

# Python Tutorial Series 2019

## Basics 2: Code Flow

Ph.D. Bruno C. Quint

bquint@gemini.edu

Data Process Developer

SUSD - GEMINI South, La Serena, Chile



# Table of Contents

- What is Python?
- What will you need?
- Python as a terminal
- Python as a script
- Types of variables
- Code Flow
- Functions
- Classes
  - Methods
  - Attributes
- Sub-Classes



# Basic Script Structure

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
    This is a docstring with information about the file.
"""

import external_package
from external_package import external_module

def my_function(...):
    ...

class MyClass(...):
    ...

... (some code here) ...
```



# Loops and control

## if/elif/else

```
>>> fruit = 'banana'  
>>>  
>>> if fruit is banana:  
...     eat_it()  
...  
>>> elif fruit is orange:  
...     make_a_juice()  
...  
>>> else:  
...     leave_it()
```



Easy

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

## if/elif/else

```
>>> fruit = 'banana'  
>>>  
>>> if fruit is 'apple':  
...     ....eat_it()  
  
...  
>>> elif fruit is 'orange':  
...     ....make_a_juice()  
  
...  
>>> else:  
...     ....leave_it()
```

Python relies  
on indentation, so  
**DON'T  
MESS UP!**

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

## if/elif/else

```
>>> fruit = 'banana'  
>>>  
>>> if fruit is 'apple':  
...     ....eat_it()  
...  
>>> elif fruit is 'orange':  
...     ....make_a_juice()  
...  
>>> else:  
...     →leave_it()
```

PEP-8 **HIGHLY** recommends you to use 4 spaces. And **NEVER** mix spaces and tabs. For more informations, read the [PEP-8](#).

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

## if/elif/else

```
>>> fruit = 'banana'  
>>>  
>>> if fruit is 'apple':  
...     ....eat_it()  
  
...  
>>> elif fruit is 'orange':  
...     ....make_a_juice()  
  
...  
>>> else:  
...     ....leave_it()
```

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

List, tuples, arrays,  
matrixes, dictionaries **for**

```
>>> my_list = [ 'a' , 'b' , 'c' ]  
>>>  
>>> for my_item in my_list:  
...     ...     print(my_item)  
...  
a  
b  
c
```

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

List, tuples, arrays,  
matrixes, dictionaries **for**

```
>>> my_list = ['a', 'b', 'c']
>>>
>>> for index in range(len(my_list)):
...     print(index, my_item[index])
...
0 a
1 b
2 c
```

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

List, tuples, arrays,  
matrixes, dictionaries **for**

```
>>> my_list = [ 'a' , 'b' , 'c' ]  
>>>  
>>> for index in range(len(my_list)):  
...     print(index, my_item[index])  
  
0 a  
1 b  
2 c
```

**len(x)** : return the number of elements of x.

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

List, tuples, arrays,  
matrixes, dictionaries **for**

```
>>> my_list = [ 'a' , 'b' , 'c' ]  
>>>  
>>> for index in range(3):  
...     print(index, my_item[index])  
  
0 a  
1 b  
2 c
```

**len(x)** : return the number of elements of x.

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

List, tuples, arrays,  
matrixes, dictionaries **for**

```
>>> my_list = [ 'a' , 'b' , 'c' ]  
>>>  
>>> for index in range(3):  
...     print(index, my_item[index])  
  
0 a  
1 b  
2 c
```

**range(n)** : return a list with n integers starting at 0.

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

List, tuples, arrays,  
matrixes, dictionaries **for**

```
>>> my_list = [ 'a' , 'b' , 'c' ]  
>>>  
>>> for index in [0, 1, 2]:  
...     print(index, my_item[index])  
  
0 a  
1 b  
2 c
```

**range(n)**: return a list with n integers starting at 0.

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

## While

test\_while.py

```
nInterested = 5

while nInterested > 0:
    ...print(" Keep going!")
    ...nInterested = nInterested - 1

print(" Time to go home!")
```

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

## While

test\_while.py

```
nInterested = 5
```

```
while nInterested > 0:  
    ...print(" Keep going!")  
    ...nInterested = nInterested - 1  
  
print(" Time to go home!")
```

```
$ python test_while.py  
Keep going!  
Keep going!  
Keep going!  
Keep going!  
Keep going!  
Time to go home!
```

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

test\_break.py

```
nInterested = 5

while True:
    ... print(" Keep going!")
    ... nInterested = nInterested - 1
    ... if nInterested < 0:
        ...     break

print(" Time to go home!")
```

# break

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Loops and control

## continue

```
>>> my_list = ['a', 'b', 'c']
>>>
>>> for index in [0, 1, 2]:
...     if index == 1:
...         continue
...     print(index, my_item[index])
...
0 a
2 c
```

More at <https://docs.python.org/3/tutorial/controlflow.html>



# Functions

Define a new function

```
>>> def my_function(x, y):
...     .... """I am a docstring!"""
...     .... k = 2 * x - y # I am a comment
...     .... return k
```



# Functions

Define a new function

```
>>> def my_function(x, y):  
...     .... """I am a docstring!"""  
...     .... k = 2 * x - y # I am a comment  
...     .... return k  
...  
>>> my_function(2, 4)  
0
```

**x** and **y** are **positional** parameters.



# Functions

Define a new function

```
>>> def my_function(x, y):  
...     .... """I am a docstring!"""  
...     .... k = 2 * x - y # I am a comment  
...     .... return k  
...  
>>> my_function(y=4, x=2)  
0
```

**x** and **y** are now **keyword** parameters.



# Functions

Define a new function

```
>>> def my_function(x, y, z=1, w=0):  
...     .... """I am a docstring!"""  
...     .... k = 2 * x - y / z + w  
...     .... return k  
...  
>>> my_function(y=4, x=2)  
0
```

\*BA DUM TSSS\*



**z** and **w** are now **default** parameters.

**Default** parameters have **default** values.



# Functions

Define a new function

```
>>> def my_function(x, y, z=1, w=0):  
...     .... """I am a docstring!"""  
...     .... k = 2 * x - y / z + w  
...     .... return k  
...  
>>> my_function(y=4, x=2, w=5)  
5
```

**z** and **w** are now **default** parameters.

**Default** parameters have **default** values.



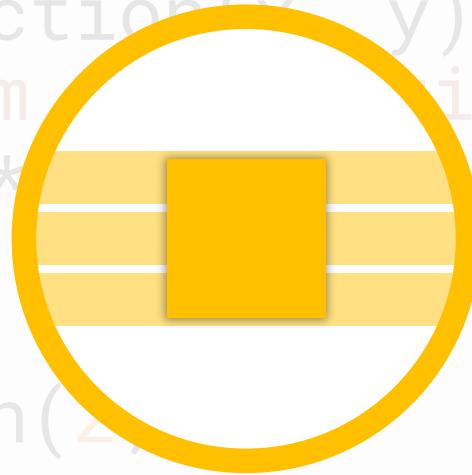
# Functions

```
>>>  
>>> def my_function(x, y):  
...     ...     """I am a docstring!"""  
...     ...     k = 2 * x - y  
...     ...     return k  
...  
>>> my_function(2, 4)  
0
```



# Functions

```
>>>  
>>> def my_function(x,y):  
...     """I am doing!  
...     k = 2 *  
...     return  
...  
>>> my_function(2,  
0
```

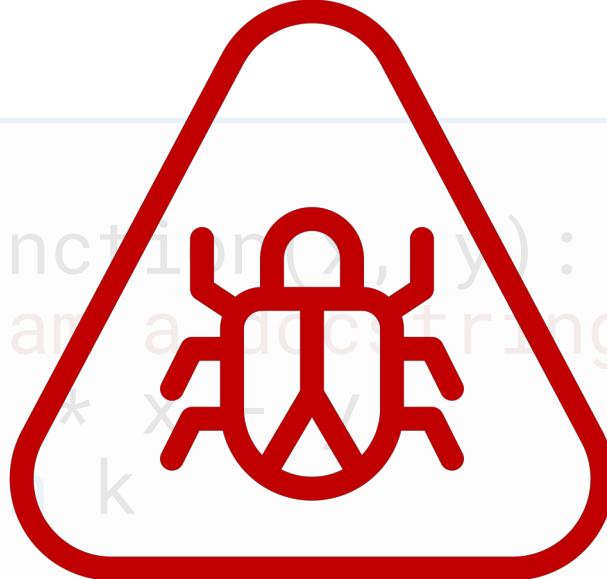


Medium



# Functions

```
>>>  
>>> def my_function(x,y):  
...     """I am a docstring!  
...     k = 2 * x + y  
...     return k  
...  
>>> my_function(2, 4)  
0
```



Bug, Alert



# Functions

Watch out namespaces!

```
>>>  
>>> def my_function(*, y):  
...     """I am a docstring!"""  
...     k = 2 * x - y  
...     return k  
...  
>>> my_function(2, 4)
```



# Functions

Watch out namespaces!

```
>>>  
>>> def my_function(*, y):  
...     .... """I am a docstring!"""  
...     .... k = 2 * x - y  
...     .... return k  
...  
>>> my_function(2, 4)  
NameError: global name 'x' is not defined
```



# Functions

Watch out namespaces!

```
>>> x = 2
>>> def my_function(x, y):
...     """I am a docstring!
...     k = 2 * x - y
...     return k
...
>>> my_function(2, 4)
```



# Functions

Watch out namespaces!

```
>>> x = 2
>>> def my_function(x, y):
...     """I am a docstring!
...     k = 2 * x - y
...     return k
...
>>> my_function(2, 4)
0
```



# Functions

Watch out namespaces!

```
>>>  
>>> def my_function(x, y):  
...     """I am a docstring!"""  
...     k = 2 * x - y  
...     return k  
...  
>>> x = 2  
>>> my_function(2, 4)
```



# Functions

Watch out namespaces!

```
>>>  
>>> def my_function(x, y):  
...     """I am a docstring!"""  
...     k = 2 * x - y  
...     return k  
...  
>>> x = 2  
>>> my_function(2, 4)  
0
```



# Functions

Watch out namespaces!

```
>>>  
>>> def my_function(x, y):  
...     .... """I am a docstring!"""  
...     .... k = 2 * x - y  
...     .... return k  
...  
>>> x = 2  
>>> my_function(3, 4)  
2
```



# Classes

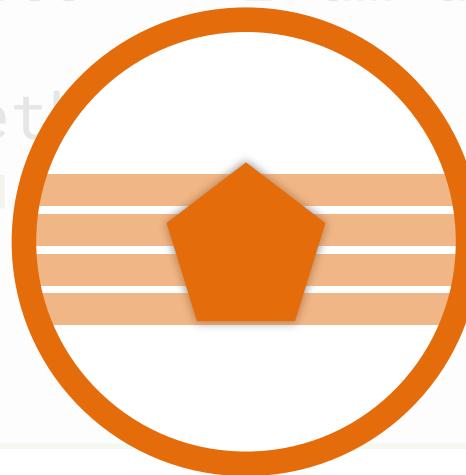
```
>>> class MyClass:  
...     ...  
...     my_attribute = "I am an alien"  
...     ...  
...     def my_method(self):  
...         ...     return "Calling my_method"  
...     ...  
>>>
```

More about [Classes on W3Schools](#)



# Classes

```
>>> class MyClass:  
...     ...  
...         my_attribute = "I am an alien"  
...     ...  
...         def my_method:  
...             return "My method"  
...     ...  
...>>>
```



# Difficult

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class MyClass:  
...     ...  
...     my_attribute = "I am an alien"  
...     ...  
...     def my_method(self):  
...         ...     return "Calling my_method"  
...     ...  
>>> my_object = MyClass()
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class MyClass:  
...     ...  
...     my_attribute = "I am an alien"  
...     ...  
...     def my_method(self):  
...         ...     return "Calling my_method"  
...     ...  
>>> my_object = MyClass()  
>>> print(my_object.my_attribute)
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class MyClass:  
...     ...  
...     my_attribute = "I am an alien"  
...     ...  
...     def my_method(self):  
...         ...     return "Calling my_method"  
...     ...  
>>> my_object = MyClass()  
>>> print(my_object.my_attribute)  
I am an alien
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class MyClass:  
...     ...  
...     my_attribute = "I am an alien"  
...     ...  
...     def my_method(self):  
...         ...     return "Calling my_method"  
...     ...  
>>> my_object = MyClass()  
>>> print(my_object.my_method)  
<bound method MyClass.my_method of <__main__.MyClass object at 0x1087d42b0>>
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class MyClass:  
...     ...  
...     my_attribute = "I am an alien"  
...     ...  
...     def my_method(self):  
...         ...     return "Calling my_method"  
...     ...  
>>> my_object = MyClass()  
>>> print(my_object.my_method())  
Calling method
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         print("I am {}!".format(  
...             self.name))  
...     ...  
>>> stranger = Person(name="Bruno")
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class Person:  
...     ...  
...         def __init__(self, name):  
...             self.name = name  
...     ...  
...         def introduce_yourself(self):  
...             print("I am {}!".format(  
...                 self.name))  
...     ...  
>>> stranger = Person(name="Bruno")
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class Person:  
...     ...  
...         def __init__(self, name):  
...             self.name = name  
...         ...  
...             def introduce_yourself(self):  
...                 print("I am {}!".format(  
...                     self.name))  
...     ...  
>>> stranger = Person(name="Bruno")  
>>> type(stranger)  
<class '__main__.Person'>
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         print("I am {}!".format(  
...             self.name))  
...     ...  
>>> stranger = Person(name="Bruno")  
>>> print(stranger.name)  
Bruno
```

More about [Classes on W3Schools](#)



# Classes

Methods and  
atributes

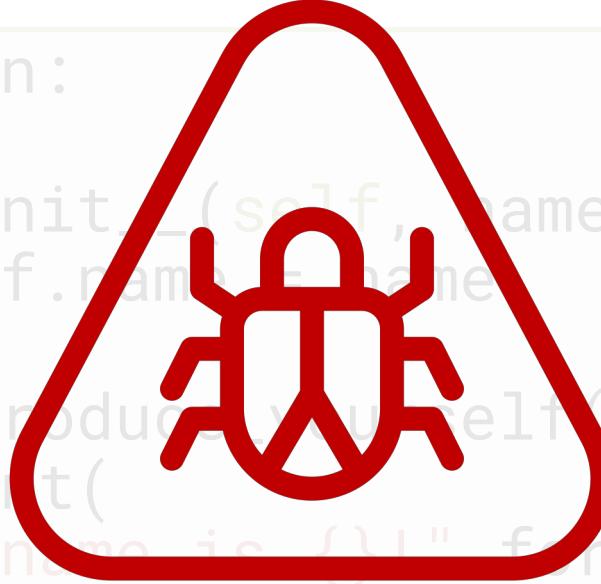
```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         print("I am {}!".format(  
...             self.name))  
...     ...  
>>> stranger = Person(name="Bruno")  
>>> print(stranger.introduce_yourself())
```

More about [Classes on W3Schools](#)



# Classes

```
>>> class Person:  
...     def __init__(self, name):  
...         self.name = name  
...     def introduce_yourself(self):  
...         print("Hello! My name is {}!".format(self.name))  
...>>> stranger = Person("Bruno")  
>>> print(stranger.introduce_yourself())
```



**Bug Alert**

More about [Classes on W3Schools](#)



# Classes



Bug Alert

```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         print("I am {}!".format(  
...             self.name))  
...     ...  
>>> stranger = Person(name="Bruno")  
>>> print(stranger.introduce_yourself())  
Hello! My name is Bruno!  
None
```

More about [Classes on W3Schools](#)



# Classes



Bug Alert

```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         print("I am {}!".format(  
...             self.name))  
...  
>>> stranger = Person(name="Bruno")  
>>> stranger.introduce_yourself()  
Hello! My name is Bruno!
```

More about [Classes on W3Schools](#)



# Classes



Bug Alert

```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         return "I am {}!".format(  
...             self.name)  
...  
>>> stranger = Person(name="Bruno")  
>>> print(stranger.introduce_yourself())  
Hello! My name is Bruno!
```

More about [Classes on W3Schools](#)



# Classes



Bug Alert

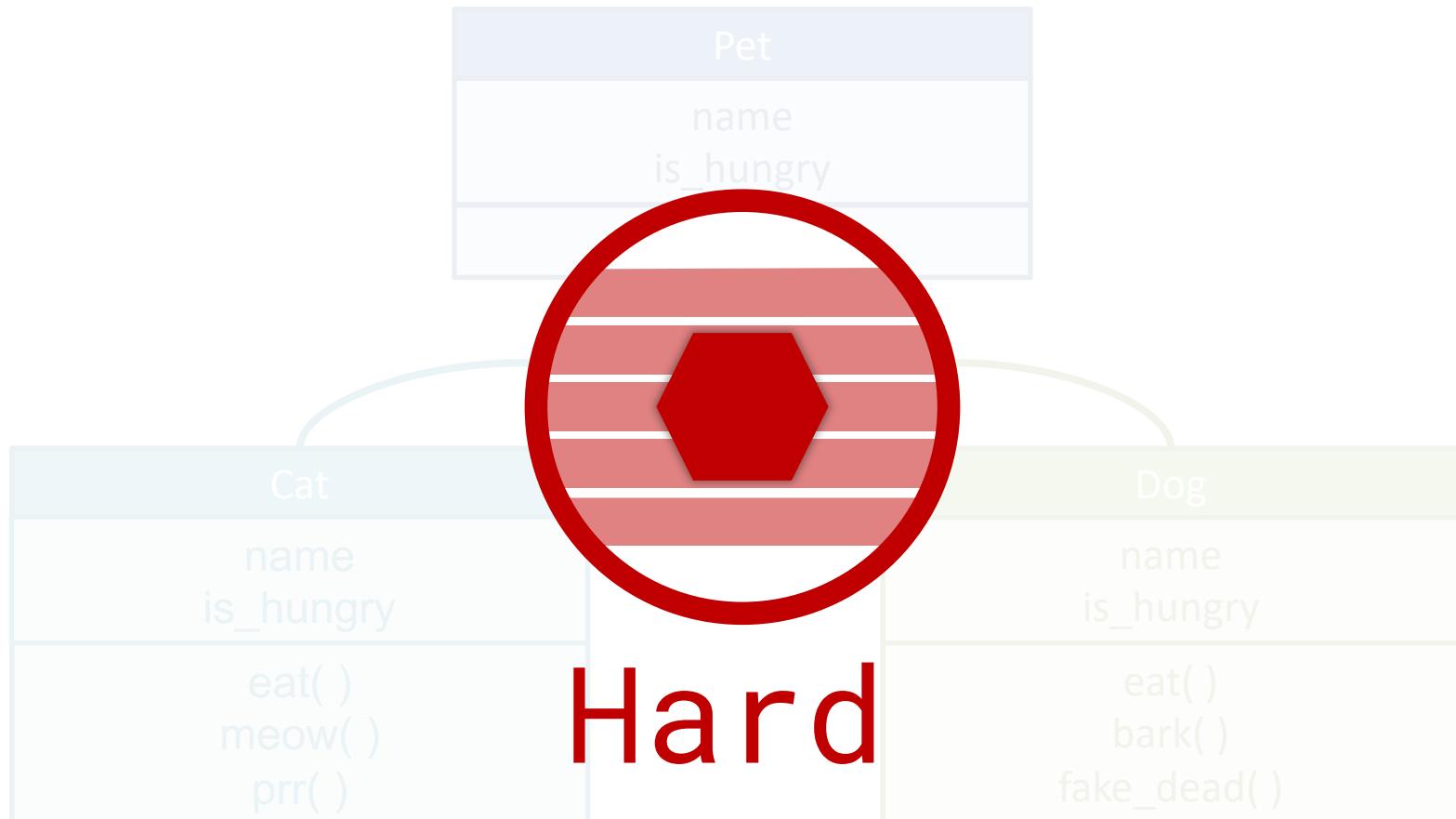
```
>>> class Person:  
...     ...  
...     def __init__(self, name):  
...         self.name = name  
...     ...  
...     def introduce_yourself(self):  
...         return "I am {}!".format(  
...             self.name)  
...  
>>> stranger = Person(name="Bruno")  
>>> print(stranger.introduce_yourself())  
Hello! My name is Bruno!
```

More about [Classes on W3Schools](#)



# Classes

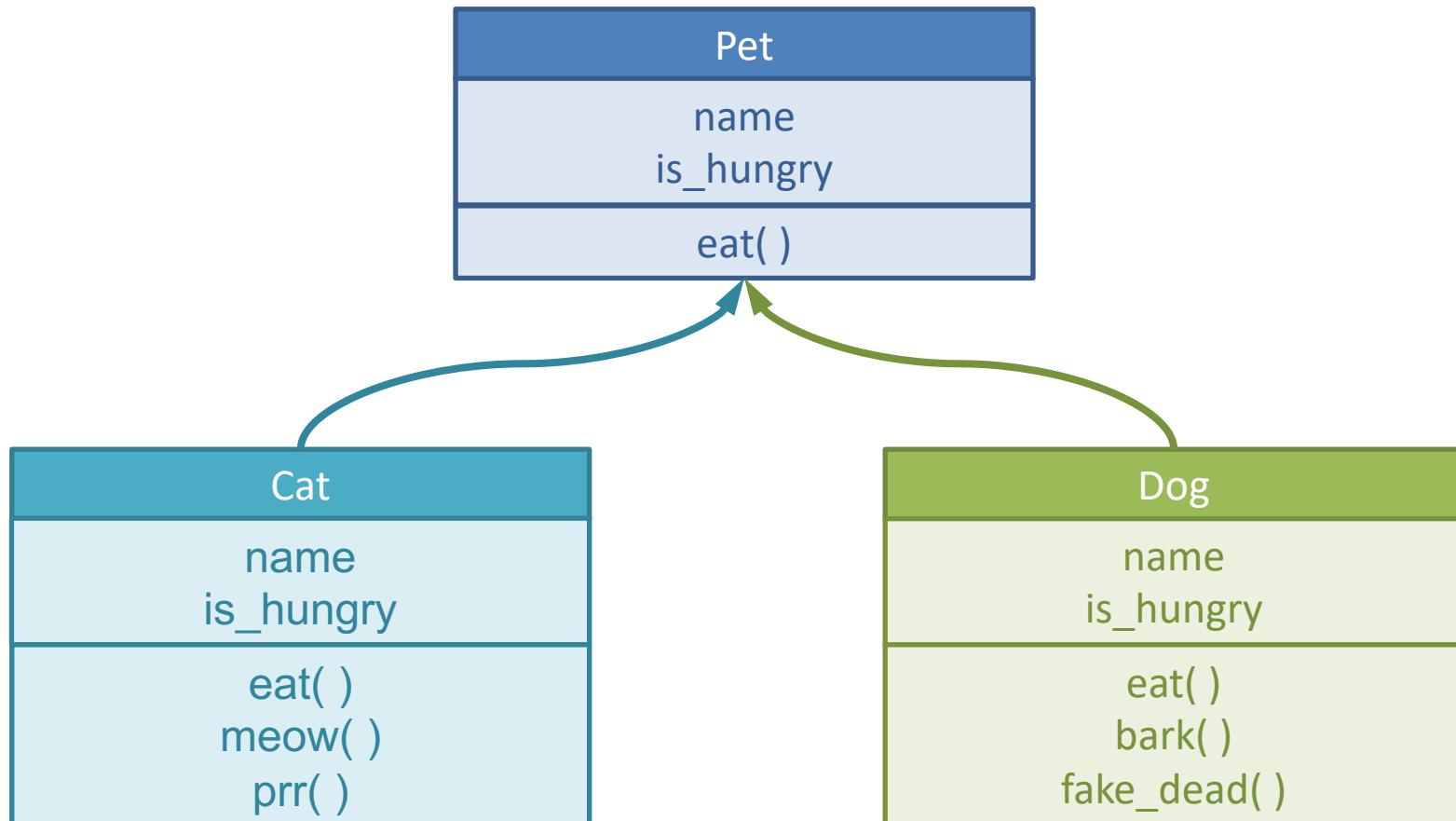
Creating sub-classes  
aka: inheritance





# Classes

Creating sub-classes  
aka: inheritance





# Classes

Creating sub-classes  
aka: inheritance

```
>>> class Pet:  
...     ...  
...     def __init__(self, new_name):  
...         self.name = new_name  
...         self.is_hungry = True  
...     ...  
...     def eat(self, food_name):  
...         ...  
...         print("{} eats {}".format(  
...             self.name, food_name))  
...     ...  
...         self.is_hungry = False  
...     ...
```



# Classes

Creating sub-classes  
aka: inheritance

```
>>> class Cat(Pet):
...     ...
...     def eat(self, food_name):
...         if food_name not in ['Biscuit']:
...             print("{} eats {}".format(
...                 self.name, food_name))
...         self.is_hungry = False
...     ...
```

```
>>> class Dog(Pet):
...     ...
...     def eat(self, food_name):
...         if food_name not in [Milk']:
...             super(Dog, self).eat(food_name)
...     ...
```

More about [Class Inheritance and super\(\)](#)



# Classes

Creating sub-classes  
aka: inheritance

```
>>> Mila = Cat('Mila')
>>> isinstance(Mila, Pet)
True
>>> isinstance(Mila, Cat)
True
>>> isinstance(Mila, Dog)
False
```

```
>>> Haru = Dog('Haru')
>>> print(Haru.is_hungry)
True
>>> Haru.eat("Biscuit")
Haru eats Biscuit
>>> print(Haru.is_hungry)
False
```

More about [Class Inheritance and super\(\)](#)



# Now what?

The [Exercise 2](#) holds a simple example of the basic Python code flow syntax. It includes:

- A method
- A class
- A sub-class
- A if/elif/else case
- A for loop

## Exercise

---

1. Find all the Syntax Errors before running
2. Debug why the cat is not eating the Meat.
3. Create a new class called Dog and write how it behaves depending on the food it eats.
  1. Create an empty class
  2. Add the *docstring* explaining how it will behave
  3. Implement the actual code in a way it behaves as you explained
4. (Challenge) Add a "while" loop that will only stop when all the pets are not hungry anymore.



# Questions?

