
https://www.shortURL.com/jaxgluaeoh
```

In [79]:

```
print(s1.d)
print(s1.URLPrefix)
```

```
{'bqrkdpum': 'appliedaicourse.com', 'jaxgluaeoh': 'gate.appliedaicou
rse.com'}
https://www.shortURL.com/
```

In [81]:

```
s1a = ShortURL1();
print(s1a.URLPrefix);
print(s1a.d)
```

```
https://www.shortURL.com/
{}
```

If you have learnt OOP in C++/Java, it is very easy to get confused with syntax and internals. Please beware

In [88]:

```
# "Private Members"

# Class: group all variables/attributes and functions/methods into a soingle log
ical unit

class ShortURL2:

    classVar = "test"; # Public => accesible directly from outside the class

    __URLPrefix = "https://www.shortURL.com/"; # "Private" memebers. Member => a
ttribute or method

    def __init__(self): # constructor; not must, but, good to have;  initialize
 all attributes here
        self.d=dict();

    # given a long URL, get a short URL
    def getShortURL(self, longURL): # first argument to all methods is "self" =>
this object
        # length = random value in 6-10
        l = random.randint(6,10);


        # generate random characters into a string of length l
        chars = string.ascii_lowercase
        shortURL = ''.join(random.choice(chars) for i in range(l))


        # check if this string is already present in dict d
        if shortURL in self.d:
            return getShortURL(longURL);
        else:
            self.d[shortURL] = longURL;


        r = self.__URLPrefix + shortURL
        return r;

    def getLongURL(self, shortURL):

       # print(self.d); # print statemnt for debugging

        # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
        k = shortURL[25:];


        if k in self.d:
            return self.d[k];
        else:
            return None;
```

In [90]:

```
s2 = ShortURL2();
print(s2.classVar)
print(s2.__URLPrefix)
```

```
test

---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-90-4fd735b2fbbe> in <module>
      1 s2 = ShortURL2();
      2 print(s2.classVar)
----> 3 print(s2.__URLPrefix)

AttributeError: 'ShortURL2' object has no attribute '__URLPrefix'
```

Abstraction: [https://en.wikipedia.org/wiki/Abstraction_principle_(computer_programming (https://en.wikipedia.org/wiki/Abstraction_principle_(computer_programming))]

- As a recommendation to the programmer, in its formulation by Benjamin C. Pierce in Types and Programming Languages (2002), the abstraction principle reads: “ Each significant piece of functionality in a program should be implemented in just one place in the source code. Where similar functions are carried out by distinct pieces of code, it is generally beneficial to combine them into one by abstracting out the varying parts. ”
- General concept in porgramming: libraries, classes
- "Its main goal is to handle complexity by hiding unnecessary details from the user. That enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity." [Source: https://stackify.com/oop-concept-abstraction/ (https://stackify.com/oop-concept-abstraction/)]

Encapsulation: [https://en.wikipedia.org/wiki/Encapsulation_(computer_programming (https://en.wikipedia.org/wiki/Encapsulation_(computer_programming))]

In object-oriented programming languages, and other related fields, encapsulation refers to one of two related but distinct notions, and sometimes to the combination thereof:

1. A language mechanism for restricting direct access to some of the object's components.
2. A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.

Design a MyURLShortner built on top of ShortURLFinal (given in a module) with some changes

- Change getShortURL to have both alphabets and numbers
- Change the URLPrefix to my website (myurlshortner.com)

In [105]:

```
# Following is the final implementation of ShortURL in the library

class ShortURLFinal:

    URLPrefix = "https://www.shortURL.com/"; # class-variable

    def __init__(self): # constructor; not must, but, good to have;  initialize
 all attributes here
        self.d=dict();

    # given a long URL, get a short URL
    def getShortURL(self, longURL): # first argument to all methods is "self" =>
this object
        # length = random value in 6-10
        l = random.randint(6,10);


        # generate random characters into a string of length l
        chars = string.ascii_lowercase
        shortURL = ''.join(random.choice(chars) for i in range(l))


        # check if this string is already present in dict d
        if shortURL in self.d:
            return getShortURL(longURL);
        else:
            self.d[shortURL] = longURL;


        r = self.URLPrefix + shortURL
        return r;

    def getLongURL(self, shortURL):

       # print(self.d); # print statemnt for debugging

        # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
        k = shortURL[25:];


        if k in self.d:
            return self.d[k];
        else:
            return None;
```

In [109]:

```
# Lets define MyURLShortner based on ShortURLFinal

# Inheritence: baseclass, derived class [https://docs.python.org/3/tutorial/classes.html#inheritance]
class MyURLShortner(ShortURLFinal):
    URLPrefix = "www.myurlshortner.com/" # overriding as same name as base-class


    # given a long URL, get a short URL
    def getShortURL(self, longURL): # overriding as same name as base-class, use both digits and lowercase
        # length = random value in 6-10
        l = random.randint(6,10);


        # generate random characters into a string of length l
        chars = string.ascii_lowercase + string.digits # both digits and lowercase
        shortURL = ''.join(random.choice(chars) for i in range(l))


        # check if this string is already present in dict d
        if shortURL in self.d:
            return getShortURL(longURL);
        else:
            self.d[shortURL] = longURL;


        r = self.URLPrefix + shortURL
        return r;

    # getLongURL and dict "d" not changed
```

In [110]:

```
m1 = MyURLShortner(); # base-class constructor is executed

print(m1.d)
```

```
{}
```

In [111]:

```
print(m1.getShortURL("amazon.com"))
print(m1.getShortURL("google.com"))
```

```
www.myurlshortner.com/mt2e9h
www.myurlshortner.com/ymnsyufq3
```

In [112]:

```
print(m1.d)
```

```
{'mt2e9h': 'amazon.com', 'ymnsyufq3': 'google.com'}
```

In [157]:

```
class A:
    def __init__(self, i):
        self.var = i;

    def printVar(self):
        print(self.var)


a = A(10);
a.printVar()

a = A(20);
a.printVar()
```

```
10
20
```

In [158]:

```
a =10;

print(type(a))
```

```
<class 'int'>
```

In [159]:

```
a =20;
```

In [115]:

```
# example from geometry: [Source: https://overiq.com/python-101/inheritance-and-polymorphism-in-python/]
import math

class Shape:

    def __init__(self, color='black', filled=False):
        self.__color = color
        self.__filled = filled

    def get_color(self):
        return self.__color

    def set_color(self, color):
        self.__color = color

    def get_filled(self):
        return self.__filled

    def set_filled(self, filled):
        self.__filled = filled


class Rectangle(Shape):

    def __init__(self, length, breadth):
        super().__init__()
        self.__length = length
        self.__breadth = breadth

    def get_length(self):
        return self.__length

    def set_length(self, length):
        self.__length = length

    def get_breadth(self):
        return self.__breadth

    def set_breadth(self, breadth):
        self.__breadth = breadth

    def get_area(self):
        return self.__length * self.__breadth

    def get_perimeter(self):
        return 2 * (self.__length + self.__breadth)


class Circle(Shape):
    def __init__(self, radius):
        super().__init__()
        self.__radius = radius

    def get_radius(self):
        return self.__radius

    def set_radius(self, radius):
        self.__radius = radius
```

```
    def get_area(self):
        return math.pi * self.__radius ** 2

    def get_perimeter(self):
        return 2 * math.pi * self.__radius


r1 = Rectangle(10.5, 2.5)

print("Area of rectangle r1:", r1.get_area())
print("Perimeter of rectangle r1:", r1.get_perimeter())
print("Color of rectangle r1:", r1.get_color())
print("Is rectangle r1 filled ? ", r1.get_filled())
r1.set_filled(True)
print("Is rectangle r1 filled ? ", r1.get_filled())
r1.set_color("orange")
print("Color of rectangle r1:", r1.get_color())

c1 = Circle(12)

print("\nArea of circle c1:", format(c1.get_area(), "0.2f"))
print("Perimeter of circle c1:", format(c1.get_perimeter(), "0.2f"))
print("Color of circle c1:", c1.get_color())
print("Is circle c1 filled ? ", c1.get_filled())
c1.set_filled(True)
print("Is circle c1 filled ? ", c1.get_filled())
c1.set_color("blue")
print("Color of circle c1:", c1.get_color())
```

```
Area of rectangle r1: 26.25
Perimeter of rectangle r1: 26.0
Color of rectangle r1: black
Is rectangle r1 filled ?  False
Is rectangle r1 filled ?  True
Color of rectangle r1: orange

Area of circle c1: 452.39
Perimeter of circle c1: 75.40
Color of circle c1: black
Is circle c1 filled ?  False
Is circle c1 filled ?  True
Color of circle c1: blue
```

In [ ]:

```

```

In [ ]:

```
### object is the base class for all classes in Python

class MyClass(object): # object is the base-class by default and implicitly.
    pass


# __new__() : creates a new object and calls the __init__()
# __init__() : default constructor
# __str__() : write code to convert object into string for printing.
```

In [161]:

```
a = [1,2,3,4];
print(type(a))

print(a)
```

```
<class 'list'>
[1, 2, 3, 4]
```

In [116]:

```
class ClassWithStr(): #default object is the base class
    def __str__(self):
        return "any string representation of the object of this class that we wa
nt"

c1 = ClassWithStr();
print(c1)
```

```
any string representation of the object of this class that we want
```

Example: https://matplotlib.org/3.1.1/api/axis_api.html (https://matplotlib.org/3.1.1/api/axis_api.html)

In [ ]:

```

```