Python

Scope and lifetime of variable, named constants, and exceptions, Arrays, Enumerators and Structures

Variables – Scope and Lifetime

- Global variables
  - Avoid them (almost) in all cases
- Automatic or Local variables
  - Functions parameters
  - Defined with a block
- Scope
  - Accessible within the block, they are defined
- Lifetime
  - Created when first time assigned a value
  - Exists till execution exits form their scope's blocks

Named constants

Just Variables in Python

- Name
  - Usually CAPITALIZED/Global
- Helps in avoiding magic numbers

- Examples
  - PI = 3.14159265358979323846
  - DATASIZE = 100
  - KM_PER_MILE = 1.60934
  - BEEP = “\a”

Situation (exceptional cases)

Function that divide first parameter with second

```
def divide(num, den):
      if (den == 0):
            // what code should be here
      return num/den
```

Solutions?

- If through coding, exceptional case can be handled, only then they should be dealt.
- Printing messages, exiting, returning special values may lead other problems

Exceptional case

- Handling not possible
  - If handing of exceptional case is not locally possible, then it is better to raise as Exception, that may lead the caller function to catch it and handle if possible.
  - raise Exception(str_expr)

```
def divide(num, den):
    if (den == 0):
        raise Exception("Den 0")
    return num/den
```

try, except, else and finally

```
try:
    print(divide(45, 5))  # (4, 0)
except Exception as e:
    print("in except, div zero", e)
else:
    print("in else, div zero")
finally:
    print("in finally, div zero")
```

https://www.tutorialspoint.com/python-exception-base-classes

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** :-)