Η μέθοδος __str__

Είδαμε ότι όταν καλούμε τον κατασκευαστή μίας κλάσης, η __init__ καλείται αυτόματα. Το παρακάτω παράδειγμα χρησιμοποιεί την κλάση Student. Εμφανίζει ένα μενού επιλογών και δέχεται δύο επιλογές: “a” για να προσθέσει ένα αντικείμενο Student σε λίστα, τα στοιχεία του οποίου διαβάζει από το πληκτρολόγιο και “l” για να παρουσιάσει τα στοιχεία της λίστας. Με "x" σταματάει.

Την εκτύπωση των στοιχείων την κάνουμε εδώ καλώντας την μέθοδο printStudent() για κάθε αντικείμενο της λίστας.

```
# Μενού που γεμίζει λίστα με αντικείμενα

class Student:
    def __init__(self, am, name):
        self.am=am
        self.name=name
    def printStudent(self):
        print(self.am,self.name)

slst=[]                 # lista students
print('a: add student') # menu
print('l: list students')
print('x: exit')
s=input("select: ")
while s!='x':
    if(s=='a'):         # prosthese se lista
        am=input('am: ')
        name=input('name: ')
        stu=Student(am,name)
        slst.append(stu)
    elif(s=='l'):       # typwse lista
        for s in slst:
            s.printStudent()
    print('a: add student') # menu
    print('l: list students')
    print('x: exit')
    s=input("select: ")
```

Εναλλακτική λύση είναι να αντικαταστήσουμε την printStudent με την μέθοδο __str__:

```
class Student:
    def __init__(self, am, name):
        self.am=am
        self.name=name
    def __str__(self):
        return '*** AM:' + self.am + ', ' + self.name
```

Η μέθοδος αυτή προφανώς δεν τυπώνει, χρησιμοποιείται όμως για να μετατρέπει το αντικείμενο σε String, και καλείται αυτόματα όταν προσπαθούμε να το τυπώσουμε απευθείας:

```
    ...
    elif(s=='l'):       # typwse lista
        for s in slst:
            print(s)
    ...
```

Εάν προσπαθήσουμε να τυπώσουμε απευθείας ένα αντικείμενο χωρίς να υπάρχει __str__ στην κλάση, τότε τυπώνεται ένα μήνυμα που περιγράφει το αντικείμενο και δίνει την διεύθυνση της μνήμης στην οποία βρίσκεται, πληροφορία σχετικά άχρηστη για τον τελικό χρήστη:

```
<__main__.Student object at 0x00000277D2D12D30>
```

Το ίδιο πρόγραμμα με χρήση __str__:

```
# Με χρήση __str__ στην κλάση Student

class Student:
    def __init__(self, am, name):
        self.am=am
        self.name=name
    def __str__(self):
        return 'AM: '+self.am+' Name: '+self.name

slst=[]                 # lista students
print('a: add student') # menu
print('l: list students')
print('x: exit')
s=input("select: ")
while s!='x':
    if(s=='a'):         # prosthese se lista
        am=input('am: ')
```

```
        name=input('name: ')
        stu=Student(am,name)
        slst.append(stu)
    elif(s=='l'):       # typwse lista
        for s in slst:
            print(s)
    print('a: add student') # menu
    print('l: list students')
    print('x: exit')
    s=input("select: ")
```

Τι είναι η μεταβλητή __name__ και πως χρησιμοποιείται.

Η μεταβλητή __name__ είναι μία global μεταβλητή που χρησιμοποιείται για να βρίσκουμε εάν το αρχείο που τρέχουμε περιέχει το κεντρικό πρόγραμμα μας. Σε αυτή την περίπτωση η τιμή της είναι '__main__'. Αλλιώς έχει το όνομα του αρχείου. Για παράδειγμα όταν τρέξουμε το nametest.py:

```
import foo

print(__name__)
foo.bar()
```

Το οποίο καλεί την συνάρτηση bar() του αρχείου foo.py:

```
def bar():
    print('function bar in foo.py here!')
    print(__name__)
```

Το αποτέλεσμα της εκτέλεσης του nametest.py είναι το παρακάτω:

```
__main__
function bar in foo.py here!
foo
>>>
```

Στην πρώτη γραμμή βλέπουμε ότι η τιμή της __name__ στο κύριο πρόγραμμα είναι __main__ και όχι nametest. Οι δύο επόμενες γραμμές δημιουργούνται από την συνάρτηση bar του αρχείου foo.py, το όνομα του οποίου τυπώνεται από την μεταβλητή __name__.

Η ιδιότητα αυτή της μεταβλητής __name__ χρησιμοποιείται συχνά για να βάλουμε στα προγράμματα μας μία main, όπως έχουμε στη C και την Java. Προσθέτουμε την συνάρτηση main

στην προηγούμενη έκδοση του προγράμματος με όλη την λογική της εφαρμογής. Ελέγχουμε στην συνέχεια εάν το πρόγραμμα μας κλήθηκε από τον χρήστη για να τρέξει και καλούμε την main(). Αυτό γίνεται έτσι:

```
class Student:
    def __init__(self, am, name):
        self.am=am
        self.name=name
    def __str__(self):
        return 'AM: '+self.am+' Name: '+self.name

def main():
    slst=[]                 # lista students
    print('a: add student') # menu
    print('l: list students')
    print('x: exit')
    s=input("select: ")
    while s!='x':
        if(s=='a'):         # prosthese se lista
            am=input('am: ')
            name=input('name: ')
            stu=Student(am,name)
            slst.append(stu)
        elif(s=='l'):       # typwse lista
            for s in slst:
                print(s)
        print('a: add student') # menu
        print('l: list students')
        print('x: exit')
        s=input("select: ")

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

Στην επόμενη εκδοχή του προγράμματος μας (Student3.py), έχουμε κατασκευάσει μια δεύτερη κλάση, την School και έχουμε τοποθετήσει σε αυτή τη λίστα με τους Students σαν Class Variable. Σαν Class Variable είχαμε παλαιότερα δηλώσει το PI και τον μετρητή count στη σειρά παραδειγμάτων με την κλάση Sphere. Είχαμε πει ότι οι μεταβλητές που ανήκουν στην κλάση υπάρχουν πάντα και μόνο μία φορά, ακόμα και όταν δεν κατασκευάσουμε κανένα αντικείμενο της κλάσης. Η διαδικασίες προσθήκης και εκτύπωσης των φοιτητών έχουν επίσης μεταφερθεί στην κλάση School.

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

```
        self.name=name
    def __str__(self):
        return 'AM: '+self.am+' Name: '+self.name

class School:
    slst=[]         # lista students
    def addStudent(self):
        am=input('am: ')
        name=input('name: ')
        stu=Student(am,name)
        School.slst.append(stu)
    def listStudents(self):
        for s in School.slst:
            print(s)


def main():
    sc=School()
    print('a: add student') # menu
    print('l: list students')
    print('x: exit')
    s=input("select: ")
    while s!='x':
        if(s=='a'):         # prosthese se lista
            sc.addStudent()
        elif(s=='l'):       # typwse lista
            sc.listStudents()
        print('a: add student') # menu
        print('l: list students')
        print('x: exit')
        s=input("select: ")

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

Στην τελευταία εκδοχή του προγράμματος έχουμε προσθέσει κατασκευαστή στην κλάση School και η λίστα μας έχει γίνει instance variable, με συνέπεια να μπορούμε εάν κατασκευάσουμε πολλά αντικείμενα School, το κάθε ένα να έχει τη δική του λίστα φοιτητών. Ο μετρητής που έχουμε βάλει στην κλάση Student είναι Class Variable και τον ζητάμε καλώντας την Student.getCount():

```
# With School constructor and Student count

class Student:
    cnt=0
    def __init__(self, am, name):
```

```
        self.am=am
        self.name=name
        Student.cnt+=1
    def __str__(self):
        return 'AM: '+self.am+' Name: '+self.name
    def getCount():
        return Student.cnt

class School:
    def __init__(self):
        self.slst=[]          # lista students
    def addStudent(self):
        am=input('am: ')
        name=input('name: ')
        stu=Student(am,name)
        self.slst.append(stu)
    def listStudents(self):
        for s in self.slst:
            print(s)

def main():
    sc=School()
    print('a: add student') # menu
    print('l: list students')
    print('c: count students')
    print('x: exit')
    s=input("select: ")
    while s!='x':
        if(s=='a'):          # prosthese se lista
            sc.addStudent()
        elif(s=='l'):        # typwse lista
            sc.listStudents()
        elif(s=='c'):
            print(Student.getCount())
        print('a: add student') # menu
        print('l: list students')
        print('c: count students')
        print('x: exit')
        s=input("select: ")

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