Python

Introduction to Functions, Arrays, Enumerators and Structures

User defined functions

Definition

```
def fname(formal parameter list):
    body of function
    composed of indented
    statements
    return r_expr
```

Usage (calling or invoking functions)

```
fname(actual parameter list)
```

can be used in expressions

Function's definition

fname: name of function

parameter list: comma separated list of variable names

return: reserve word

(ends execution of functions body)

rexpr: expression to return to caller statement

return line may be omitted completely or rexpr is not mandatory (None return is both cases)

Functions

```
// to compute and return average
// of three integer parameters
def average(a, b, c):
      sum = a+b+c;
      avg = sum/3;
      return avg;
```

Using a function

```
print(average(23, 7, 6))
```

Arrays

var = [initval] * size

or

var = [expr_list]

var is array name, size is size of the array, and initval is the ***initial value*** filled in each cell of array

Each cell is accessible with an integer index starting from 0 to size-1, e.g., to access the 3rd cell of array named var, we use var[2]

expr_list is list of comma separated expressions

Example

```
# to compute and return average of
# whole integer array with N values

def average(a, N):
      sum = 0
      j = 0
      while j < N:
            sum = sum+a[j]
            j = j+1

      avg = sum/N
      return avg
```

```
def main():
      v = [5,3,4,7,4]
      av = average(v,5)
      print(av)
      return
```

User defined types (UDTs) (classes)

Enumerators

A list of values having unique type, e.g., PucitDegree, RGBColor, Gender, Coins, etc

Degree can be from *SE, CS, IT, DS*

Coin can be from *penny=1, dime=10, quarter=25, dollar=100*

Better to use classes for enums

Structures

A composite data type to store more than one value, accessible by component name rather index as in array.

```
class Student:
        pass
```

have rollno, name, cgpa, etc as components

Enumerators

```
class Gender:
   Male = 1
   Female = 2


def main():

      gn = Gender.Male  # or Female

      if gn == Gender.Male:
            print("Male")
      else:
            print("Female")
```

Structures

```
class Box:
      pass
```

or

```
class Box:
      height = 0
      width = 0
      depth = 0

def main():
      b = Box()
      b.height = 3
      b.width = 5
      b.depth = 2

      int v = b.height * b.width * b.depth

      print("Volume of box is", v)
```

Functionalities for UDTs

The UDTs, being user defined were not known to Python language makers, so almost NO functionality is available in Python for UDTs, including input, output, comparison, arithmetic operator and other functions whatever is applicable to them.

So, being user defined, programmers has to build that required functionality through creation of several functions.

Exercise: build a **vector** UDT, **rational** UDT, etc

Program Composition

Data

- Values used in statements.
- Variables/Arrays of
  - Built-in type
  - Enumeration type
  - Structures type
  - Mixture of above
  - Other data types that may not discuss in PF

- *Simple variables*
- *Simple Arrays*
- *Simple Structures and enumerator*
- *Their combinations*
  - *Arrays of structures*
  - *Array as a structure component*
  - *Structure as structure component*
  - *Etc, etc*

Code

Statements or instructions like

- Definitions,
- Assignment operation,
- Output,
- Input,
- Function call
- if, if-else
- while
- etc

TOKENS
SYNTAX

Pick the PACE

I am not asking you to get READY, you are supposed to be READY, by now.

I am not giving a **kick** to you, you have to do it

- By yourself
- For yourself

- Do '**die hard**' work with **honesty** and dedication
- Put **ego** aside and ask questions
- Avoid friends seated nearby (or possibly **tape your mouth** :-)