**Methods**

Don’t repeat yourself, you want to be lazy.

They can do as much or little as you need them to.

*Def hello*

*Puts “hello universe!”*

*Def really\_hard\_math*

*Answer = (-5 + math.sqrt (5\*\*2 – 4 \* (8\*15)))/(2 \* 8)*

*Puts answer*

*End*

*Object.add\_it\_up* applies the method to the object

To “call” a method, just write it.

**Return**

Similar to puts, but assigning a value to the method without running the code

Like a “fixed variable”

So if we:

*Def add\_it\_up*

*Sum = 3 + 5*

*Return sum*

*End*

After this, *add\_it\_up* is treated as 8 so you can include it… for example:

*Puts 8 + add\_it\_*up is now 16.

*X = add\_it\_up*

Assigns the value returned to a variable

You also can just put nothing and it will return it automatically.

**Arguments & Parameters**

Can put passed values to play with.

*Def add\_it\_up (num1, num2)*

*Sum = num1 + num2*

*End*

*Puts add\_it\_up(4, 5)*

*Def add\_it\_up(num1, num2)*

*Sum = num1 + num2*

*End*

*Time = 4*

*Space = 5*

*Puts add\_it\_up(time, space)*

**OOP**

Atrributes (tall, brown hair, smart)

Methods (walks, plays drums, etc.)

Define an object using “class” and “end” to tell Ruby when the class is over.

*Class object*

*End*

*Class objet*

*Def method1*

*End*

*Def method2*

*End*

*end*

**Instance & Initialization**

Instance variables are accessible anywhere in the whole class

*class Object*

*def initiatlize (attr1, attr2)*

*@attr1 = attr1*

*@attr2 = attr2*

*end*

*end*

#Always start class names with a capital letter

class GamePlay < Casino

#GamePlay inherits characteristics of the Casino class

class Person

def initialize(name, age)

@name = name

@age = age

end

end

my.profile = person.net(“aaron”,34)

Initialize is on its own and just sets things up, still need methods for each thing.

**Getter and Setter**

attr\_reader: instance\_variable 🡪 This is your getter

attr\_writer :instance\_variable 🡪 This is your setter

attr\_accessor :instance\_ariable 🡪 This is both

atrr\_reader: age … what it really means is:

*def age=(value)*

*@age = value*

*end*

reader is just accessible info that the user can’t change

accessor is both readable and writeable

can do:

*attr\_accessor :name, :age, :kind*

to do multiples

Accessor is typically what you use

Attr allows other methods to access instance variables in other methods.

**Attr with default values**

*Class Pet*

*Attr\_reader :kind*

*Attr\_writer :age*

*Attr\_accessor :name*

*Def initialize(name =’Marley’, breed =’Mix’, age, sound)*

*@name = name*

*@age = age*

*@sound = sound*

*@kind = breed*

*end*

*def sound*

*puts @sound*

*end*

*end*

*dog = Put.new(5, ‘woof’)*

*dog.sound*

*puts dog.kind*

*dog.age = 37*

*puts dog.name*

*dog.name = ‘Snoopy’*

*puts dog.name*

**Class Vars**

Class Inventory

@@owner = “Walid”

attr\_accessor :name, :brand

This doesn’t change… not used very often.

**Named Parameters**

*sound:*

*name:”Lukas”* 🡪 key/value hash

*hash {name: “Lukas”, age: 30}*

Can also add that and pass it as a hash… not really any point.