```
!date
```

```
Sat Jan 27 02:47:58 PM UTC 2024
```

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: MJ (Margad Gereltbaatar)

NetID: mg7502

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


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):
        if isinstance(other, Pizza):
          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. answer: first, it will print the price of object p3 as we definre the **STR** function to print the price of that object, second, it will print addition of the prices of each objects (pizza1 and pizza2), and will phrase it as "the price is, {price}"

Double-click (or enter) to edit

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?
        Tree.__init__(self, 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-4-184d568c39ce> in <cell line: 27>()
     26
     27 if __name__ == '__main__':
---> 28     main()

<ipython-input-4-184d568c39ce> in main()
     23     tree1 = Tree("Funny", 20)
     24     print(tree1.get_name())  # What does this print (3)?
---> 25     print(tree1.get_color())  # What does this print (4)?
     26
     27 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)? error, since the Tree class does not have the method get_color

Topic 4 (Misc):

```
# Coding 1

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

    def get_name(self):
        return self.name

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

```
        def calc_area(self):
            return 3.14*pow(self.radius,2)

        def calc_perimeter(self):
            return 2*3.14*self.radius

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

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

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

    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.get_name())
        print(rectangle1.calc_area())  #12
        print(rectangle1.calc_perimeter())  #14

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

```
78.5
31.400000000000002
lucky
12
14
```

```
    class Polynomial:
        def __init__(self, coeffs):
            self.coeffs = coeffs
            self.len = len(coeffs)

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

        def __iadd__(self, other):
            max_len = max(self.len, len(other.coeffs))
            self.coeffs = (max_len - self.len) * [0] + self.coeffs
            other.coeffs = (max_len - len(other.coeffs)) * [0] + other.coeffs

            for i in range(max_len):
                self.coeffs[i] += other.coeffs[i]

            self.len = max_len
            return self

        def __str__(self):
            res = []
            for i in range(self.len):
                coeff = self.coeffs[i]
                if coeff != 0:
                    power = self.len - 1 - i
                    if power > 0:
                        term = f"{coeff}x^{power}"
                    else:
                        term = f"{coeff}"
                    res.append(term)
            return " + ".join(res)

    def main():
        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

    coeffs = [4, 6, 8, 10]
    poly2 = Polynomial(coeffs)
    print(poly2)  # Outputs: 4x^3 + 6x^2 + 8x^1 + 10
    poly += poly2
    print(poly)  # Outputs: 5x^4 + 8x^3 + 11x^2 + 12x^1 + 5, this seems wrong as it adds coeffs with different powers

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

```
57
179
1x^4 + 2x^3 + 3x^2 + 4x^1 + 5
4x^3 + 6x^2 + 8x^1 + 10
1x^4 + 6x^3 + 9x^2 + 12x^1 + 5
```

Topic 5 Problem 1 Reverse Digit

```
from os import XATTR_CREATE
#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):
    res = 0
    sign = -1 if x < 0 else 1
    x *= sign
    while x != 0:
        last_dig = x%10
        x = x//10
        res = res * 10 + last_dig
    return res * sign


# 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):
    for i in [2,3,5]:
        while num % i == 0:
            num//=i
    return num == 1


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

```
True
True
False
```