```
!date
```

```
Sun Sep  3 03:50:52 AM UTC 2023
```

Please run the above line to refresh the date before your submission.

CSCI-SHU 210 Data Structures

Recitation 1 Object-Oriented Programming Review

You should work on the tasks as written in the worksheet.

- For students who have recitation on Wednesday, you should submit your solutions by Friday 11:59pm.
- For students who have recitation on Thursday, you should submit your solutions by Saturday 11:59pm.
- For students who have recitation on Friday, you should submit your solutions by Sunday 11:59pm.

Name: Haochen Yang

NetID: hy2611

Please submit the following items to the Gradescope:

- URL: Your Colab notebook link. Click the Share button at the top-right, share with NYU, and paste to Gradescope
- PDF: The printout of your run in Colab notebook in pdf format

Topic 1 (Creating a class)

```
class Student:
    def __init__(self, name, age, GPA):
        self.name = name
        self.age = age
        self.GPA = GPA


    def get_GPA(self):
        return self.GPA

    def set_GPA(self, GPA):
        self.GPA = GPA
        return self


def main():
    bob = Student("Bob", 15, 3.0)
    print(bob.get_GPA()) #3.0

    bob.set_GPA(4.0)
    print(bob.get_GPA()) #4.0

if __name__ == '__main__':
    main()
```

```
 3.0
 4.0
```

What does the keyword self do in Python?

Topic 2 (underscore ***** functions):

```
class Pizza:
    def __init__(self, price):
        self.price = price

    def __add__(self, other):
```

```
        new_pizza = Pizza(self.price)
        new_pizza += other
        return new_pizza

    def __iadd__(self, other):
        self.price += other.price
        return self

    def __str__(self):  # str(self), self.__str__()
        return 'the price is, ' + str(self.price)

def main():
    pizza1 = Pizza(5)
    pizza2 = Pizza(6)
    p3 = pizza1 + pizza2 # pizza1.__add__(pizza2)
    print("p3,:", p3)
    pizza1 += pizza2
    print(pizza1)  # print(str(pizza1))

if __name__ == '__main__':
    main()
```

```
p3,: the price is, 11
the price is, 11
```

a) What does the code above print? Don't run the program, try to predict the output first.

p3,: the price is, 11 the price is, 11

b) Complete the following table, suppose the variable name is X. When will these underscore functions get called? Answer for 1st row has been given for your convenience.

Double-click to edit

1. X.__getitem__(self, index) : X[index]
2. X.__setitem__(self, index, value) : X[index] = value
3. X.__delitem__(self, index) : del X[index]
4. X.__add__(self, other) : X + other
5. X.__iadd__(self, other) : X += other
6. X.__eq__(self, other) : X == other
7. X.__len__(self) : len(X)
8. X.__str__(self) : str(X)
9. X.__repr__(self) : repr(X)
10. X.__contains__(self, value) : value in X
11. X.__iter__(self) : iter(X)

Topic 3 (Inheritance):

```
class Tree:
    def __init__(self, name, age):
        self._name = name
        self._age = age

    def get_name(self):
        return self._name


class Palm(Tree): # Palm(Tree) means, Palm inherits Tree.
    def __init__(self, name, age, color):
        # First you have to initialize the parent class. What should we write here?
        super(Palm, self).__init__(name, age)

        self._color = color

    def get_color(self):
```

```
        return self._color

def main():
    palm1 = Palm("Lucky", 30, "Green")
    print(palm1.get_name())  # What does this print (1)?
    print(palm1.get_color())  # What does this print (2)?
    tree1 = Tree("Funny", 20)
    print(tree1.get_name())  # What does this print (3)?
    print(tree1.get_color())  # What does this print (4)?

if __name__ == '__main__':
    main()
```

```
Lucky
Green
Funny
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-13-c72e3e610f15> in <cell line: 28>()
     27
     28 if __name__ == '__main__':
---> 29     main()

<ipython-input-13-c72e3e610f15> in main()
     24     tree1 = Tree("Funny", 20)
     25     print(tree1.get_name())  # What does this print (3)?
---> 26     print(tree1.get_color())  # What does this print (4)?
     27
     28 if __name__ == '__main__':

AttributeError: 'Tree' object has no attribute 'get_color'
```

SEARCH STACK OVERFLOW

What does the code above print? Don't run the program, try to predict the output first.

1. What is the output for print (1)? Lucky
2. What is the output for print (2)? Green
3. What is the output for print (3)? Funny
4. What is the output for print (4)? AttributeError: 'Tree' object has no attribute 'get_color'

Topic 4 (Misc):

```
# Coding 1

class Shape:
    def __init__(self, name):
        self.name = name

    def get_name(self):
        return self

class Circle:
    def __init__(self, name, radius):
        self.name = name
        self.radius = radius

    def calc_area(self):

        return float((self.radius)**2) * 3.14

    def calc_perimeter(self):

        return 2 * 3.14 * self.radius

class Rectangle:
    def __init__(self, name, width, height):
        self.name = name
        self.width = width
        self.height = height
```

```
    def calc_area(self):
        return self.width * self.height

    def calc_perimeter(self):
        return 2*(self.width + self.height)

def main():
    circle1 = Circle("fancy", 5)
    print(circle1.calc_area())  #78.5
    print(circle1.calc_perimeter())  #31.400000000000002

    rectangle1 = Rectangle("lucky", 3, 4)
    print(rectangle1.calc_area())  #12
    print(rectangle1.calc_perimeter())  #14

if __name__ == '__main__':
    main()
```

```
78.5
31.400000000000002
12
14
```

```
# Coding 2

class Polynomial:

    def __init__(self, coeffs):
        self.coeffs = coeffs

    def evaluate_at(self, x):
        result = 0
        for i in range(len(self.coeffs)):
            result += self.coeffs[i] * x**(len(self.coeffs)-1-i)
        return result

    def __add__(self, other):
        result_coeffs = []
        max_degree = max(len(self.coeffs), len(other.coeffs)) - 1
        for i in range(max_degree+1):
            result_coeffs.append(0)
        for i in range(len(self.coeffs)):
            result_coeffs[i] += self.coeffs[i]
        for i in range(len(other.coeffs)):
            result_coeffs[max_degree-i] += other.coeffs[i]

        return Polynomial(result_coeffs)

    def __iadd__(self, other):
        new_poly = self.__add__(other)
        self.coeffs = new_poly.coeffs
        return self

    def __str__(self):
        result = ""
        for i in range(len(self.coeffs)):
            term = ""
            if self.coeffs[i] != 0:
                if i == 0:
                    term += str(self.coeffs[i])
                else:
                    term += str(self.coeffs[i]) + "x^" + str(len(self.coeffs)-i-1)

            if i > 0:
                term += " + "
            result += term
        return result[:-3]


def main():
    # 1x^4 + 2x^3 + 3x^2 + 4x + 5
    coeffs = [1,2,3,4,5]
    poly = Polynomial(coeffs)
    print(poly.evaluate_at(2))   # 57
    print(poly.evaluate_at(3))   # 179
    print(poly)  # Outputs: 1x^4 + 2x^3 + 3x^2 + 4x^1 + 5
```

```
    print(poly)  # Outputs: 1x^4 + 2x^3 + 3x^2 + 4x^1 + 5

    # 4x^3 + 6x^2 + 8x^1 + 10
    coeffs = [4,6,8,10]
    poly2 = Polynomial(coeffs)
    print(poly2)  # Outputs: 4x^3 + 6x^2 + 8x^1 + 10
    poly += poly2
    print(poly)  # Outputs: 1x^4 + 6x^3 + 9x^2 + 12x^1 + 15

if __name__ == '__main__':
    main()
```

```
57
179
12x^3 + 3x^2 + 4x^1 + 5x^0
46x^2 + 8x^1 + 10x^0
112x^3 + 11x^2 + 10x^1 + 9x^0
```

Topic 5 Problem 1 Reverse Digit

```
#Given a 32-bit signed integer, return the reversed digits of this integer.
#Note:
#Try to solve this problem using math equations.
#Eg: don't cast this number to str/list/etc.


def reverse(x):
    if x < 0:
      symble = -1
      x = -x
    else:
      symble = 1

    result = 0
    while x:
      result = result * 10 + x%10
      x //= 10

    return result * symble


# test case
print(reverse(1200)) #21
print(reverse(123)) #321
print(reverse(-123)) #-321
```

```
21
321
-321
```

Topic 5 Problem 2

```
#Write a program to check whether a given number is a Funny number.
#Funny numbers are positive numbers whose prime factors only include 2, 3, 5.
#For example, 6, 8 are Funny while 14 is not Funny since it includes another prime factor 7.


def isFunny(num):
    if num == 1:
      return True
    if num <= 0:
      return False

    if num % 2 == 0:
      return isFunny(num/2)

    if num % 3 == 0:
      return isFunny(num/3)
```

```
    if num % 5 == 0:
      return isFunny(num/5)

    return False

# test case
print(isFunny(6)) #True
print(isFunny(8)) #True
print(isFunny(14)) #False
```

```
True
True
False
```