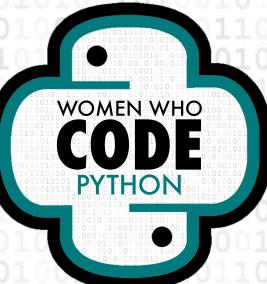


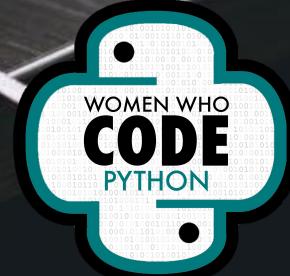
# Welcome everyone!

- You can find these slides on GitHub here:  
<https://github.com/WomenWhoCode/WWCodePython>
- Please make sure your chat is set to “All panelists and attendees”.
- Some housekeeping rules:
  - Everyone will be muted throughout the webinar, but there will be opportunities for participation!
  - Please share your thoughts on the chat and/or ask questions in the Q&A.
  - The entire team is here today. Please reach out to us with any technical questions!



# Women Who Code: Python

Beginner Python Study Group  
Session 7: Write Your Own Module





Meet us!

# Hi I'm Rishika!

Graduate Student  
Lead at WWCode  
@Rishika Singh



# Hi I'm Karen!

R&D - AI Sector  
Volunteer at WWCode  
@Karen W



# Hi I'm Stephanie!

Marketing  
Evangelist @ WWCode  
@Stephanie



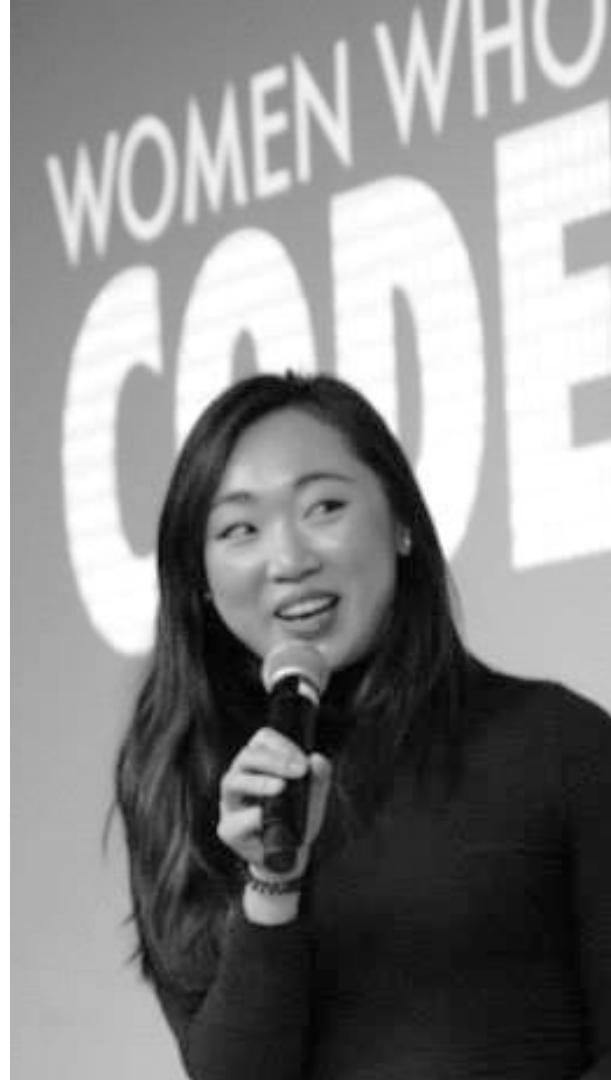
# OUR MISSION

Inspiring women to  
excel in technology  
careers.



# OUR VISION

A world where women are representative as technical executives, founders, VCs, board members and software engineers.



# OUR TARGET

Engineers with two or more years of experience looking for support and resources to strengthen their influence and levelup in their careers.



# CODE OF CONDUCT

**WWCode is an inclusive community**, dedicated to providing an empowering experience for everyone who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, creed, political affiliation, or preferred programming language(s).

Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome. We do not tolerate harassment of members in any form. Our **Code of Conduct** applies to all WWCode events and online communities.

Read the full version and access our incident report form at [womenwhocode.com/codeofconduct](http://womenwhocode.com/codeofconduct)

# 230,000 Members

70 networks in 20 countries  
Members in 97+ countries  
10K+ events  
\$1025 daily Conference tickets  
\$2M Scholarships  
Access to [jobs](#) + [resources](#)  
Infinite connections



# OUR MOVEMENT

As the world changes, we can be a connecting force that creates a sense of belonging while the world is being asked to isolate.



# Upcoming Events!

WOMEN WHO

FRI  
27  
NOV

## ? Ask Me Anything(AMA) with Patricia Tillotson ?

3:00 PM - 4:00 PM (EST) | 🔍 Zoom

[Register](#)

SUN  
29  
NOV

## ★ Discover NLP with Python Study Group: Vectorization ★

7:00 PM - 8:00 PM (EST) | 🔍 Zoom

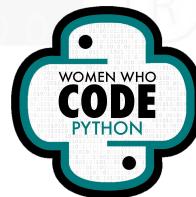
[Register](#)

WED  
02  
DEC

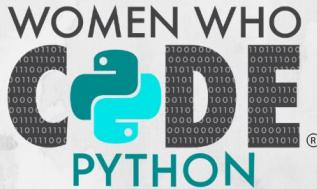
## ★Beginner Python Study Group ★Session 8: Mini Project Recurring

7:00 PM - 8:30 PM (EST) | 🔍 Zoom

[Register](#)



# Stay Connected



WOMEN WHO  
CODE.<sup>®</sup>  
PYTHON

JOIN US ON SOCIAL MEDIA!

@WWCODEPYTHON

[WOMENWHOCODE.COM/PYTHON](http://WOMENWHOCODE.COM/PYTHON)



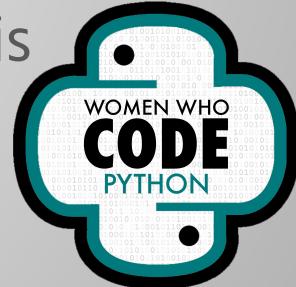
# Today's Agenda

1. Module
2. Definition  $\stackrel{\text{def}}{=}$
3. Class 
4. Print function 
5. Comments #
6. Wrap-Up 

# Module

- A file containing Python definitions and statements
- .py extension (like a regular Python script file)

...Confused? No worries! You'll understand by end of this session.

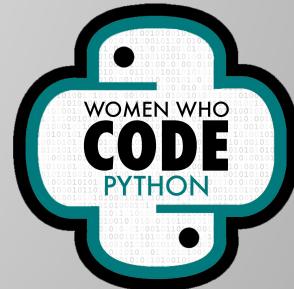


# Function Definition

- Function blocks begin with keyword def followed by the function name and parameters
- Return value is not mandatory

Just picture it...

- Instead of writing the happy birthday song over and over, we can use a function to call out the lyrics!



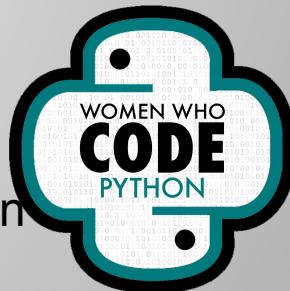
# Function Definition

```
# create category list, folder name list and  
# label lists from data directory  
def getlists(path): ←  
    # create empty lists for file name and labels  
    file_names, labels = [], []  
    # read files in directory  
    folder_names = os.listdir(path)  
  
    # loop folder names and split 2 variables  
    for f in folder_names:  
        file_names.append(f)  
        labels.append(f.split('_')[0])  
  
    # define and encoder classes  
    category = sorted(set(labels))  
    le = LabelEncoder()  
    le.fit(category)  
    labels = labels2cat(le, labels)  
  
    # return values  
    return category, file_names, labels ←
```

Define function and parameter(s)

Actions

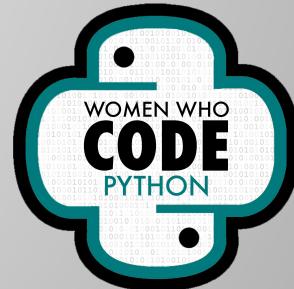
Return values of the function



# Class

- Python as an “object-oriented programming language”, all implemented codes can be referred as an object, thus objects can be grouped together
- Both variables and functions can be called within the class by **classname.variable** or **classname.function()** outside of the class
- Using **self** to use the variable or function within the class

```
class myname:  
    name = 'Karen'  
  
print('Hi I am ', myname.name)
```



# Now we put the 2 together!

```
class janitor:  
    ''' housekeeping tasks'''  
  
    # import parameters / set paths  
    def __init__(self, root_path, folder_name):...  
  
    def dup_gif(self):...  
  
    def dup_png(self):...  
  
    def noface_gifcsv(self):...  
  
    def noface_gif(self):...  
  
    def noface_png(self):...  
  
    def nogoodsize_gif(self):...  
  
    def remove_bysize(self):...  
  
    def inconsistent(self):...  
  
    def remove(self):...  
  
root_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), '..', '..')  
run = janitor(root_path, 'facial_pinterest')  
run.remove()
```

Class name

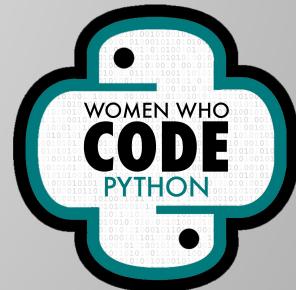
Class description

Initiate all the input values

List of functions

Input parameter values

Execute one of the functions from class



# def vs class

```
def myname():
    name = 'Karen'
    return name

print('Hi I am ', myname())
>>Hi I am Karen
```

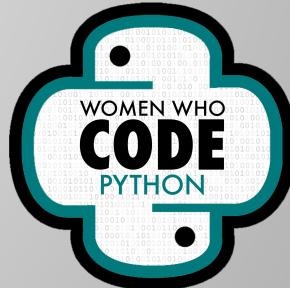
VS

```
class myname:
    name = 'Karen'

print('Hi I am ', myname.name)
>>Hi I am Karen
```

## Quick Memo ❤:

- To call out a function, type function name followed by bracket "()"
- To call out a variable, just type the variable name



# Print() function

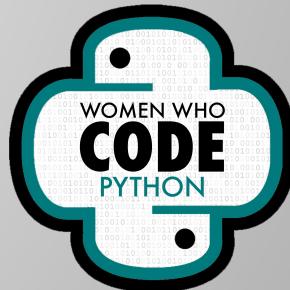
2 ways (or more) to print a statements with values: % and {}

```
name = 'Mr. Bean'  
age = 65  
item = 'his teddy bear'  
print('This is is %s.'%(name))  
print('%s is %d years  
old.'%(name,age))  
print('His favourite companion is  
%s.'%(item))
```



```
name = 'Mr. Bean'  
age = 65  
item = 'his teddy bear'  
print('This is is {}'.format(name))  
print('{} is {:d} years  
old.'.format(name,age))  
print('His favourite companion is  
{}'.format(item))
```

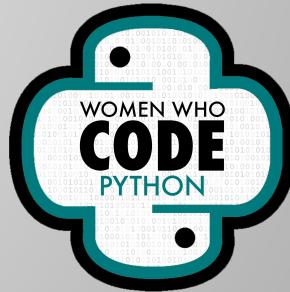
This is is Mr. Bean.  
Mr. Bean is 65 years old.  
His favourite companion is his teddy bear.



# Lost? Comment More!

BAD example (with no comments)

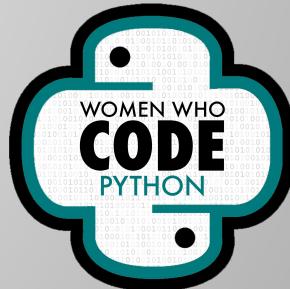
```
# ----- Video Clip - avi to gif ----- #
def videoclip(vfile, start_frame, end_frame, cate):
    video = editor.VideoFileClip(os.path.join(path_video, vfile))
    clip = video.subclip(start_frame, end_frame)
    clip.write_gif(os.path.join(path_gif, '_ori_gif', 'o_' + cate + '.gif'))
    print('created clip for ', cate)
```



# Lost? Comment More!

BETTER example (with comments!!)

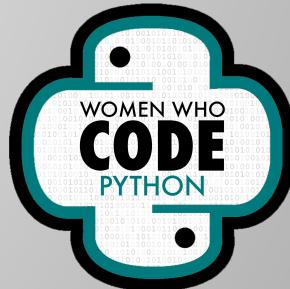
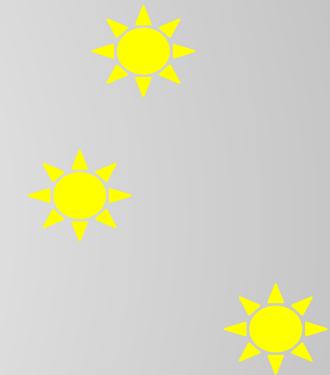
```
# ----- Video Clip - avi to gif ----- #
def videotoclip(vfile, start_frame, end_frame, cate):
    # access the video file with its associated path
    video = editor.VideoFileClip(os.path.join(path_video, vfile))
    # indicate the time frame the video clip starts and ends
    clip = video.subclip(start_frame, end_frame)
    # use the function write_gif to save clipped video
    clip.write_gif(os.path.join(path_gif, '_ori_gif', 'o_' + cate + '.gif'))
    # indicate what category it is clipping for
    print('created clip for ', cate)
```



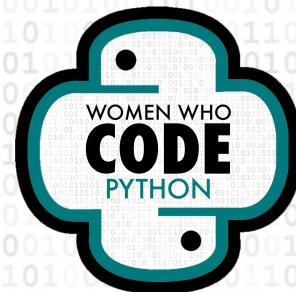
# Lost? Comment More!

GOOD example (with comments and description!!)

```
# ----- Video Clip - avi to gif ----- #
def videoclip(vfile, start_frame, end_frame, cate):
    '''to make .gif video clips from original video
    parameters: original video file (vfile), start frame(start_frame),
    end frame(end_frame) and category(cate)'''
    # access the video file with its associated path
    video = editor.VideoFileClip(os.path.join(path_video, vfile))
    # indicate the time frame the video clip starts and ends
    clip = video.subclip(start_frame, end_frame)
    # use the function write_gif to save clipped video
    clip.write_gif(os.path.join(path_gif, '_ori_gif', 'o_' + cate + '.gif'))
    # indicate what category it is clipping for
    print('created clip for ', cate)
```



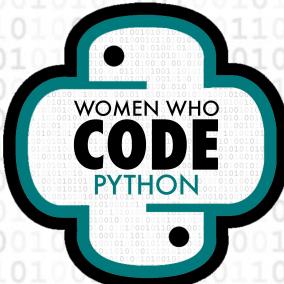
# Q&A Time!



# Time for Live Coding!

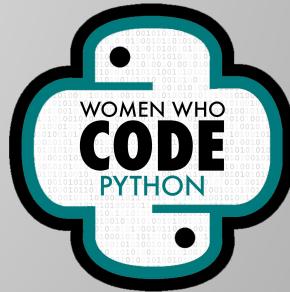
Google Colab link:

<https://colab.research.google.com/drive/1aBujVG9CFXUkqJ0Irc4wR7ozhUEGMOD7#scrollTo=4a6XJXZdcdlv>



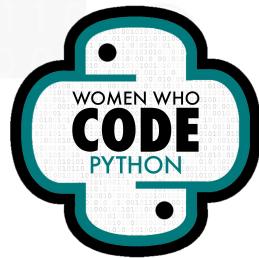
# Wrap-Up

- Module - a file contains list of functions/classes
- Definition - a function
- Class - a class with groups of functions / variables
- Print function - print()
- Comments - helps you to figure out where the bugs are



# WOMEN WHO Questions?

Join our Slack channel: #beginner-python-stdy-grp



# Thanks, everyone!

