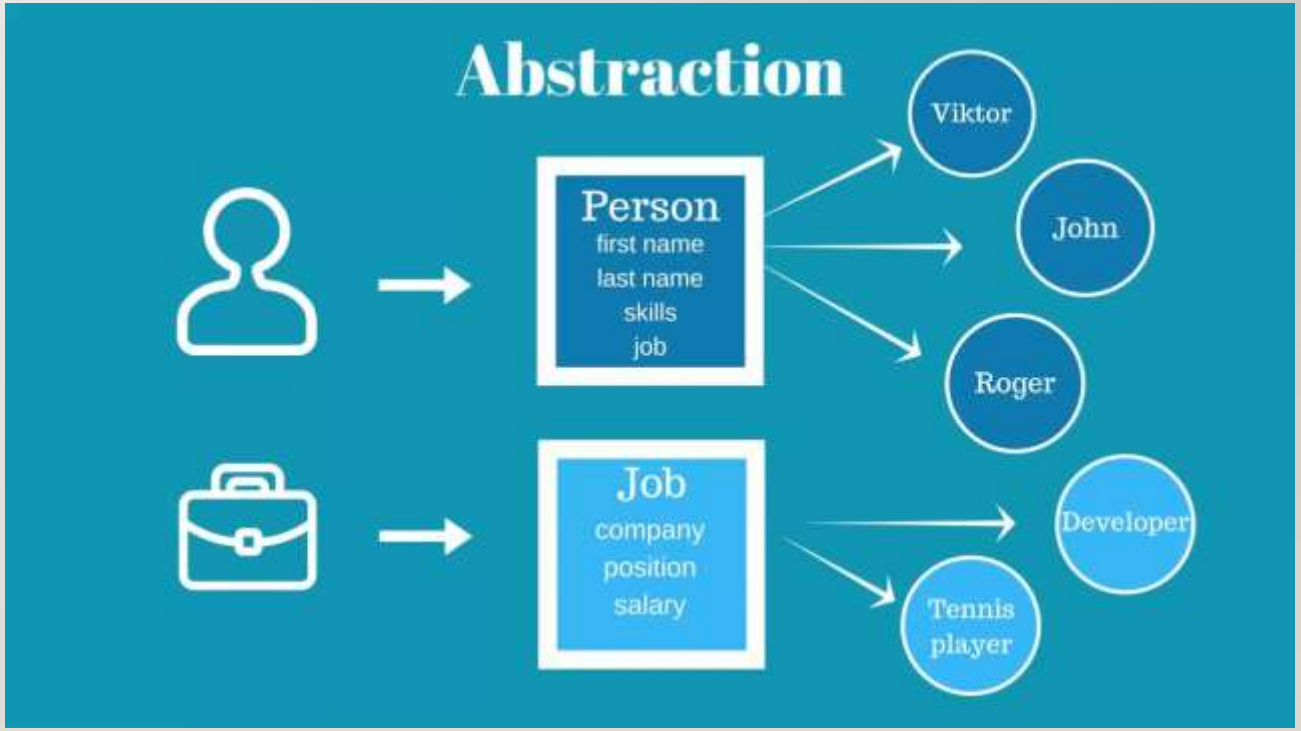
UNDERSTANDING PYTHON

OBJECT ORIENTED PROGRAMMING

- OOP is a type of programming which revolves around the concept object having some attributes and behaviors.
- Basic OOPs concepts:
  - Classes
  - Abstraction
  - Encapsulation
  - Polymorphism
  - Inheritance

ABSTRACTION

- The process of showing the essential details while hiding the unnecessary information reducing the programming complexity.



PYTHON EXAMPLE

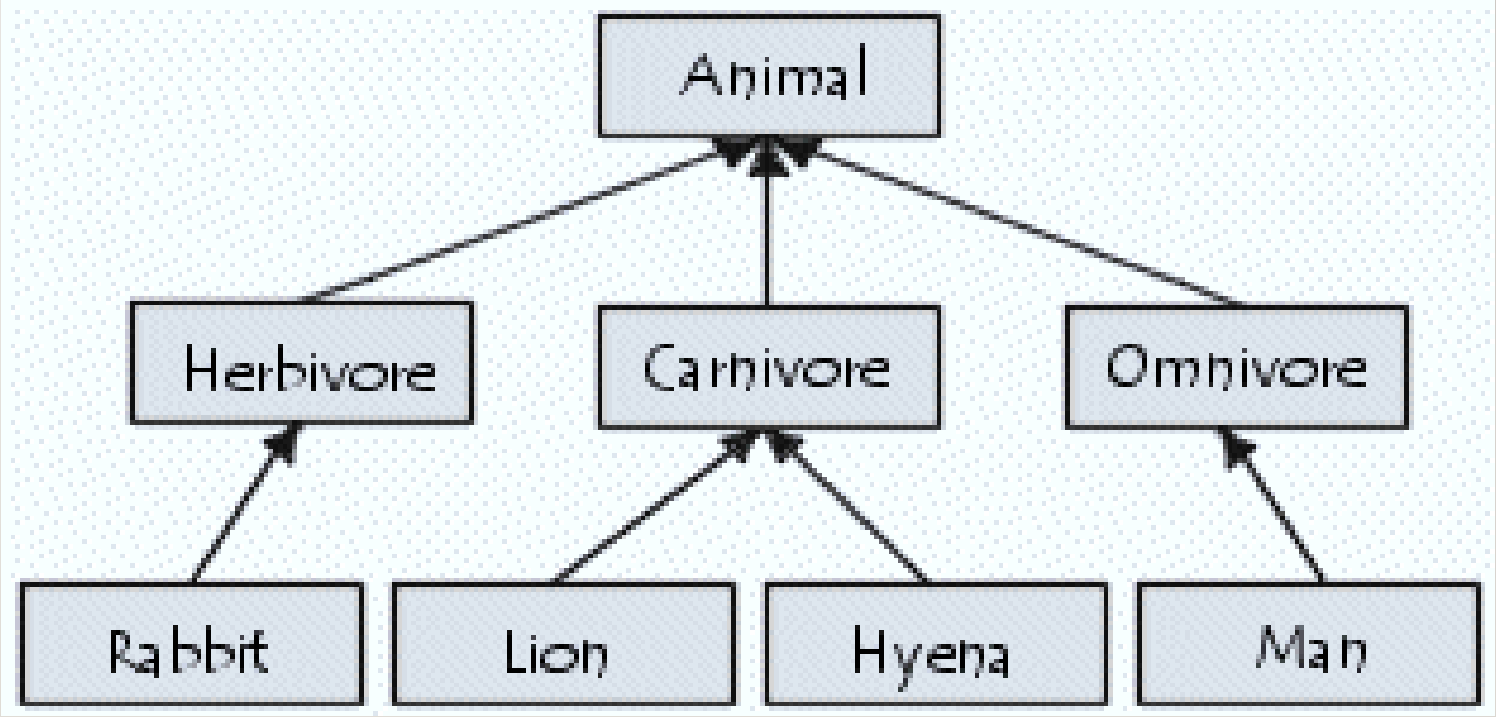
```
import abc
from abc import ABC, abstractmethod

class R(ABC):
    def rk(self):
        print("Abstract Base Class")

class K(R):
    def rk(self):
        super().rk()
        print("subclass ")

r = K()
r.rk()
```

Output:

```
Abstract Base Class
subclass
```

INHERITANCE

- It is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods.



PYTHON EXAMPLE

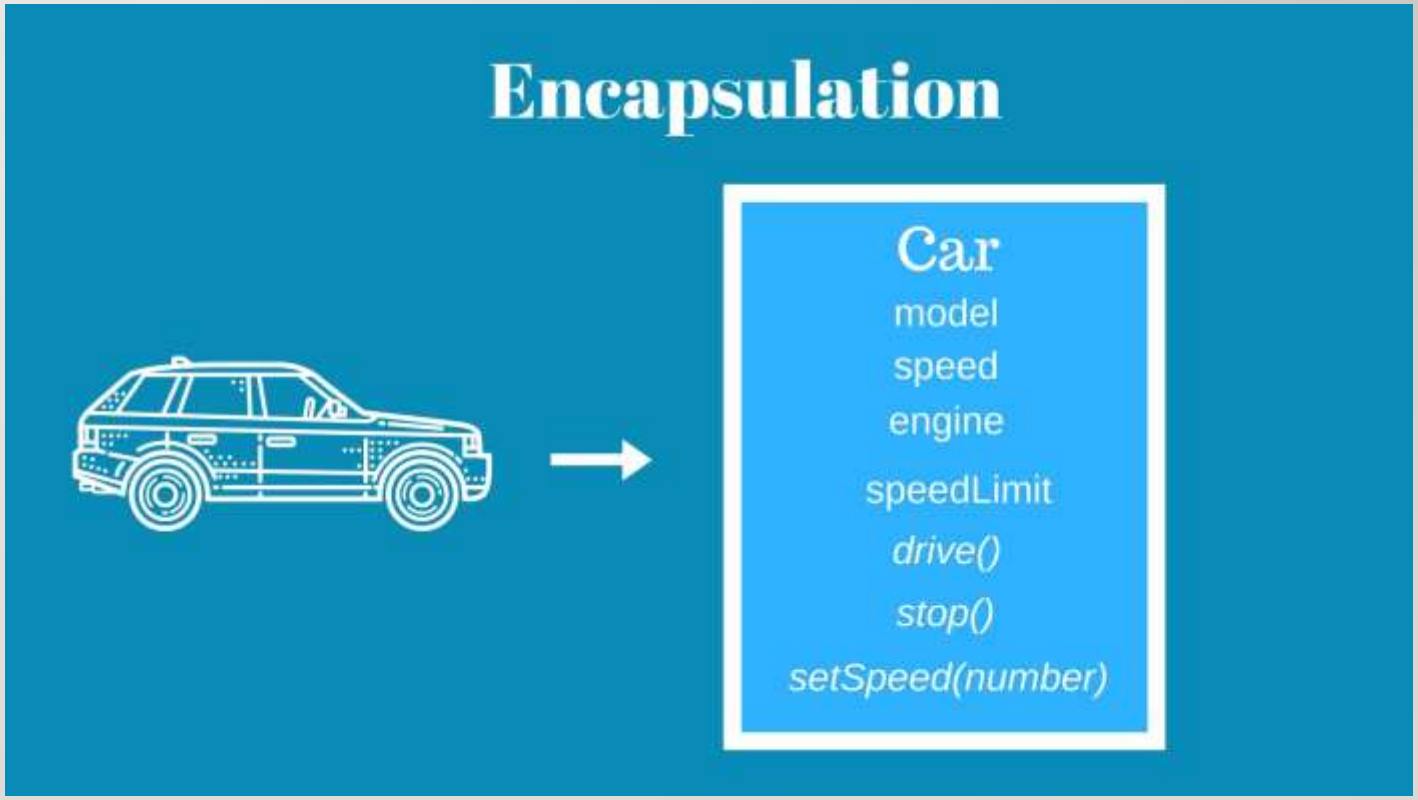
```
class Person(object):
    def __init__(self, name):
        self.name = name
    def getName(self):
        return self.name

    def isEmployee(self):
        return False

class Employee(Person):

    def isEmployee(self):
        return True

emp = Person("Anshu")
print(emp.getName(),
emp.isEmployee())

emp = Employee("Amit")
print(emp.getName(),
emp.isEmployee())
```

Output:

```
Anshu False
Amit True
```

ENCAPSULATION

- It describes the idea of bundling data and methods that work on that data within one unit, e.g., a class in Java.



PYTHON EXAMPLE

```
class Base:
   def __init__(self):

      self._a = 2

class Derived(Base):
   def __init__(self):

      Base.__init__(self)
      print("Calling protected member of
base class: ")
      print(self._a)

obj1 = Base()
print(obj1.a)
obj2 = Derived()

Output:
Calling protected member of base class:
2
Traceback (most recent call last):
  File
"/home/6fb1b95dfba0e198298f9dd02469e
b4a.py", line 25, in
   print(obj1.a)
AttributeError: 'Base' object has no
attribute 'a'
```

STRUCTURAL DATA TYPES IN PYTHON

- **String**: A string value is a collection of one or more characters put in single, double or triple quotes.
- **List** : A list object is an ordered collection of one or more data items, not necessarily of the same type, put in square brackets.
- **Tuple**: A Tuple object is an ordered collection of one or more data items, not necessarily of the same type, put in parentheses.
- A dictionary object is an unordered collection of data in a key : value pair form, enclosed in curly brackets.

Ex.
String : "Gurvansh", "I work in the field of ML".

list: ['a','c','d'] , ['I', 'work', 'in', 'the','field', 'of' 'ML']

tuple: (1,2,3,4) , ('a','d','f')

dictionary: {1: "Amit", 2:"Anshu", 3:"Gurvansh"}

INPUT IN PYTHON

- To take input from the user we use: 'input()' function.
- The input() function has a limitation to take all the inputs as string type, therefore it is used with implementation of other function to convert the data type of the input.

Ex.

```
a = input()
type(a)

output: str

b = int(input())
type(b)

output: int
```

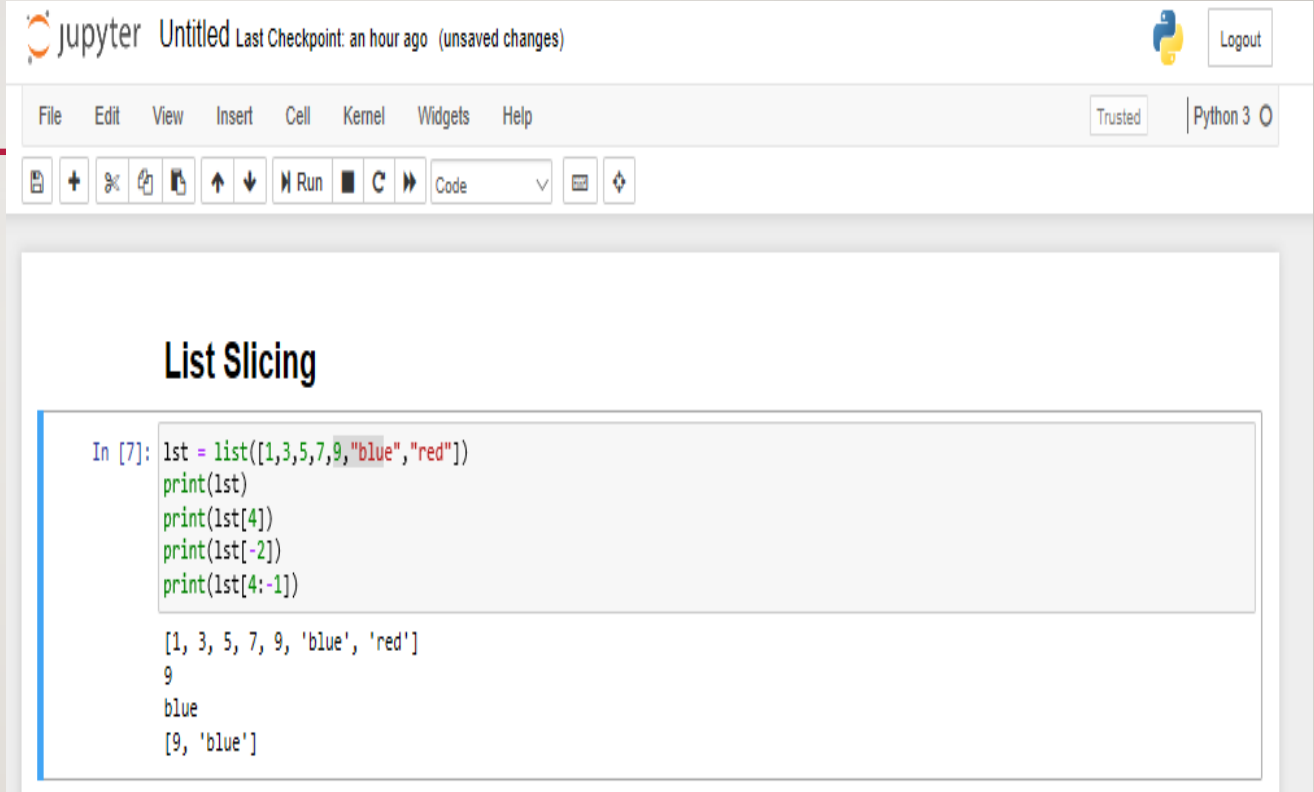
WORKING WITH LIST IN PYTHON(INPUT)

- To take input in the list the input() function is used with conjunction with looping statements and the following functions:
  - Append()
  - Map()
  - Split()

```
lst = []
n = int(input("Enter number of elements : "))

for i in range(0, n):
    ele = int(input())

    lst.append(ele)
print(lst)

output:
Enter the number of elements : 5
23
44
75
8
2
[23, 44, 75, 8, 2]
```

LIST (SLICING)

- Slicing in a list done with the help of index number.
- The index number in python can be read in two ways i.e. positive(start – end) or negative(end – start) and are written in the [] brackets after the name of the list variable.



THANK YOU