PYTHON CLASS 11

LISTS

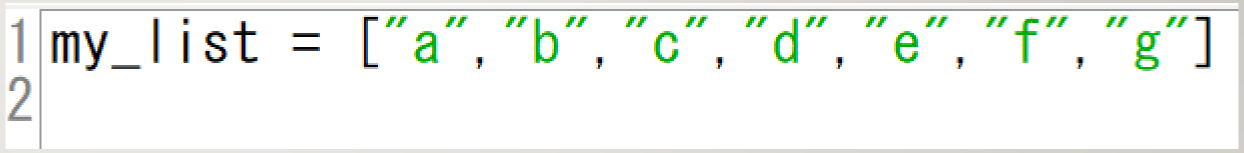
LISTS

- A list is a group of things.
- It is a type of variable, and it can have lots of information.
- The index(添字) of lists starts at 0.

PYTHON LIST

- A list in Python needs square brackets [ ].
- Each item in the list must be

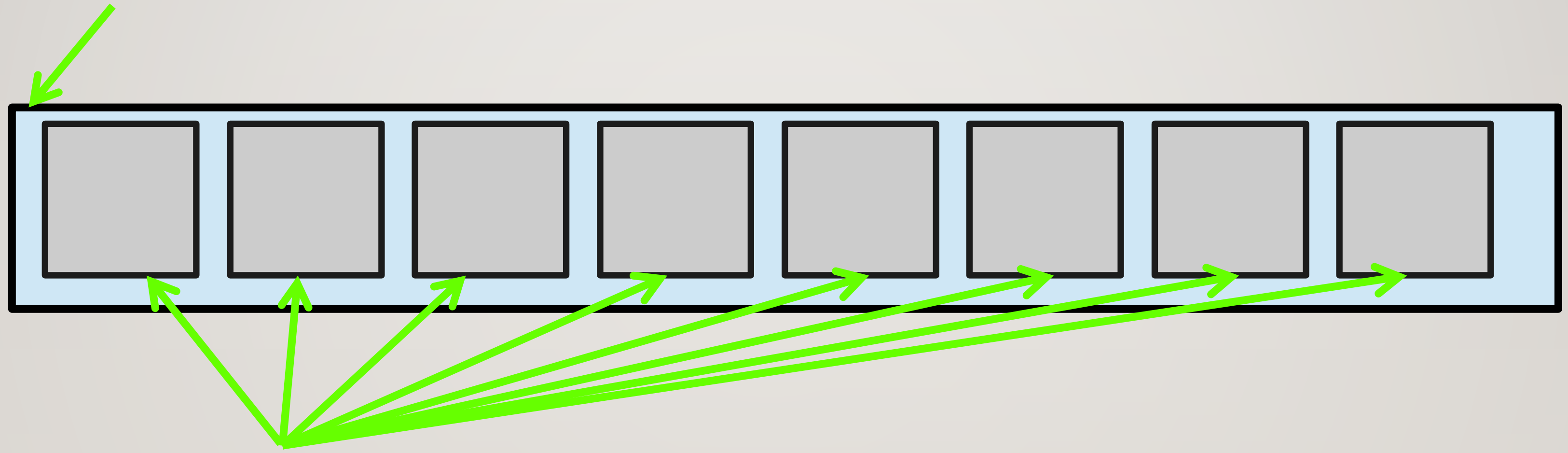
  separated by a comma **,**



```
my_list = ["a","b","c","d","e","f","g"]
```

LIST STRUCTURE

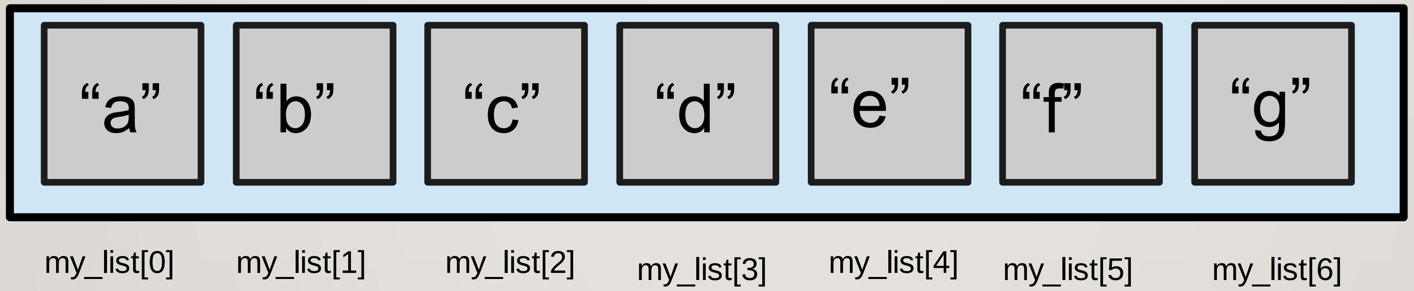
- The list is like one big box



The list has many smaller boxes inside it.
These smaller boxes each hold different information.

LIST STRUCTURE, 2

- my_list = [“a”,“b”,“c”,“d”,“e”,“f”,“g”]



MAKING A LIST

- Make a list that has the names of all your family members in it.

```
family = ["Kathy","Doug","Michael","David","Mark","Sarah","Rachel"]
```

- Access each member in the list and print it. We use square brackets **[ ]** to access an item.

```
print(family[5])
```

- Change the value of a member. Again, we use square brackets **[ ]**.

```
family[0] = "Katherine"
```

LIST PRACTICE

- Change the value of a list item.
- Use the in keyword for a list.
- Use the len() function for a list.
- Make lists of strings, booleans, integers, and floats.
- Make a mixed list.

LIST PRACTICE, 2

- Check the type of the list.
- Check the type of the items in a list.
- Check the last item using -1.
- Use list.insert(index, item)
- Use list.append(item)
- Use list.remove(item)
- Use list.pop(index)

LIST GUIDED PRACTICE

- Write code that checks if 56 is in my_list. If it is, print the index.
- my_list = [56,37,82,2340,5]

COMMON ERRORS

SPELLING

- Name error

```
n = “Tom”

print(na)
```

```
if x.isupper():
	print(“It’s upper! Yay!”)
```

- Capitalization

```
import datetime

date = datetime.date(2022,6,21)
print(Date.weekday())
```

INDENTATION

- Your indentation doesn't match the rest of the code.

```
import datetime


    date = datetime.date(2022,6,21)
 print(Date.weekday())
```

- Your indentation doesn't match what you want to do.

```
import datetime
date = datetime.date(2022,6,21)
if Date.weekday() == 1:
      print("It's Tuesday!")

print("It's a weekday")
```

NUMBERS

- Zero division error
  - you are trying to divide a number by 0. Check where it happens.

```
b = 0
a = 7 / b
```

- Index error
  - you tried to use an index that is larger (smaller) than the list

```
li = [“Arthur”,“Lancelot”,“Gawain”,“Kay” ]
print(li[4])
print(li[-5])
```

TYPE/ASSIGNMENT

- Type error

```
import datetime
s = “Tuesday”
d = datetime.date(2022,6,22)
x = d + s
```

- Using assignment instead of equality
  - = versus ==

```
li = [“Arthur”,“Lancelot”,“Gawain”,“Kay” ]
li2 = [“Arthur”,“Lancelot”,“Gawain”,“Kay” ]

li = li2
print(li == li2)
```

FUNCTION RESULTS

- Remember, some functions give a return value, but some give None.
- You need to know which is which.

```
l = [“Hayao Miyazaki”, “Kentarou Miura”, “Takehiko
Inoue”]

l = l.append(“Kohei Horikoshi”)
```

append doesn’t have a return value, so now l is lost!!!

GLOBAL VS. LOCAL / ATTRIBUTE

- Unbound local error
  - You tried to assign a value to a global variable inside a function

```
x = 5
def sq():
    x = x**2
```

- Attribute error
  - you tried to access a function or piece of information that doesn’t exist

```
li = [“Arthur”,“Lancelot”,“Gawain”,“Kay” ]
li.isupper() #only exists for str variables
li.ascii_uppercase # only exists for the special string library, which you need to import
```

LOOPS AND LOOPING

REPITITION

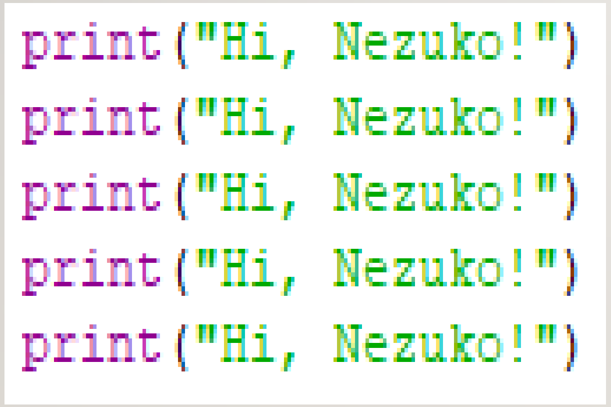
- https://youtu.be/KbiSxunJatM?t=34

GROUNDHOG DAY

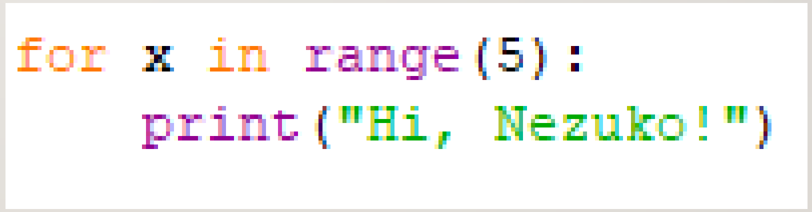
- Groundhog Day (Clip 3) - Repeated Dying Sequence - YouTube

LOOPS

- A loop is a way to repeat a statement or command.

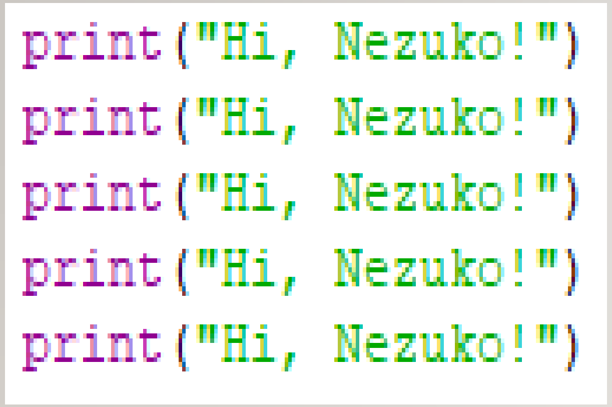


```
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
```

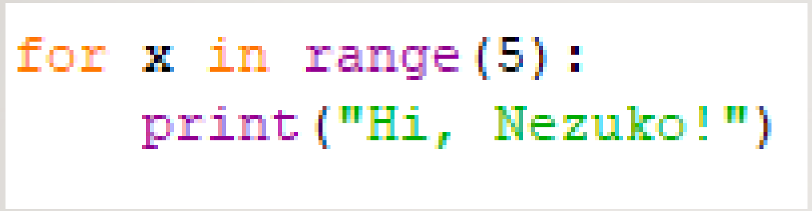


```
for x in range(5):
    print("Hi, Nezuko!")
```

LOOPS



```
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
```

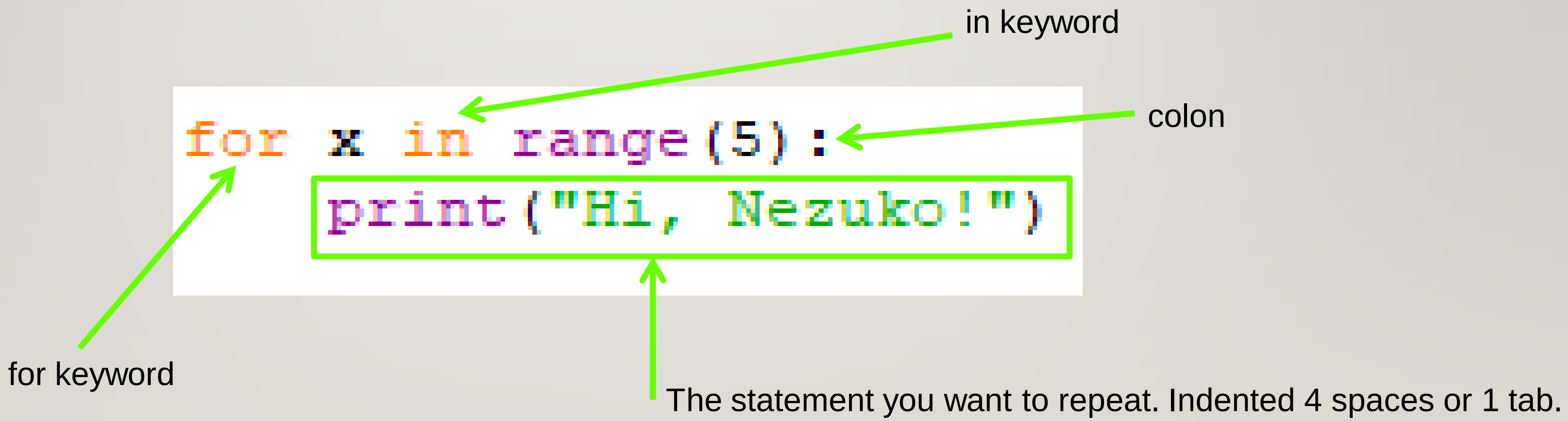


```
for x in range(5):
    print("Hi, Nezuko!")
```

What are loops good for?

LOOPS, CONTINUED

- In Python, the most basic kind of loop is a for loop.
- A for loop has this syntax:



```
for x in range(5):
    print("Hi, Nezuko!")
```

in keyword
colon
for keyword
The statement you want to repeat. Indented 4 spaces or 1 tab.

LOOPS AND LISTS

```
days = [“Mon”, “Tues”, “Wed” , “Thurs” , “Fri”, “Sat”, “Sun”]

for x in days:
        print(x)
```

SPACES VERSUS TABS

- https://www.youtube.com/watch?v=SsoOG6ZeyUI

WAYS TO CONTROL A FOR-LOOP

- Use range(a)
- Use range(a,b)
- Use in + a list
- Use len(a)

LOOPS, IF AND SCOPE

- Loops and **if** work very differently than functions.

```
def my_func(a):
      x = 5 * (a*a) + 11

my_func(10)

print(x) # this gives an error
```

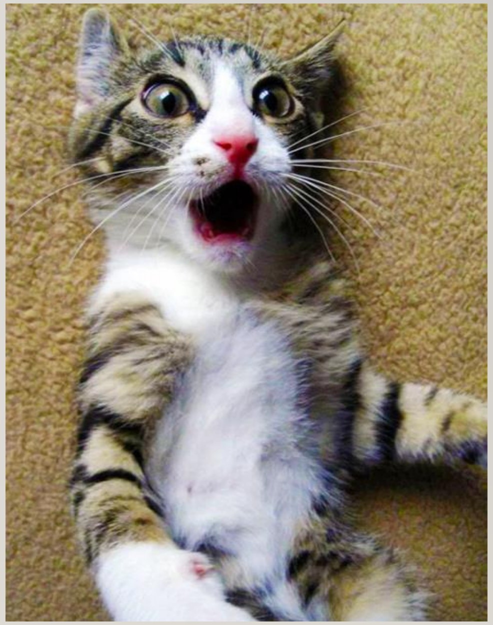
```
for x in range(10):
    a = 2 * x
    print(a)

print(a) #this is fine!
print(x) #this is fine, too!
```

```
n = "y"

if n == "w":
    b = "y2"
else:
    c = "x"

print(b) #this gives an error
print(c) #this is fine!
```



LOOPS AND SCOPE

- If you need to keep a value while running a for-loop, you should make the variable before it.

```
for x in range(10):
	a = 0
	a = a + x

print(a)
```

a is set to 0 each time you run through the loop, so the last value is 9…

```
a = 0

for x in range(10):
	a = a + x

print(a)
```

a is set to 0 outside the loop, so previous value gets added to it each time

LOOP EXERCISES

- Create a loop that counts from 0 to 100
- Create a loop that multiplies the numbers from 1-20.
- Create a loop that adds random numbers to a list.
- Use a loop to find the largest and smallest numbers, and the average.

WAYS TO CONTROL ACTION INSIDE A LOOP

- You can use if-statements and break or continue to control action

```
li = [2,4,6,8,10,11,14,16,18]

for x in li:
    if x % 2 != 0:
        break
    print(x)
```

we want to stop the loop completely if we have an odd number

```
li = [1,4,9,16,25,0,36,49,64]

for x in li:
    if x == 0:
        continue
    else:
        print((li.index(x) +1) / x)
```

we just want to skip the number to avoid dividing by 0