How chr() works?

```
print(chr(97))
print(chr(65))
print(chr(1200))
```

Integer passed to chr() is out of the range

```
print(chr(-1))
```

When you run the program, ValueError is raised.
It's becauase the argument passed to the chr() method is out of the range.

Return Value from compile()

```
codeInString = 'a = 5\nb=6\nsum=a+b\nprint("sum =",sum)'
codeObejct = compile(codeInString, 'sumstring', 'exec')

exec(codeObejct)
```

Here, the source is in normal string form. The filename is sumstring. And, the exec mode later allows the use of exec() method.
The compile() method converts the string to Python code object. The code object is then executed using exec() method.

Create class method using classmethod()

```
class Person:
        age = 25

        def printAge(cls):
        print('The age is:', cls.age)
```

```
Person.printAge = classmethod(Person.printAge)

Person.printAge()
```

When do you use class method?

1. Factory methods

Create factory method using class method

```
from datetime import date

# random Person
class Person:
        def __init__(self, name, age):
        self.name = name
        self.age = age

        @classmethod
        def fromBirthYear(cls, name, birthYear):
        return cls(name, date.today().year - birthYear)

        def display(self):
        print(self.name + "'s age is: " + str(self.age))

person = Person('Adam', 19)
person.display()

person1 = Person.fromBirthYear('Arcot',  1985)
person1.display()
```

How class method works for inheritance?

```
from datetime import date

# random Person
class Person:
        def __init__(self, name, age):
        self.name = name
        self.age = age

        @staticmethod
        def fromFathersAge(name, fatherAge, fatherPersonAgeDiff):
        return Person(name, date.today().year - fatherAge + fatherPersonAgeDiff)

        @classmethod
        def fromBirthYear(cls, name, birthYear):
        return cls(name, date.today().year - birthYear)

        def display(self):
        print(self.name + "'s age is: " + str(self.age))

class Man(Person):
        sex = 'Male'

man = Man.fromBirthYear('Arcot', 1985)
print(isinstance(man, Man))

man1 = Man.fromFathersAge('Arcot', 1965, 20)
print(isinstance(man1, Man))
```

Return Value from complex()

```
z = complex(2, -3)
print(z)

z = complex(1)
print(z)

z = complex()
print(z)

z = complex('5-9j')
print(z)
```

Create complex Number Without Using complex()

```
a = 2+3j
print('a =',a)
print('Type of a is',type(a))

b = -2j
print('b =',b)
print('Type of b is',type(a))

c = 0j
print('c =',c)
print('Type of c is',type(c))
```

Return Value from delattr()

```
class Coordinate:
  x = 10
  y = -5
  z = 0

point1 = Coordinate()

print('x = ',point1.x)
print('y = ',point1.y)
print('z = ',point1.z)

delattr(Coordinate, 'z')

print('--After deleting z attribute--')
print('x = ',point1.x)
print('y = ',point1.y)


print('z = ',point1.z)
```

Deleting Attribute Using del Operator

```
class Coordinate:
  x = 10
  y = -5
  z = 0

point1 = Coordinate()

print('x = ',point1.x)
print('y = ',point1.y)
print('z = ',point1.z)


del Coordinate.z

print('--After deleting z attribute--')
print('x = ',point1.x)
print('y = ',point1.y)

print('z = ',point1.z)
```

Create Dictionary Using keyword arguments only

```
numbers = dict(x=5, y=0)
print('numbers = ',numbers)
print(type(numbers))

empty = dict()
print('empty = ',empty)
print(type(empty))
```

Create Dictionary Using Iterable

```
numbers1 = dict([('x', 5), ('y', -5)])
print('numbers1 =',numbers1)


numbers2 = dict([('x', 5), ('y', -5)], z=8)
print('numbers2 =',numbers2)
```

```
numbers3 = dict(dict(zip(['x', 'y', 'z'], [1, 2, 3])))
print('numbers3 =',numbers3)
```

Create Dictionary Using Mapping

```
numbers1 = dict({'x': 4, 'y': 5})
print('numbers1 =',numbers1)

numbers2 = {'x': 4, 'y': 5}
print('numbers2 =',numbers2)

numbers3 = dict({'x': 4, 'y': 5}, z=8)
print('numbers3 =',numbers3)
```