py4kids (https://github.com/wgong/py4kids)

Object and Class

In this lesson, we learn object-oriented programming in python. (Read Chapter 8 of textbook - Python for Kids)

Key concepts

- Object - how computer models / simulates a Thing (or everything) in the real world
- Class - how to generalize (abstract, or template) objects

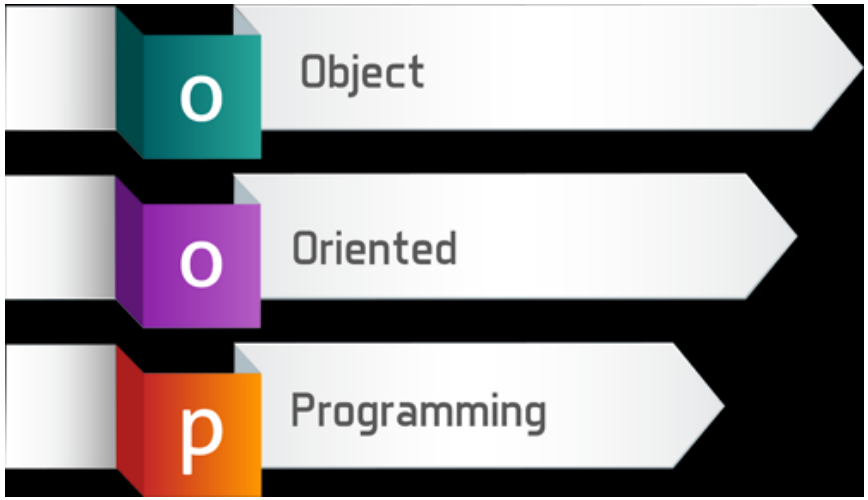
Object has Property and Method. In general, Property describes Form, Method describes Function.

Object to Class is like Cookie to Cookie Cutter. Class is Object Creator.

In [1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

Out[1]:

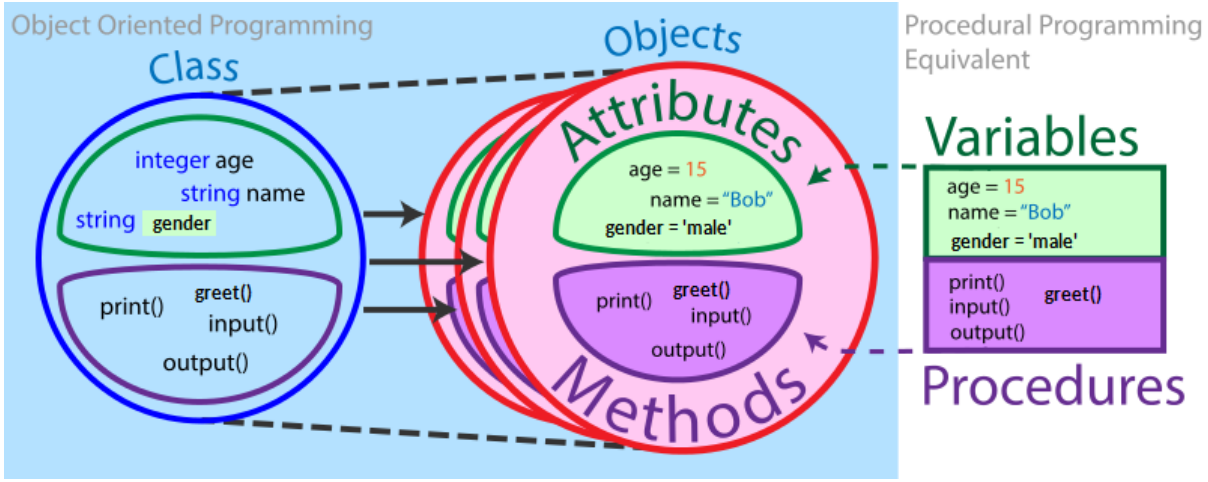
Why OOP?



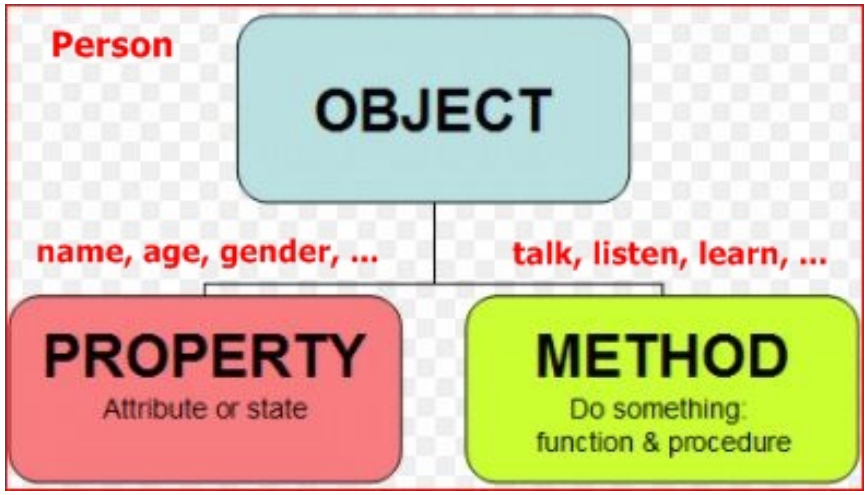
- Mankind is creating new digital virtual world, which imitates how real world works.
- Modular and Component-based design is one of the best engineering principles and practices to solve complex problems.
- OOP codes have better efficency, robustness, ease of use, maintainability and reusability.
- Introduction to Computer Science Object Oriented Programming: Advantages of OOP (https://www.cs.drexel.edu/~introcs/Fa15/notes/06.1_OOP/Advantages.html?CurrentSlide=3)
- 4 Advantages of Object-Oriented Programming (https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming)

Object-oriented programming (OOP)

Procedural Programming vs Object-Oriented Programming

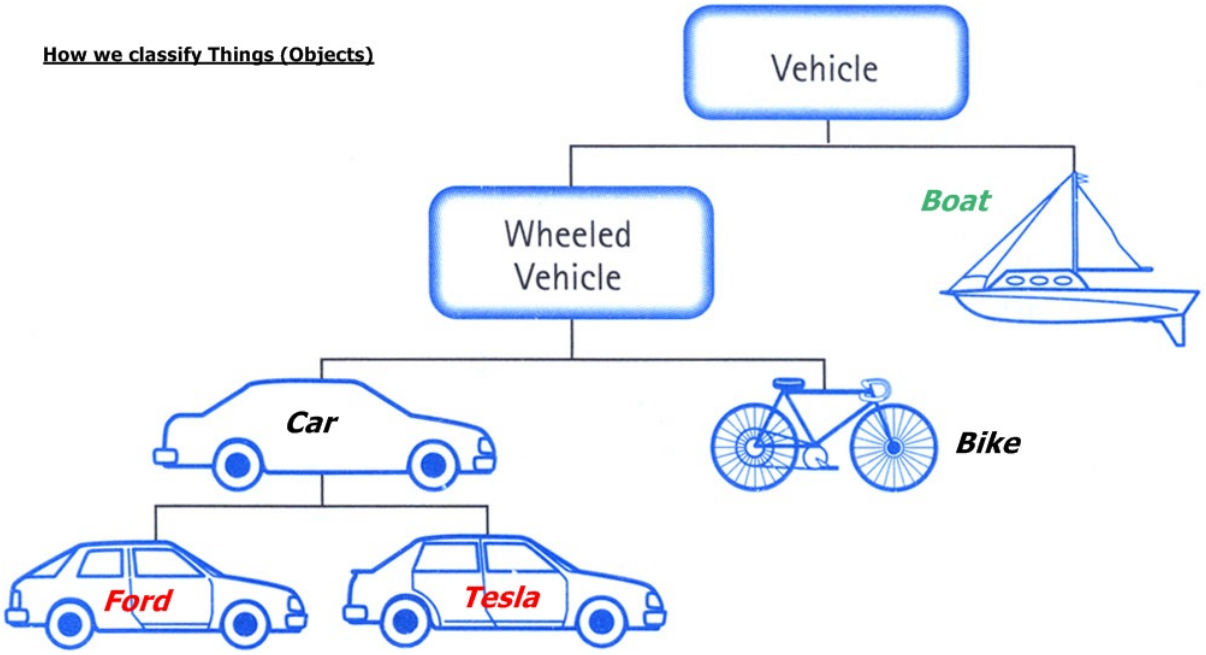


Object has two key components: properties and methods



Object has hierarchy - parent and child relationship : child object inherits from parent object.

Base Class is parent, SubClass is child.



In [2]:
```
# base class: vehicle
class Vehicle:
    pass    # fill in details later
    # vehicle is a machine that moves
```

In [3]:
```
# WheeledVehicle and Boat class inherit parent class=Vehicle
class WheeledVehicle(Vehicle):
    pass    # fill in details later
    # vehicle that runs on land

class Boat(Vehicle):
    pass    # fill in details later
    # vehicle that runs in water
```

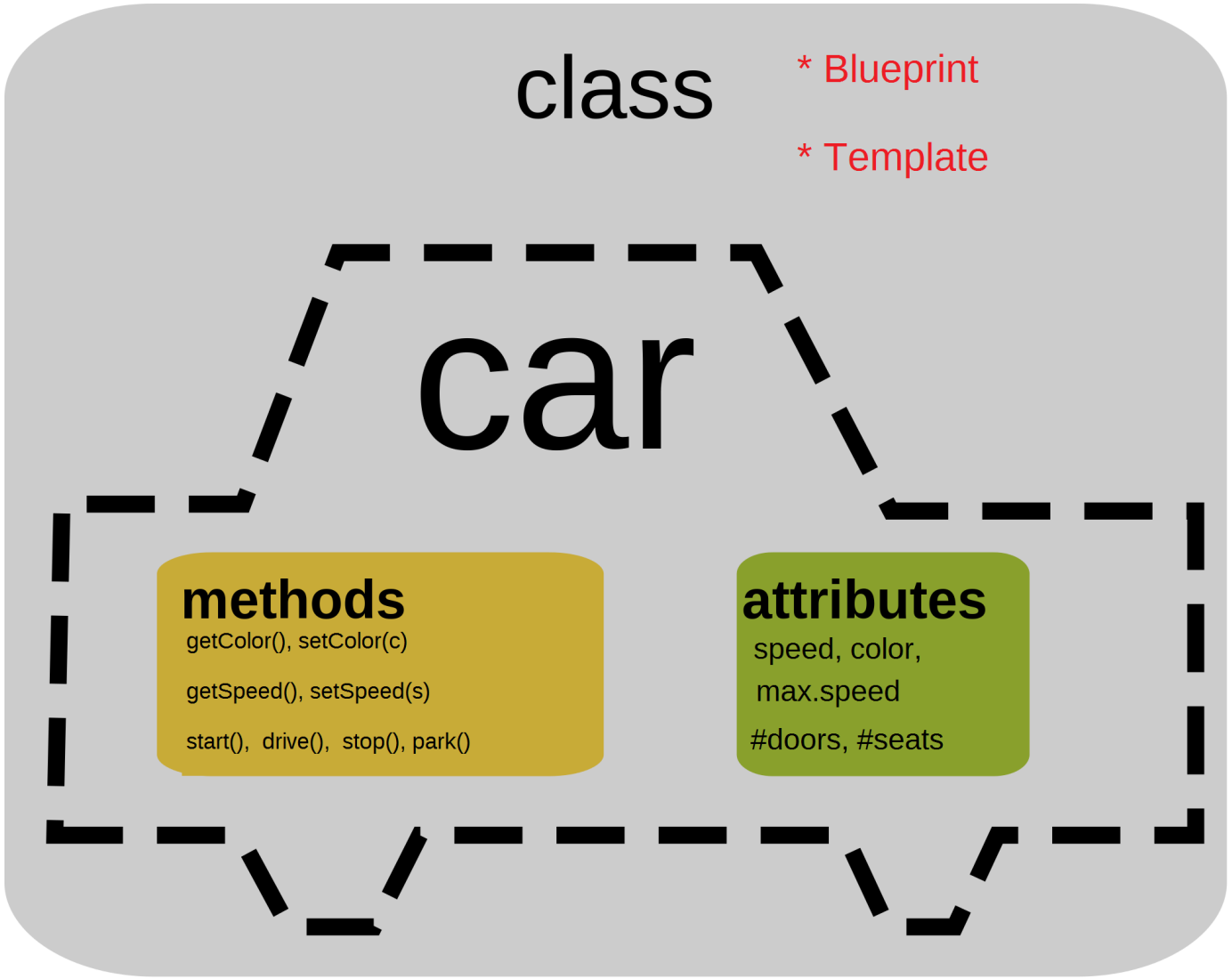
In [4]:
```
class Car(WheeledVehicle):
    pass
    # passenger vehicle wiht 4 wheels

class Bike(WheeledVehicle):
    pass
    # 2 wheels
```

In [5]:
```
class Ford(Car):
    pass
    # car made by Ford, burns gas

class Tesla(Car):
    pass
    # electrical car by Musk, uses electricity
```

Car Class


class
* Blueprint
* Template
car
methods
getColor(), setColor(c)
getSpeed(), setSpeed(s)
start(), drive(), stop(), park()
attributes
speed, color,
max.speed
#doors, #seats

In [6]:
```
class Car:
    # constructor: define properties and how object is created initially
    def __init__(self, number_of_wheels, number_of_doors, color, seating_capacity, maximum_speed):
        self.number_of_wheels = number_of_wheels
        self.number_of_doors = number_of_doors
        self.color = color
        self.seating_capacity = seating_capacity
        self.maximum_speed = maximum_speed
        self.speed = 0
        self.alarm = False

    # getters and setters
    def getColor(self):
        return self.color

    def setColor(self, color):
        self.color = color

    def getSpeed(self):
        return self.speed

    def setSpeed(self, speed):
        self.speed = speed

    # private method
    def _accelerate(self, target_speed, minutes_to_accelerate):
        delta_per_min = (target_speed - self.speed) / minutes_to_accelerate
        for tm in range(int(minutes_to_accelerate)):
            self.speed = self.speed + delta_per_min

    def _steer(self):
        pass

    def _brake(self):
        pass

    # utility
    def show_properties(self):
        props = (self.number_of_wheels, self.number_of_doors, self.color, self.seating_capacity, self.maximum_spe
        print("""This Car object's properties: \n\tnumber_of_wheels=%s, \n\tnumber_of_doors=%s, \n\tcolor=%s, \n\

    # public methods
```

```
    def start(self, target_speed, minutes_to_accelerate=2):
        self._accelerate(target_speed, minutes_to_accelerate)
        pass

    def drive(self):
        self._steer()
        if self.alarm:
            self._brake()

    def stop(self):
        self._accelerate(0, 2)
        # turn off engine
```

what is the purpose of self? (https://stackoverflow.com/questions/2709821/what-is-the-purpose-of-self)

The official explanation by Python Tsar - Guido van Rossum (http://neopythonic.blogspot.com/2008/10/why-explicit-self-has-to-stay.html)

Let's say you have a class ClassA which contains a method methodA defined as:

```
    def methodA(self, arg1, arg2):
        # do something
```

and ObjectA is an instance of this class.

Now when ObjectA.methodA(arg1, arg2) is called, python internally converts it for you as:

ClassA.methodA(ObjectA, arg1, arg2) The self variable refers to the object itself.

Counter example:

```
class A:
    foo = []
a, b = A(), A()
a.foo.append(5)
b.foo
ans: [5]

class A:
    def __init__(self):
        self.foo = []
a, b = A(), A()
a.foo.append(5)
b.foo
ans: []
```

Car Objects

From one Car Class, you create as many Car objects as you wish.

In [7]:
```
# let us create a car in computer
# __init__(self, number_of_wheels, number_of_doors, color, seating_capacity, maximum_speed)

my_first_car = Car(4, 2, 'red', 4, 120)
```

In [8]:
```
# Class/Object is custom built type

type(my_first_car)
```

Out[8]:
```
__main__.Car
```

In [9]:
```
my_first_car.__class__
```

Out[9]:
```
__main__.Car
```

In [10]:
```
# query class/object information
dir(my_first_car)
```

Out[10]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_accelerate',
 '_brake',
 '_steer',
 'alarm',
 'color',
 'drive',
 'getColor',
 'getSpeed',
 'maximum_speed',
 'number_of_doors',
 'number_of_wheels',
 'seating_capacity',
 'setColor',
 'setSpeed',
 'show_properties',
```

```
 'speed',
 'start',
 'stop']
```

In [11]:
```
my_first_car.show_properties()
```

```
This Car object's properties:
        number_of_wheels=4,
        number_of_doors=2,
        color=red,
        seating_capacity=4,
        maximum_speed=120,
        speed=0,
        Alarm?=False
```

In [12]:
```
my_first_car.getColor()
```

Out[12]:
```
'red'
```

In [13]:
```
my_first_car.color
```

Out[13]:
```
'red'
```

In [14]:
```
my_first_car.setColor('White')
```

In [15]:
```
my_first_car.color
```

Out[15]:
```
'White'
```

In [16]:
```
my_first_car.getSpeed()
```

Out[16]:
```
0
```

In [17]:
```
my_first_car.start(15,2)
```

In [18]:
```
my_first_car.getSpeed()
```

Out[18]:
```
15.0
```

In [19]:
```
my_first_car.drive()
```

In [20]:
```
my_first_car.getSpeed()
```

Out[20]:
```
15.0
```

In [21]:
```
my_first_car.stop()
```

In [22]:
```
my_first_car.getSpeed()
```

Out[22]:
```
0.0
```

Inheritance

In [23]:
```
class FordEscape(Car):
    def __init__(self, number_of_wheels, number_of_doors, color, seating_capacity, maximum_speed, style):
        Car.__init__(self, number_of_wheels, number_of_doors, color, seating_capacity, maximum_speed)
        self.manufacturer = 'Ford'
        self.style = style

    # add subclass properties
    def show_properties(self):
        Car.show_properties(self)
        print("""\tmanufacturer=%s, \n\tstyle=%s""" % (self.manufacturer, self.style))
```

Since FordEscape Car class is built from base Car class, it inherits Car class's properties and methods (for free)

In [24]:
```
my_2nd_car = FordEscape(4, 4, 'Blue', 6, 140, 'SUV')
```

In [25]:
```
type(my_2nd_car)
```

Out[25]:
```
__main__.FordEscape
```

In [26]:
```
dir(my_2nd_car)
```

Out[26]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_accelerate',
 '_brake',
 '_steer',
 'alarm',
 'color',
 'drive',
 'getColor',
 'getSpeed',
 'manufacturer',
 'maximum_speed',
 'number_of_doors',
 'number_of_wheels',
 'seating_capacity',
 'setColor',
 'setSpeed',
 'show_properties',
```

```
 'speed',
 'start',
 'stop',
 'style']
```

In [27]:
```
my_2nd_car.show_properties()
```

```
This Car object's properties:
        number_of_wheels=4,
        number_of_doors=4,
        color=Blue,
        seating_capacity=6,
        maximum_speed=140,
        speed=0,
        Alarm?=False
        manufacturer=Ford,
        style=SUV
```

In [28]:
```
my_2nd_car.getColor()
```

Out[28]:
```
'Blue'
```

In [29]:
```
my_2nd_car.style, my_2nd_car.manufacturer
```

Out[29]:
```
('SUV', 'Ford')
```

In [30]:
```
my_2nd_car.start(30,2)
my_2nd_car.getSpeed()
```

Out[30]:
```
30.0
```

In [31]:
```
my_2nd_car.drive()
my_2nd_car.getSpeed()
```

Out[31]:
```
30.0
```

In [32]:
```
my_2nd_car.stop()
my_2nd_car.getSpeed()
```

Out[32]:
```
0.0
```

Learn by example - the Open Source way

In the open source world, it is not only legal to copy open-sourced codes and programs as long as one credits the original source, it is also encouraged to develop and innovate.

Check out The Free Software Foundation (FSF) (http://www.fsf.org/), and its founding father Richard Stallman (https://www.wikiwand.com/en/Richard_Stallman), and GNU Project (https://www.gnu.org/)

Pygame (https://www.pygame.org/) sample program - Fist punches Chimp

It has the basic ingredients for a simple game with sound, image, object collision detection.

More examples can be found at C:\Anaconda3\Lib\site-packages\pygame\examples (for window installation)

To play chimp game, type

```
python chimp.py
```

Here are the Fist and Chimp classes/objects

```
#classes for our game objects
class Fist(pygame.sprite.Sprite):
    """moves a clenched fist on the screen, following the mouse"""
    def __init__(self):
        pygame.sprite.Sprite.__init__(self) #call Sprite initializer
        self.image, self.rect = load_image('fist.bmp', -1)
```

```
            self.punching = 0

        def update(self):
            "move the fist based on the mouse position"
            pos = pygame.mouse.get_pos()
            self.rect.midtop = pos
            if self.punching:
                self.rect.move_ip(5, 10)

        def punch(self, target):
            "returns true if the fist collides with the target"
            if not self.punching:
                self.punching = 1
                hitbox = self.rect.inflate(-5, -5)
                return hitbox.colliderect(target.rect)

        def unpunch(self):
            "called to pull the fist back"
            self.punching = 0


    class Chimp(pygame.sprite.Sprite):
        """moves a monkey critter across the screen. it can spin the
           monkey when it is punched."""
        def __init__(self):
            pygame.sprite.Sprite.__init__(self) #call Sprite intializer
            self.image, self.rect = load_image('chimp.bmp', -1)
            screen = pygame.display.get_surface()
            self.area = screen.get_rect()
            self.rect.topleft = 10, 10
            self.move = 9
            self.dizzy = 0

        def update(self):
            "walk or spin, depending on the monkeys state"
            if self.dizzy:
                self._spin()
```

```
            else:
                self._walk()

        def _walk(self):
            "move the monkey across the screen, and turn at the ends"
            newpos = self.rect.move((self.move, 0))
            if self.rect.left < self.area.left or \
                self.rect.right > self.area.right:
                self.move = -self.move
                newpos = self.rect.move((self.move, 0))
                self.image = pygame.transform.flip(self.image, 1, 0)
            self.rect = newpos

        def _spin(self):
            "spin the monkey image"
            center = self.rect.center
            self.dizzy = self.dizzy + 12
            if self.dizzy >= 360:
                self.dizzy = 0
                self.image = self.original
            else:
                rotate = pygame.transform.rotate
                self.image = rotate(self.original, self.dizzy)
            self.rect = self.image.get_rect(center=center)

        def punched(self):
            "this will cause the monkey to start spinning"
            if not self.dizzy:
                self.dizzy = 1
                self.original = self.image
```

In [ ]:

In [ ]: