

# PYTHON CLASS 5

---

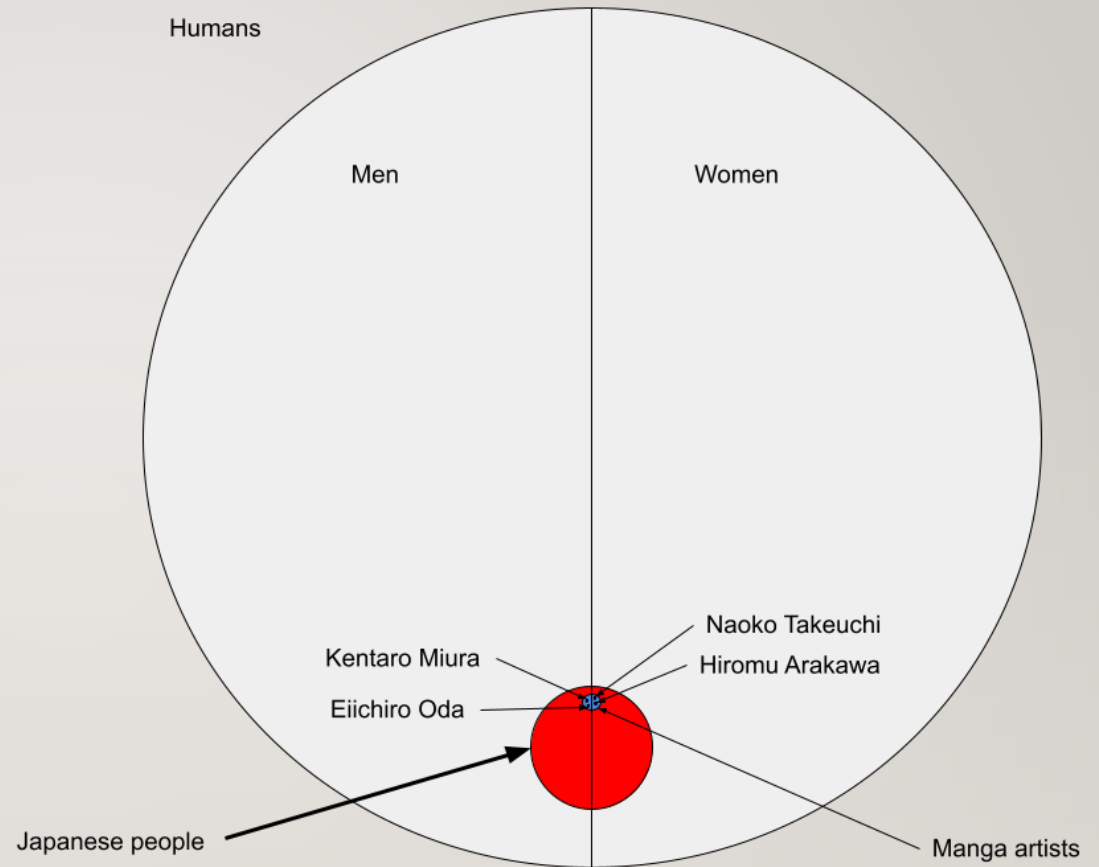
INSTANCE VERSUS CLASS DISTINCTION;  
FUNCTIONS



# INSTANCE VERSUS CLASS

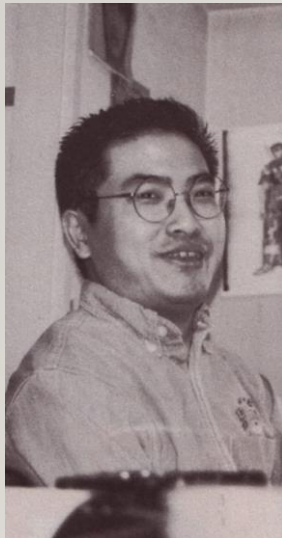
- There is an important difference between an instance and a class.
- The instance is the actual object (実体; 実例)
- The class is the group (種類).
- Humans, Japanese people, and Manga artists are all groups or classes.
- Kentarou Miura, Eiichiro Oda, Naoko Takeuchi, and Hiromu Arakawa are instances of each class.

# INSTANCE VERSUS CLASS

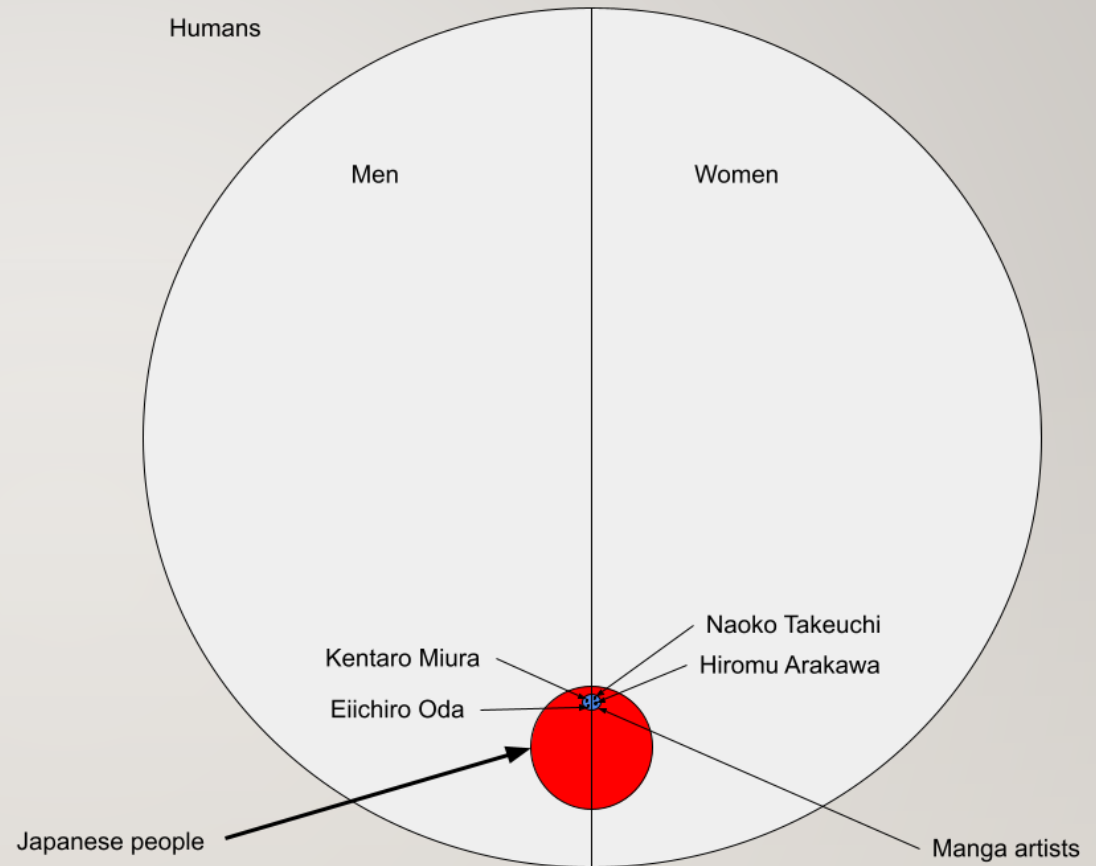


- The smaller groups and instances have all the properties of the larger groups.

For example, Kentaro Miura is a Japanese male manga artist. He belongs to all these groups, so he has the properties of each group.



name = "Kentaro Miura"  
is\_male = True  
can\_draw = True  
nationality = "Japanese"  
can\_vote\_in\_Japan = True



# INSTANCES IN PROGRAMMING

---

```
pet_name = "Angel"
```

`pet_name` is an *instance* of the string class.

We know it is a string because of the value we assigned it.

Because it is an *instance* of the string class, we can use string functions, like

```
pet_name.isupper()
```



# INSTANCES IN PROGRAMMING

---

```
pet_age = 8.6
```

`pet_age` is an *instance* of the float class.

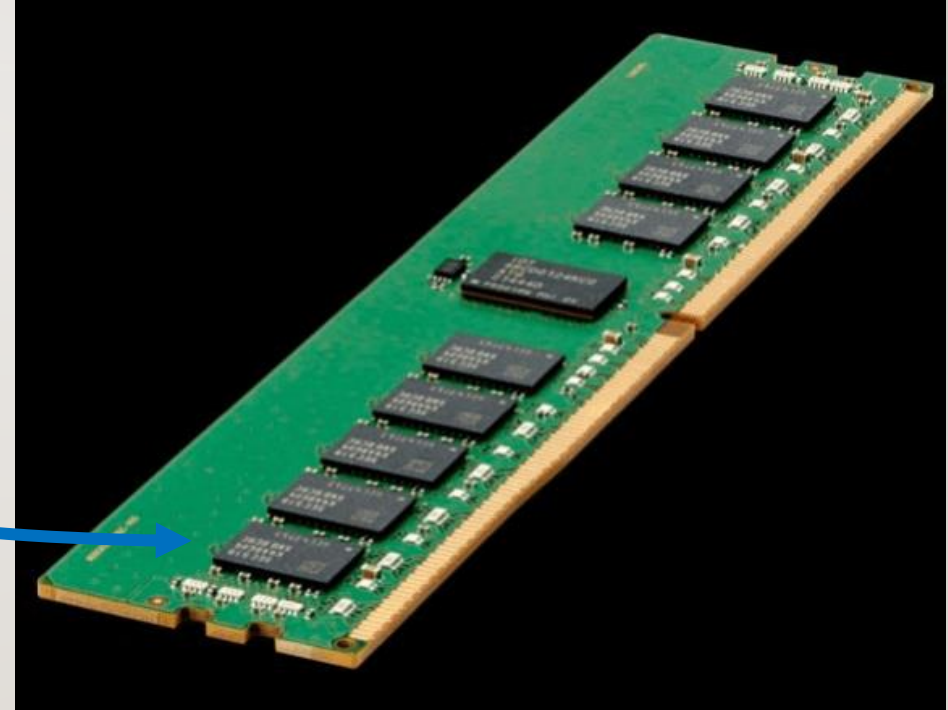
We know it is a float because of the value we assigned it.

Because it is an *instance* of the float class, we can use float functions, like

```
pet_age.hex()  
pet_age.is_integer()
```

# INSTANCES VERSUS CLASS: USING THE ID FUNCTION

- Make 4 variables – a string, float, int, and bool.
- Use `id(variable)` to see the variable's *memory location* on the computer.
- Use `id(type(variable))` to check the memory location of the variable's type.



# INSTANCES VERSUS CLASS: USING THE ID FUNCTION

- What do you notice about the length of the id?
- Try running this several times.
- What do you notice now?





# LITERALS

- The data that we store in variables is mostly *literals* ([リテラル - Wikipedia](#))



# USING FUNCTIONS

- Let's practice using some functions.
- Remember, you can use this website if you need help. [Built-in Types — Python 3.10.4 documentation](#)
- Or, you can do this:

```
help(variable)
```

```
help(variable.function)
```

# EXAMPLE

---

```
first_name = "steven"
```

```
last_name = "erikson"
```

```
#because these are names, we want to make sure they are capitalized!
```

```
first_name = first_name.upper()
```

```
last_name = last_name.upper()
```

```
#let's print the names together
```

```
print(first_name + " " + last_name)
```

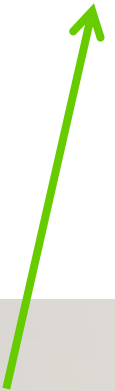
# CREATING FUNCTIONS

- We can create a function using `def`




# FUNCTION SYNTAX

```
def my_function():  
    print("Hello, Hello")
```



When you create a new function, you must use the keyword (予約語) **def**



**def** is an abbreviation (略語) of define (定義する) or definition (定義)

# FUNCTION SYNTAX

---

```
def my_function():  
    print("Hello, Hello")
```

When you create a new function, you must use the keyword (予約語) `def`

This is the function name. When we use the function, we will call (叩いた) `< >` this name. Function names follow the same rules as variable names.

# FUNCTION SYNTAX

---

```
def my_function():  
    print("Hello, Hello")
```

This is the function name. When we use the function, we will call (呼び出す) this name. Function names follow the same rules as variable names.

The parentheses hold any arguments (引数). This function does not have an argument.

# FUNCTION SYNTAX

---

```
def my_function():  
    print("Hello, Hello")
```


The parentheses hold any arguments (引数). This function does not have an argument.

Each function definition (定義) must end with a colon.



# FUNCTION SYNTAX

```
def my_function():  
    print("Hello, Hello")
```



The contents (中身) or commands (命令) of the function.

# FUNCTION SYNTAX

```
def my_function():  
    print("Hello, Hello")
```

The commands (命令) we  
want the function to do  
must be indented (凹む) 4  
spaces.

The contents (中身) or commands (命令) of the function.

# THIS IS IMPORTANT!

- The code below looks similar, but it is not the same!

```
def combine_string(a, b):  
    print(a+b)
```

```
def combine_string(a, b):  
    print(a+b)
```

# SPACING

- In Python, the number of spaces at the start of a line is really important.
- The number of spaces tells Python what lines belong together.

```
first_name = "David"  
last_name = "Hunter"  
def combine_string(a, b):  
    print(a+b)  
    print(b+a)  
print(first_name+last_name)
```

```
first_name = "David"  
last_name = "Hunter"  
def combine_string(a, b):  
    print(a+b)  
    print(b+a)  
print(first_name+last_name)
```



# SPACING

- If the number of spaces at the front do not match, Python will give you errors!!!!

```
first_name = "David"  
last_name = "Hunter"  
def combine_string(a, b):  
    print(a+b)  
    print(b+a)  
print(first_name+last_name)
```

```
first_name = "David"  
)last_name = "Hunter"  
def combine_string(a, b):  
    print(a+b)  
    ←print(b+a)  
print(first_name+last_name)
```

# USING A FUNCTION


---

- To use a function you have created, just use the function name, parentheses, and any arguments inside the parentheses.

```
def my_function():  
    print("Hello, Mr. Hunter!")
```

```
my_function()
```

Using the function, or  
calling the function



# USING A FUNCTION, 2

---

- To use a function you have created, just use the function name, parentheses, and any arguments inside the parentheses.

```
def my_function(name):  
    print("Hello, " + name + "!")
```

```
my_function("Sara")
```

```
my_function("Aoi")
```

Using the function, or  
calling the function

```
def my_function(name):  
    print("Hello, " + name + "!")
```

```
student_1 = "Aoi"  
student_2 = "Sara"
```

```
my_function(student_1)
```

```
my_function(student_2)
```

# USING A FUNCTION, 3

---

- You **cannot** use a function before you define it.

```
my_function("Tom")
```

```
def my_function(name):  
    print("Hello, " + name + "!")
```



# USING A FUNCTION, 3

---

- You **cannot** use a function before you define it.



```
my_function("Tom")
```

```
def my_function(name):  
    print("Hello, " + name + "!")
```

# USING A VARIABLE

---

- This is the same as for variables.



```
my_function("Tom")
```

```
def my_function(name):  
    print("Hello, " + name + "!")
```



```
print(name)
```

```
name = "Thomas"
```

# CREATE SOME STRING FUNCTIONS

---

- Greet someone
- Combine strings
- Print a recipe
- Print lyrics

# CREATE SOME MATH FUNCTIONS

- Add two numbers.
- Subtract two numbers.
- Divide two numbers
- Multiply two numbers.
- Compare two numbers.



# WHY DO WE HAVE FUNCTIONS?



# FUNCTIONS AND SCOPE (有効範囲)

- If you create a variable inside a function, the scope of the variable is the function.
- This means, you cannot use the variable outside the function.



# ADVANCED FUNCTIONS

- Besides doing things, functions can give back values.
- To give back a value, we use the **return** keyword ( 予約語 ).

```
def dog_age(human_age):  
    return human_age * 7  
  
my_age = 39  
  
my_dog_age = dog_age(my_age)  
print(my_dog_age)
```

# ADVANCED FUNCTIONS, 2

```
def dog_age(human_age):  
    return human_age * 7
```

```
my_age = 39
```

```
my_dog_age = dog_age(my_age)  
print(my_dog_age)
```

What is the type  
my\_dog\_age?

How can you check?



# ADVANCED FUNCTIONS, 3

```
first_name = "David"
last_name = "Hunter"
def combine_string(a, b):
    return a+b

name = combine(first_name,last_name)
```

What is the type name?

How can you check?

# MAKE AN ADVANCED FUNCTION

- Pick some of our functions from before and make them return a value.
- Use the `type()` function to test the return value.
- Assign the return value to a variable.
- Print the variable.

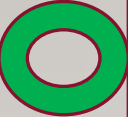
ゆうこう はん い

# FUNCTIONS AND SCOPE (有効範囲)

- If you create a variable inside a function, the scope of the variable is the function.
- This means, you cannot use the variable outside the function.



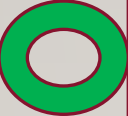
# FUNCTIONS AND SCOPE (有効範囲)



```
def cube(n):  
    a = n * n * n  
    return a
```



```
print(cube(5))
```



```
num = 10
```



```
print(cube(num))
```



```
print(a)
```

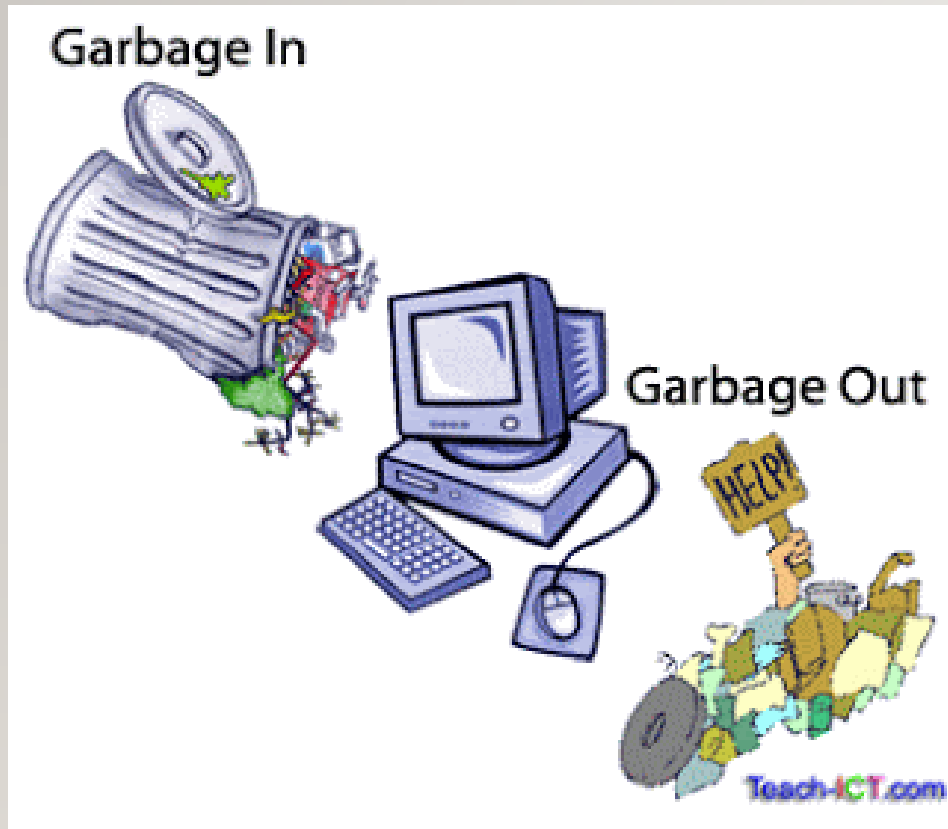


# MAKE AN ADVANCED FUNCTION

- Pick some of our functions from before and make them return a value.
- Use the `type()` function to test the return value.
- Assign the return value to a variable.
- Print the variable.



# INPUT (入力) AND OUTPUT (出力)



# OUTPUT

- Using print
- Use + to combine different strings.
- If you want to combine a string and another variable type (integer, float, boolean), you must cast (かたへんかん 型変換) it.
- Python has 4 built-in casting functions:

int()	float()	bool()	str()
-------	---------	--------	-------

# GETTING INPUT FROM THE USER

---

- You can use the function `input()` to get information from the user.
- This input is ALWAYS formatted as a string.
- If you want to use it in a different way, you need to cast (型変換) it to another type.



# USING INPUT

---

- Open IDLE.
- Use `input()` to receive a name or string from the user.
- Call a function to do something with this name or string.

# USING INPUT,2

---

- Use `input()` to receive a float or int from the user.
- Call a function to do something with this float or int.