# Introduction to Programming W15

Introduction to Object-Oriented Programming in Python II.

2022/07/15

## Exercise 15.1

### The unoptimized healer

Create a child class `Healer` that inherits from the parent class `Adventurer`. The main idea is that if an `Adventurer` object needs healing and the `Healer` has medicine, it will try to restore the `Adventurer` to its maximum health. For more details, follow the instructional comments in the code. The `Adventurer` class is already complete and the only thing you need to add there is a function docstring for the `.needs_heal()` method. In the file header, you do not need to add a program description, because the explanations should be in the docstrings (student ID and name are still needed). So do not forget to write a class docstring and function docstrings for `Healer` (the only exception is the `.__init__()` method). Name the file in the following way, and submit to Manaba R+:
ip_ex151_<student id number>.py, for example: ip_ex151_012345678-9.py

```
from numpy import random

class Adventurer:
    """
    Instantiation of the Adventurer class defines an object with class
    attribute maximum health, and instance attributes name and (current) health.
    An Adventurer can introduce itself, and has an instance method that decides
    if the Adventurer needs healing or not.
    """
    max_health = 100

    def __init__(self, name):
        self.name = name
        self.health = random.randint(60,91)

    def intro(self):
        """
        Adventurer introduction.
        Returns a string.
        """
        return "Hello my name is " + self.name +"!"

    def needs_heal(self):
        """
        function docstring
        """
        if self.health < 80:
            print("I need healing!")
            return True
        else:
            print("I'm fine!")
            return False
```

```
class Healer(Adventurer):
    """
    The Healer class inherits from the...
    """

    def __init__(self, name, medicine_stock):
        #inherit name from the parent class __init__ constructor, and add an
        #additional instance attribute medicine_stock, where the value can be
        #assigned at object instantiation

    def healer_intro(self):
        """
        function docstring
        """
        #use the parent class .intro() method (and print)
        #additionally, print the Healer's current medicine_stock


    def heal(self, adventurer):
        """
        function docstring
        """
        heal_value = adventurer.max_health-adventurer.health

        #use conditionals in this order: if the medicine stock is 0 just print:
        #"I'm out of medicine, sorry!"
        #if the medicine stock is lower than the heal_value, print:
        #"I can only help this much, sorry!", and heal the adventurer as much as
        #possible, and set the medicine stock to 0
        #in every other case, print "Here you go, fully healed!", set the
        #adventurer's health the max_health, and reduce the medicine stock by
        #heal value

healer = Healer("Bob the Healer", 80) #feel free to experiment with other values
#create a list of adventurer names, for example Emma, Peter, Anna, Jim, John
adventurers =
#define an empty list called "healed_list"

#in a for loop, create Adventurer objects using the names from the list adventurers
for i in adventurers:
    print() #just prints an empty line, you don't need to change this
    #create the current Adventurer object in the loop
    adventurer =
    print(adventurer.intro()) #printing adventurer introduction
    #next, try to heal the adventurer it if it needs healing!
    if
        print("Health of", adventurer.name, "is", adventurer.health)
        healer.healer_intro() #healer introduction
        #next, (try to) heal the adventurer

        print("Health of " + adventurer.name + " now is", adventurer.health)
    #outside of the conditional, append the name and health of the current
    #adventurer to the healed_list as a tuple!

print("\n", healed_list) #just printing a newline and the healed list
```

Examples of expected output: