UNIT-V

Classes and Functions:

Time: We will define a class called **Time** that records the time of day.

The class definition looks like this:

Class Time:

“”Represents the time of day.

Attributes: hour , minute , second””.

Pure Functions:

Two kinds of functions: **Pure functions and Modifiers or prototype and patch.**

The function creates a new Time object , initializes its attributes,and returns a reference to the new object. This is called **purefunction**

Because it does not modify any of the objects passed to its arguments and it as argument and it has no effect,like displaying a value or getting user input,other than returning a value.

Here is simple prototype of **add_time:**

```
def add_time(t1,t2):
    sum=Time()
    sum.hour=t1.hour+t2.hour
    sum.minute=t1.minute+t2.minute
    sum.second=t1.second+t2.second
    retutn sum
```

Modifiers:

Sometimes it is useful for a function to **modify** the objects it gets as Parameters.

Whenever the changes are **visible** to the caller. Such functions that work this way are called **modifiers.**

❖ Increments, which adds a given numberto seconds to a Time object , can be written naturally as a modifier.

Here is the rough draft:

```
def  increment(time,seconds):
    time.second+=seconds
    if time.seconds>=60
    time.second-=60
```

Prototyping versus planning:

❖ An alternative is desinged develpoment, in which high-level insight into the problem can make the programming much easier. In this case, the insight is that a Time object is really a three digit number in base 60.

❖ Here is a function that converts Time to integers:

```
def  time_to_int(time)
    minutes=time.hour*60+time.minute
    seconds=minutes*60+time.second
    return seconds
```

Classes and methods:

Object oriented features:

Python is an object oriented programming language,which means that

It provides features that support object-oriented programming,which

Has these defining characteristics:

- Programs include class and method definitions.
- Most of the computation is expressed in terms of operation on objects.
- Objects often represent things in the real world,and methods often correspond to the ways things in the real world interact.]

Printing objects:

To make print_time a method, all we have to do is move the function definition inside the class definition.

```
Class Time:
    def print_time(time):
    print('%.2d:%.2d:%.2d'%(time.hour,time.minute,time.second))
```

Now there are two ways to call print_time.

1.Funtion syntax:

>>>Time.print_time(start)

09:45:00

2.Method syntax:

>>>start.print_time()

09:45:00

The _ _init_ _ method:

The init method is a special method that gets invokrd when an object Is initiallied. Its full name is _ _init_ _(two underscore characters, followed by init,and then two more under scores).

The _ _str_ _ method:

_ _str_ _ is a special method,like _ _init_ _ that is supposed to return a string reprentation of an object.

Operator Overloading:

Operator Overloading means giving extended meaning beyond their predefined operational meaning .For example operator+ is used to add two integers as well as join two strings and merge two lists.

```
Print(1  + 2)
# add  two  numbers
Print("Python" + "For")
# concatenate two strings
```

Type based dispatch:

The built-in function is instance takes a value and a class object, and returns True if the value is an instance of the class.

Polymorphism :

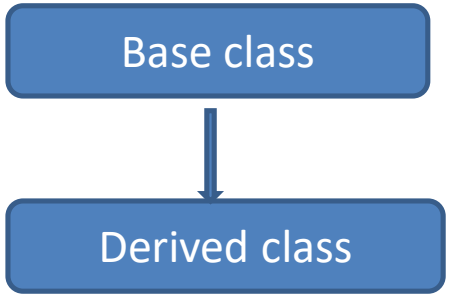
The word polymorphism means having many forms. In programming , polymorphism means the same function name being used for different types.

Interface and Implementation:

An interface is a collection of method signatures that should be provided by the implementing class. Implementing an interface is a way of writing an organised code and achieve abstraction.

Inheritance:

- Classes can inherit functionally of other classes.
- If an object is created using a class that inherits from a superclass.
- The object will contain the methods of both the class and the super class.
- The same holds true for variables of both the superclass and the class that inherits from the super class.



Card Objects:

- If we want to define a new object to represent a playing card,it is obvious what the attributes should be : rank and suit.
- One possible is to use strings containing words like ‘spade’for suits and ’Queen’ for ranks. One problem with this implementation is that it would not be easy to compare cards.
- An alternative is to use integers to encode the ranks and suits.

Class attributes:

Variables like suit_names and rank_names, which are defined inside a class but outside of any method , are called class attributes because they are associated with the class Object card.

Comparing Cards:

For built –in types, there are relational operators(<,>,=,etc.) that compares values
And determine which one is greater than,less than or equal to another.

Decks:

- Now that we have cards, the next step is to define Decks. Since a deck is made

up of cards, it is natural for each Deck to contain a list of cards as an attribute.

- The init method creates the attributes cards and generates the standard set of fifty-two cards.

Printing the deck:

This method demostrates an efficient way to accumulate a large string: building a
list of strings and then using the string method join. The built-in function str
invokes the _ _ str_ _method on each card an returns the string representation.

Add,remove,shuffle&sort:

- To add a card, we can use the list method append
- POP removes the last card in the list.
- We can write a deck method named shuffle usingthe function shuffle from the random module.
- A deck method named sort that uses list method sort to sort the cards in Deck.

Data encapsulation:

It describes the idea of wrapping data and the methods that work on data within one
unit . This puts restrictions on accesing variables and methods directly and can prevent the
accedental modification of data

The Goodies:

Conditional expressions:

Ternary operators are also known as conditional expressions are operators that evaluate something based on a conditional being true or false.It simply allows testing a condition in a single line replacing the multiline if –else making the code Compact.

Example program:

```
a, b = 10 ,20
min= a if a < b else b
print(min)
```

Output:10

List Comprehension:

List Comprehensions are used for creating new lists from other iterables like tuples, Strings,arrays,lists,etc. It consists of brackets containing the expression, which is Executed for each element along with the for loop to iterate over each element.

Generator Expressions:

Generators are written just like a normal function but we use yeild() instead of return() for returning a result. It is more powerful as a tool to implement iterators.

Any and all:

- Any returns true if any of the items is true. It returns false if empty or all are false.
- All return true if all the items are True. All can be thought of as a sequence of AND Operation on provides iterables. Stops Execution as soon as the result is known.

Sets:

A set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Set classes represent the mathematical notation of an set. It has a highly optimized method for checking whether a specific elemeny is contained in the set. It is based on data structures known as a hash table.

Counters:

The collection modules has container datatypes and contains many useful data structures that you can use to store information in memory. Athe counter keeps the count of the number of items in the container.

Defaultdict:

Defaultdict is a sub class of the dictionary class that returns a dictionary like object. The functionality of both dictionaries and default are almost same except for the fact that defaultdict never raises a keyError. It provides a default value for the key that does not exists.

Named Tuples:

Python supports a type of container like dictionaries called “namedtuple()”present in module”, “collections”.

Gathering keyword Args:

The special syntax * args is function definitions in python is used to pass a variable number of arguments to a function.