

OBJECT ORIENTED PROGRAMMING WITH PYTHON

Second Class

1st Semester

Bug-busting checklist

Exceptions

Procedural programming concept

Object Oriented Programming Concepts

Benefits of OOP

BUG-BUSTING CHECKLIST

Sometimes you might think that you'll never get a program to work, but don't give up! If you follow the tips in this handy checklist, you'll be able to identify most errors.

Ask yourself...

- If you build one of the projects in one of the books and it doesn't work, check that the code you've typed matches the book exactly.
- Is everything spelled correctly?
- Do you have unnecessary spaces at the start of a line?
- Have you confused any numbers for letters, such as 0 and O?
- Have you used upper-case and lower-case letters in the right places?
- Do all open parentheses have a matching closing parenthesis? () [] { }
- Do all single and double quotes have a matching closing quote? ' ' " "
- have you checked the name of the python file that you are currently running?

EXCEPTIONS

Exceptions are errors that crash our programs. They often happen because of bad input or programming errors. It's our job to anticipate and handle these exceptions to prevent our programs from crashing.

```
age = int(input("Enter your age: "))  
print(age)
```

Enter your age : 25
25

Process finished with exit code 0

Enter your age : abc



ValueError: invalid literal for int() with base 10: "abc"
Process finished with exit code 1

EXCEPTIONS

As a good programmer you should anticipate this situation and handle it like printing a proper message ..to manage the pervious example use (try.....except) as follows:

```
try:
    age = int(input("Age: "))
    print(age)
except ValueError:
    print("Not a valid number")
```

Notice that we have to use the same error message that appears (**ValueError**)

EXCEPTIONS

If we have a mathematical operations in our code like a DIVISION we have to think about the division by zero error

try:

```
age = int(input("Age: "))
```

```
income = 20000
```

```
risk = income / age
```

```
print(age)
```

except ValueError:

```
print("Not a valid number!")
```

Enter your age : 0

ZeroDivisionError: division by zero

Process finished with exit code 1

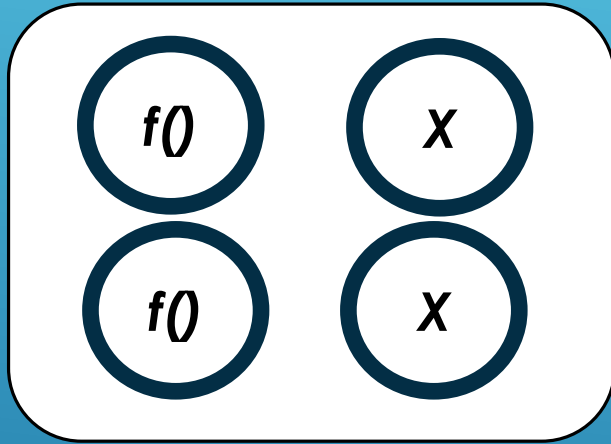
EXCEPTIONS

To manage this error, first copy the Error name then add it to the exception statement

```
try:
    age = int(input("Age: "))
    income = 20000
    risk = income / age
    print(age)
except ValueError:
    print("Not a valid number")
except ZeroDivisionError:
    print("Age cannot be 0")
```

PROCEDURAL PROGRAMING CONCEPT

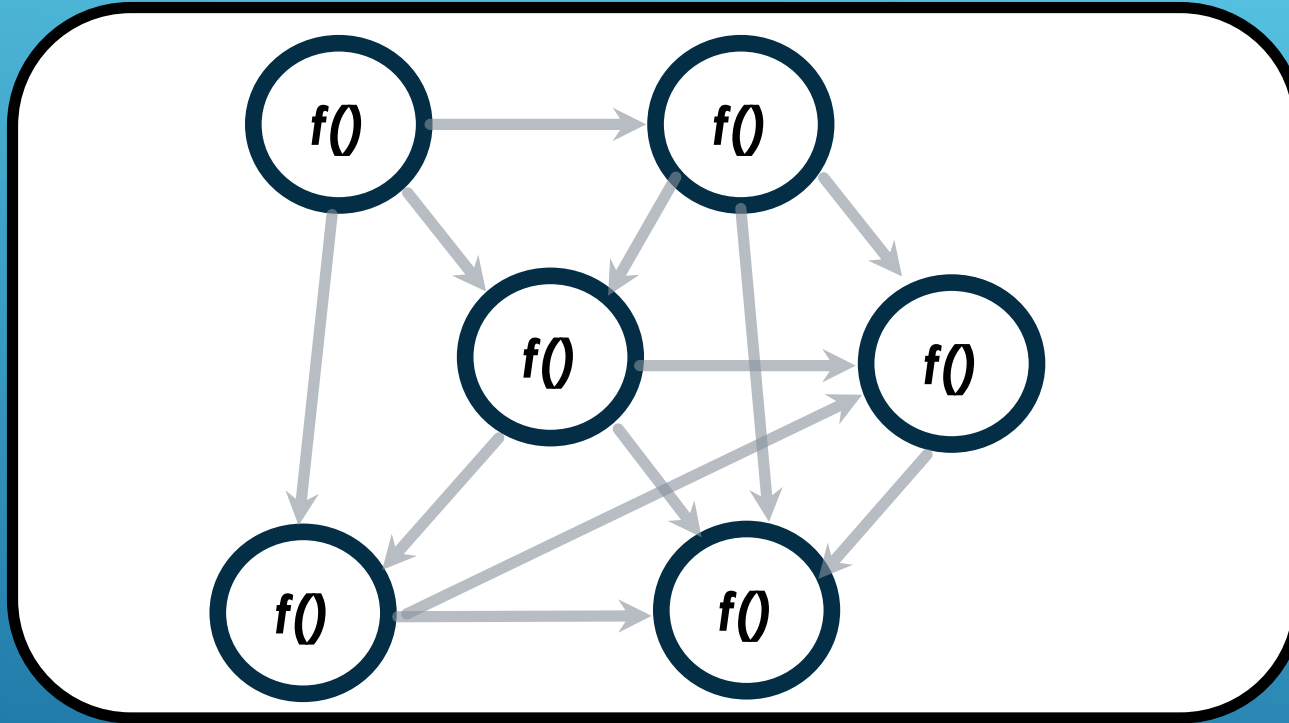
- ▶ In procedural programming, the program divided into a set of function, so we have data stored in a set of variables and functions that operate on the data.



- ▶ This style of programming is very simple and straightforward, but after your programs grow it will end up with a lot of functions .
- ▶ You might find yourself coping and pasting lines of code over and over

PROCEDURAL PROGRAMING CONCEPT

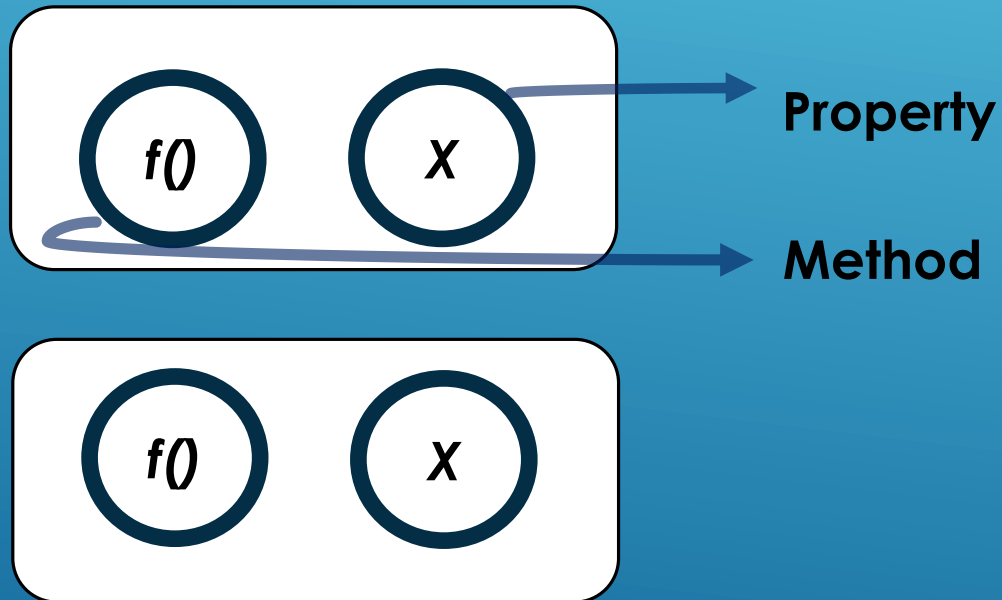
- If you make change on one function, several other functions breaks.



- There is so much interdependence between all these functions, thus, it becomes problematic

OBJECT ORIENTED PROGRAMING CONCEPT

- ▶ In object oriented programming we combine a group related variables and functions into a unit, that unit is called an object.
- ▶ We refer to variable as property
- ▶ and the function as method



OBJECT ORIENTED PROGRAMING CONCEPT

- ▶ As an **Example** we can think in a Car, a car is an object with
- ▶ properties such as (make, model, color)
- ▶ Methods like (start, stop and move)

Grouping related variables and functions that operates on them into object is called **encapsulation**

Benefits of encapsulation

- 1- simpler interface (no too many parameters).
- 2- reuse objects in different parts of a program or in different programs

CAR

Make
Model
Color

Start()
Stop()
Move()

OBJECT ORIENTED PROGRAMING CONCEPT

Abstraction

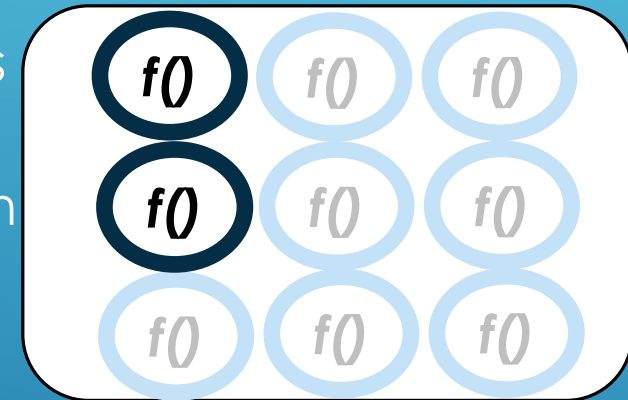
Let us look at DVD player as an object this DVD player has a complex logic board inside and a few buttons on the outside that you interact with, you simply press the play button and you don't care what happened on the inside, that complexity is hidden from you. This is **Abstraction**.



We can use the same technique in our objects, so we can hide some of the properties and methods from the outside and this gives us a couple of **benefits**

1-make the interface of the objects simpler using, understanding an object with few properties and methods is easier than an objects with several properties and methods

2- help us to reduce the impact of change. If we made any changes to the object in the future (changing parameters, methods or deletion) will not impact the rest of the application that use that objects



OBJECT ORIENTED PROGRAMING CONCEPT

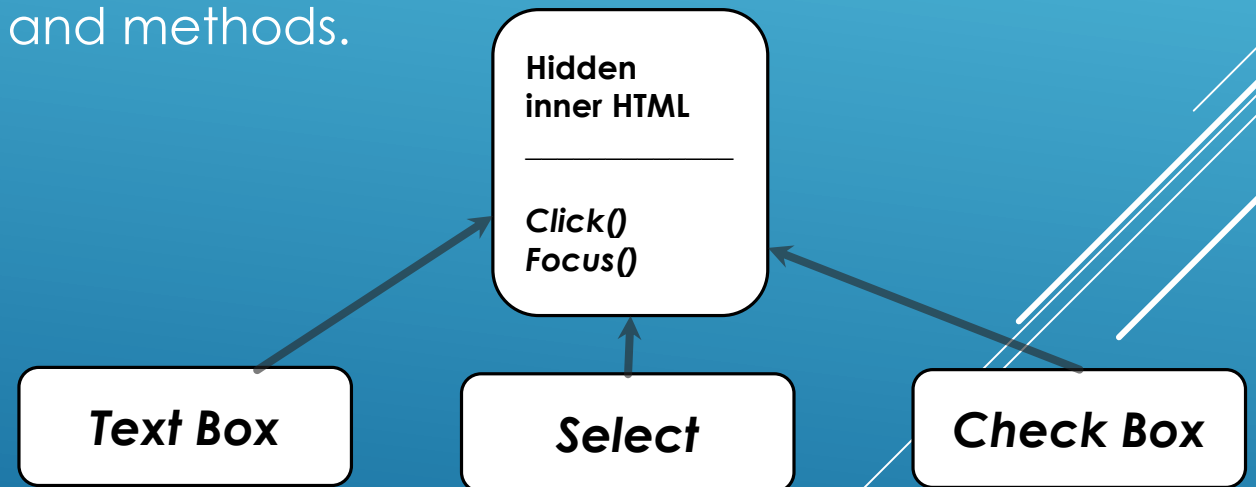
Inheritance

Is a mechanism that allows you to eliminate redundant code .

as an **example** think of HTML element,

All the elements has a common **properties** like(hidden and inner HTML) and **methods** like (click and focus)

Instead of redefining all these properties and methods for every type of HTML elements we can define them once in a generic object called **HTML element** and other objects inherit these properties and methods.



OBJECT ORIENTED PROGRAMING CONCEPT

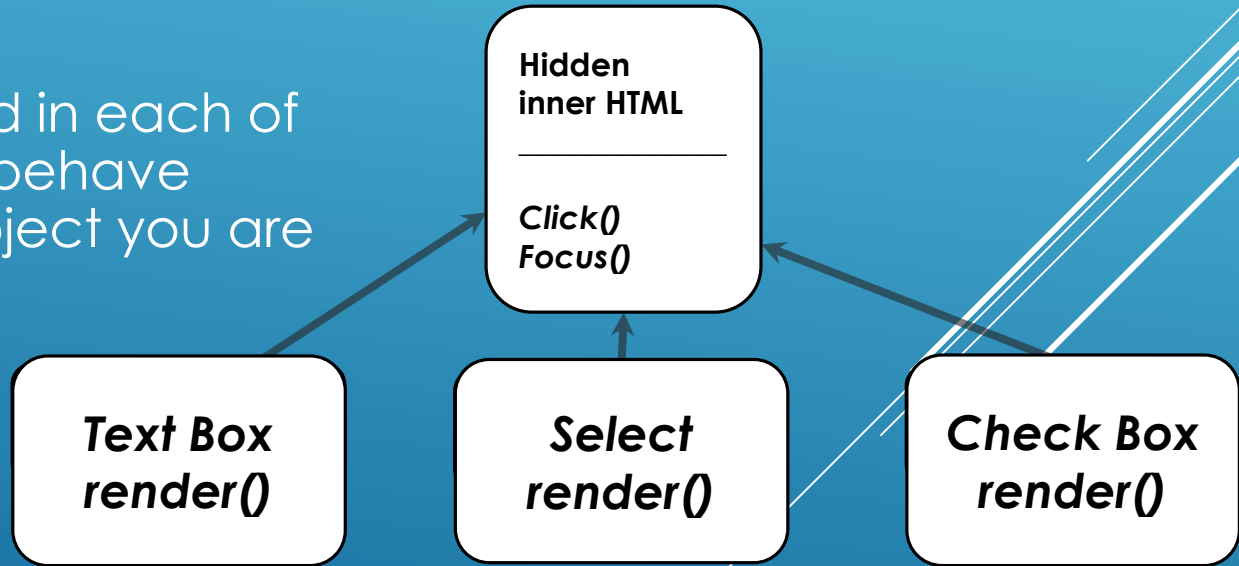
Polymorphism

Poly means Many, and **morph** means Shape

In OOP **Polymorphism** is a technique provides a method of creating multiple forms of a function by using a single function name.

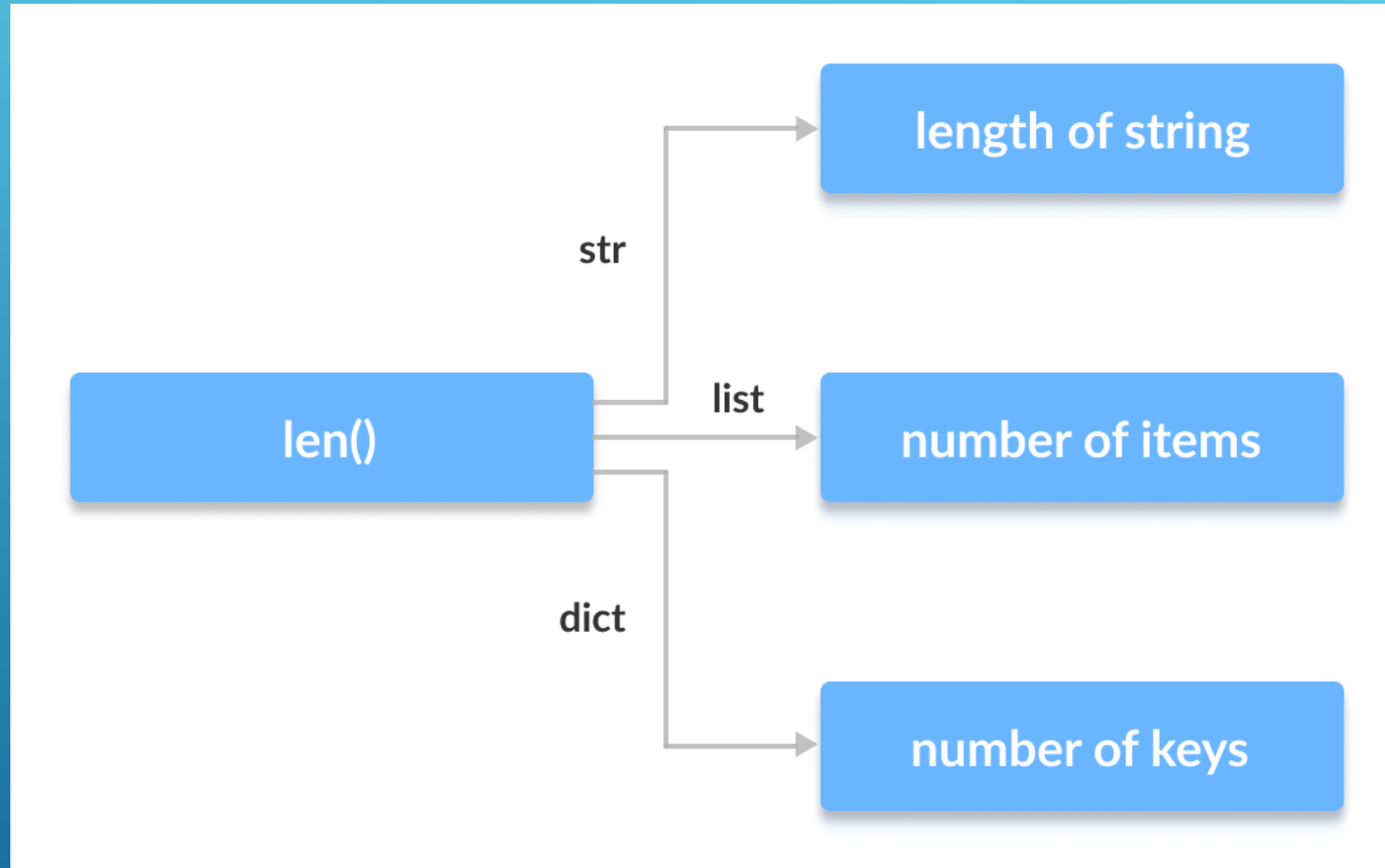
Back to the HTML example all these objects have the ability to be rendered on a page, but, the way each element is rendered is different from others.

In OOP we can implement a render method in each of these objects. Thus, the render method will behave differently depending on the type of the object you are referencing.




OBJECT ORIENTED PROGRAMING CONCEPT

Polymorphism



BENEFITS OF OOP

Encapsulations:	Reduce complexity + Increase reusability
Abstraction:	Reduce complexity + Isolate impact of changes
Inheritance:	Eliminate redundant code
Polymorphism:	makes programming more intuitive and easier

A series of three parallel white diagonal lines in the bottom right corner of the slide, extending from the middle of the right edge towards the bottom left.