Session 2 - Object Oriented Programming Fundamental

September 3, 2018

1 Python Training: Day 1 Session 2

References (more or less):

- https://docs.python.org/3/tutorial/
- https://realpython.com/
- https://www.learnpython.org/en/
- https://www.tutorialspoint.com/
- https://stackoverflow.com/questions/43211296/pgadmin4-postgresql-application-server-could-not-be-contacted
- https://stackoverflow.com/questions/12562928/psql-exe-password-authentication-failed-in-windows
- https://en.wikipedia.org/wiki/Bulls_and_Cows
- https://en.wikipedia.org/wiki/Fibonacci_number

1.1 Fundamentals of function

A function: "a named section of a program that performs a specific task." (https://www.webopedia.com/TERM/F/function.html)

A function might have input parameter and might return a value.

1.1.1 Function declaration

```
In [ ]: def hi(name):
            print('hi, my name is', name)

In [ ]: hi('budi')
        hi('ani')

In [ ]: def square(x):
            return x * x

In [ ]: n = square(2)
        print(n)
```

1.1.2 Lambda expression

Sometimes a short function can be written with lambda expression.

```
In [ ]: (lambda x: x + 10)(2)
```

```
In [ ]: double = lambda x: x + x
```

```
In [ ]: double(3)
```

1.1.3 Variable Scope

Contrast between global and local variable.

```
In [ ]: # Ordinary function that prints a global variable

        a = 10
        def whatever():
            print(a)
```

```
In [ ]: whatever()
```

```
In [ ]: # This way, a is treated as local variable withing the scope of the function

        def whatever_2():
            a = 3
            print(a)

        whatever_2()
        print(a)
```

```
In [ ]: # By defining with global keyword, the variable refers to the global variable

        def whatever_3():
            global a
            a = 100
            print(a)

        whatever_3()
        print(a)
```

1.2 Fundamentals of Object Oriented Programming

Concepts to be covered: 1. Class and objects, attributes and methods 2. Instance member and class member

```
In [ ]: class Person:
            pass
```

```
In [ ]: p = Person()
```

```
In [ ]: print(type(p))
```

1.2.1 Instance member

```
In [ ]: class Person:
            def __init__(self):
                self.name = ''
            def introduce(self):
                print('hi, I\'m', self.name)
```

```
In [ ]: p = Person()
        p.name = 'andi'
```

```
In [ ]: p.introduce()
```

1.2.2 Class member

```
In [ ]: class Persian:
            animal = 'cat'
            def __init__(self, catname):
                self.name = catname
            def introduce(self):
                print('meow, my name is', self.name)
```

```
In [ ]: c1 = Persian('doraemon')
        c2 = Persian('molly')
```

```
In [ ]: c1.introduce()
        c2.introduce()
```

```
In [ ]: print(c1.animal, c2.animal)
```

```
In [ ]: Persian.animal = 'CAT'
```

```
In [ ]: print(c1.animal, c2.animal)
```

1.2.3 Assignment 1.2.1

Write a 'BankAccount' class that contains the following attribute and operations

1. Account owner name and balance
2. Account type: can be standard or gold
3. Operation: deposit(amount), withdraw(amount), and transfer(account) For standard account, the maximum transfer is 5000000, while for gold account is 15000000

Example of use:

```
>>> b = BankAccount('Hadi', 1000000, 'g')
>>> b.deposit(3000000)
>>> b.withdraw(6000000) # unable to do the operation

error, not enough money
```

```
>>> b.print_info()

Hadi, 4000000, g

>>> n = BankAccount('Nana', 10000000, 's')
>>> n.transfer(b, 6000000)

error, standard account can only transfer 5000000 at maximum per transfer

>>> n.print_info()

Nana, 10000000, 's'

>>> n.transfer(b, 2000000)
>>> print(b.balance, n.balance)

6000000, 8000000
```

1.3 Custom module

This session describes how to use functions or classes in separate file.

```
In [ ]: import mymodule.mylibrary

In [ ]: mymodule.mylibrary.myfunction()

In [ ]: import mymodule.mylibrary as m

In [ ]: m.myfunction()

In [ ]: from mymodule.mylibrary import myfunction

In [ ]: myfunction()

In [ ]: from mymodule.mylibrary import myfunction as m

In [ ]: m()
```

1.4 Built in functions and classes

1.4.1 Date and Time

```
In [ ]: from datetime import date

In [ ]: help(date)

In [ ]: d = date.today()
        print(d)

In [ ]: print(d.year, d.month, d.day)
```

```
In [ ]: import datetime

        print(datetime.datetime.now())
        print(help(datetime.datetime))
```

1.4.2 Assignment 1.2.2

Write a program to compute someone's age based on birthday input
Example

```
>>> Your birthday
>>> year: 1989
>>> month: 10
>>> day: 31

You are 28.84 years old
```

1.4.3 Random number

```
In [ ]: import random

In [ ]: random.seed(0)
        print(random.randint(0,10), random.randint(0,10), random.randint(0,10), random.randint(0
```

1.4.4 Assignment 1.2.3

Write a program for Pauli test exercise. The program prints two number and takes the sum from user (one least significant digit only). The program stops when the user type 'exit'.