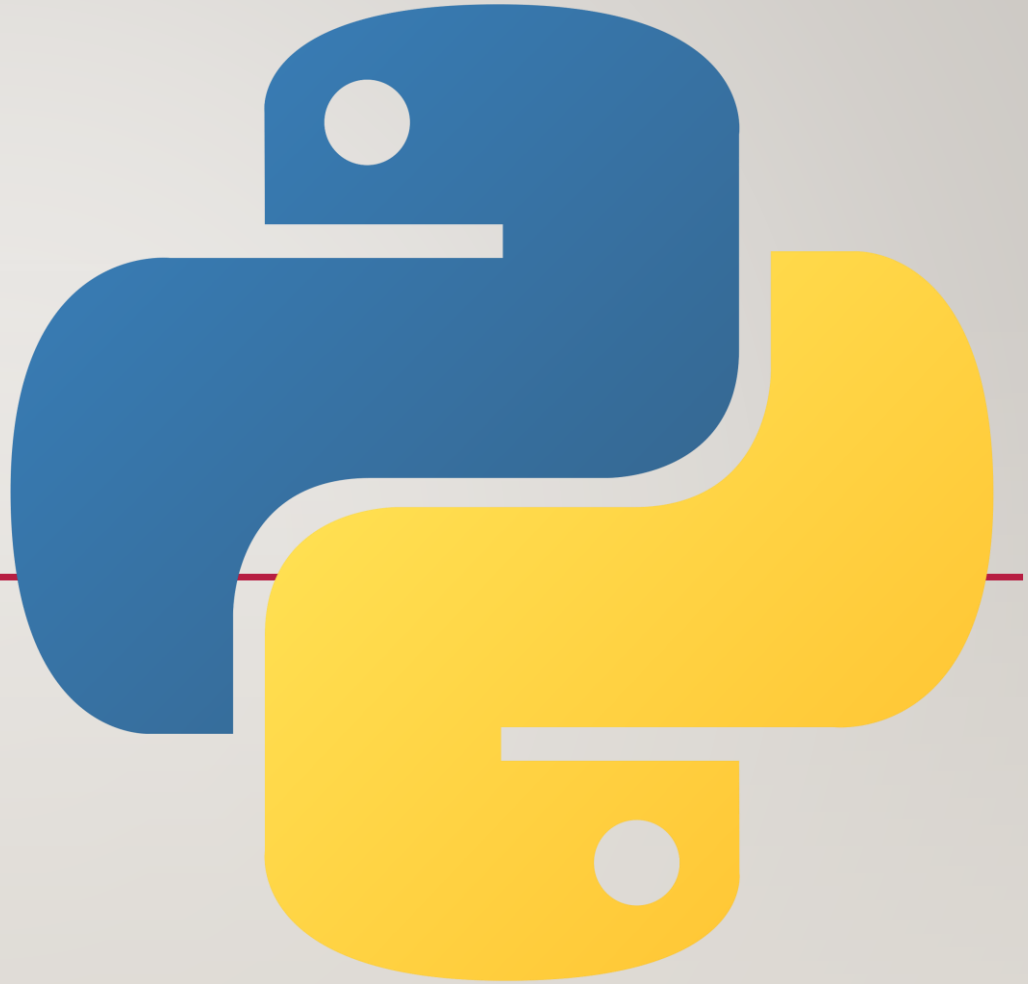
PYTHON CLASS 13

INDENTATION REVIEW AND LISTS



INDENTATION

- In Python, code with the same level of indentation is grouped together.

```
import datetime

user_name = "Mr. Hunter"

name = input("What is your name?")
if name == user_name:
        print("Hello, " + user_name)
else:
        print("You are a new user")
        b_day_str = input("What is your birthday? (YYYY-MM-DD)")
        b_day_str = b_day_str.split("-") #creates a list [YYYY,MM,DD]
if b_day_str[0] >=  datetime.date.today().year: #causes error if user
                                                #name is Mr. Hunter
        print("you are too young, or you are from the future")
```

```
import datetime

user_name = "Mr. Hunter"

name = input("What is your name?")
if name == user_name:
        print("Hello, " + user_name)
else:
        print("You are a new user")
        b_day_str = input("What is your birthday? (YYYY-MM-DD)")
        b_day_str = b_day_str.split("-") #creates a list [YYYY,MM,DD]
        if b_day_str[0] >=  datetime.date.today().year:
                print("you are too young, or you are from the future")
        print("Last message")
```

INDENTATION, 2

- Grouping code is a really important idea in programming.
- Every modern programming language allows programmers to group code -> Java, Python, C#, C++, etc.
- Grouping is a little different in other languages.

C#/C++/JAVA GROUPING

```
int number = 0;
public void Add(int x)
{
        number +=x;
}

public void Subtract(int x)
{
        number -=x;
}

bool BiggerThan(int x)
{
        return number > x;
}
```

```
if (BiggerThan(7))
{
        Add(20);
        if(BiggerThan(20))
        {
                printf("Really big");
        }
        else
        {
                printf("OK");
        }

}
else
{
        Subtract(5);
        if(BiggerThan(12))
        {
                printf("Still big");
        }
}
```

INDENTATION, 3

- Each group of code has the same indentation level.

```
import datetime


def GetBirthYear(b_day):
        b_day = b_day.split("-") #creates a list [YYYY,MM,DD]
        return int(b_day[0]) #change to an integer and give back

b_day_str = input("What is your birthday? (YYYY-MM-DD)")

year = GetBirthYear(b_day_str)

if year <= 1996 and year > 1980:
        print("You are a Mi")
        print("You are a Mil")
        print("You are a Millennial")
        if year == 1982:
                print("You were born in the best year!")
elif year <= 1980 and year > 1965:
        print("You are a Gen Xer")
elif year > 1996:
        print("You are a Zoomer")
        print("You are a Zoomer")
else:
        print("You are really old")
```

LIST PRACTICE, 3

- Go back to the Vending Machine Project.
- Replace the product variables with one list → product_list = ["Black Coffee", *other products*]
- Fix the rest of the code so that the program runs correctly.
- Let's make a new variable, current_product_index.
- This will make the next step easier.

LIST PRACTICE, 3

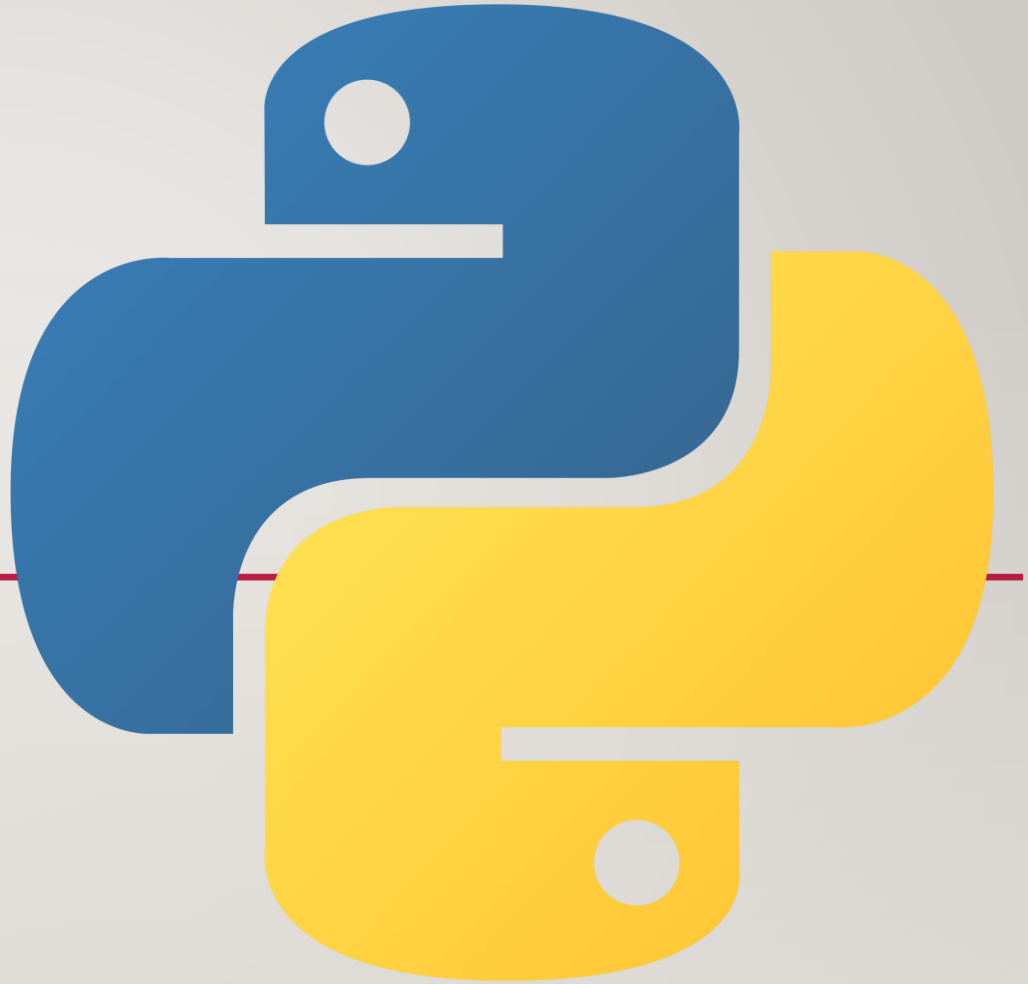
- Make a list of product prices. → product_price_list = [130, 160, *other products' prices*]
- Make a list of product amounts. → product_amount_list = [50, 20, *other products' amounts*]
- Fix the rest of the code so that the program runs correctly.

EVALUATE THE CHANGES IN VENDING MACHINE

Do you think using lists is better or worse than before?

Why?

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!!!

FUNCTION ARGUMENTS

- Python always expects a function's arguments to be in the same order.

```
name =  "Kentarou Miura"

name = name.replace("Taro", "Kentarou") # name.replace("taro", "Kentarou")

print(name) # name is still Kentarou Miura!!
```

this will not give an error, but the replace function expects the part we want to find **first**, **then** what we want to change it to

GLOBAL VS. LOCAL / ATTRIBUTE

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

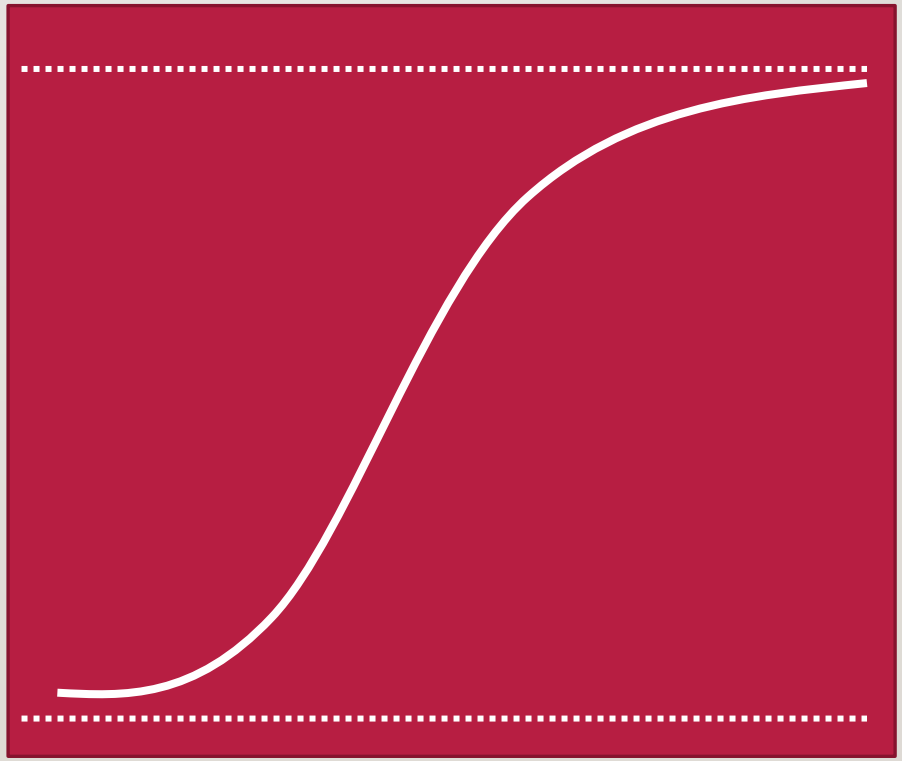
```
x = 5
def sq():
    x = x**2
```

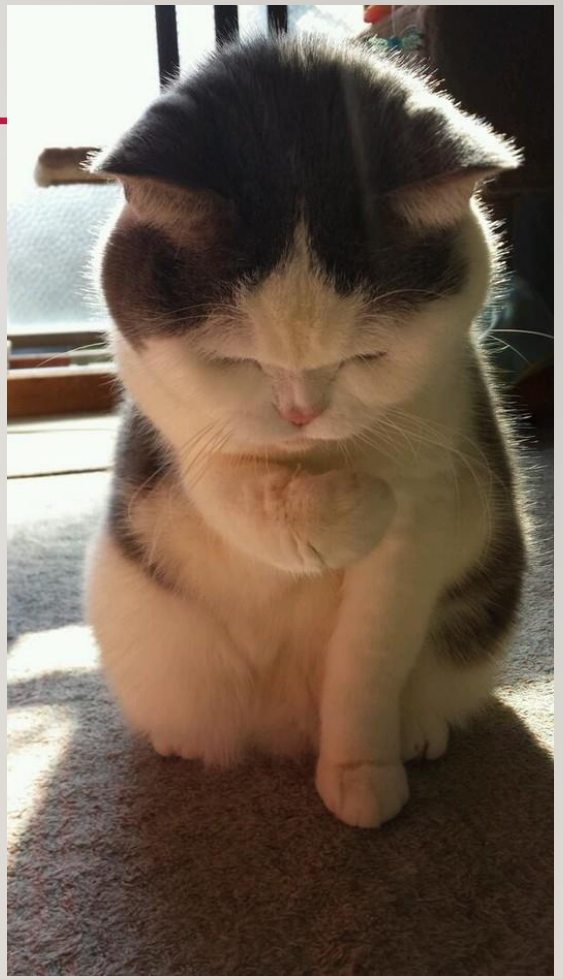
- Attribute error
  - you tried to access a function or piece of information that doesn't exist

```
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
```

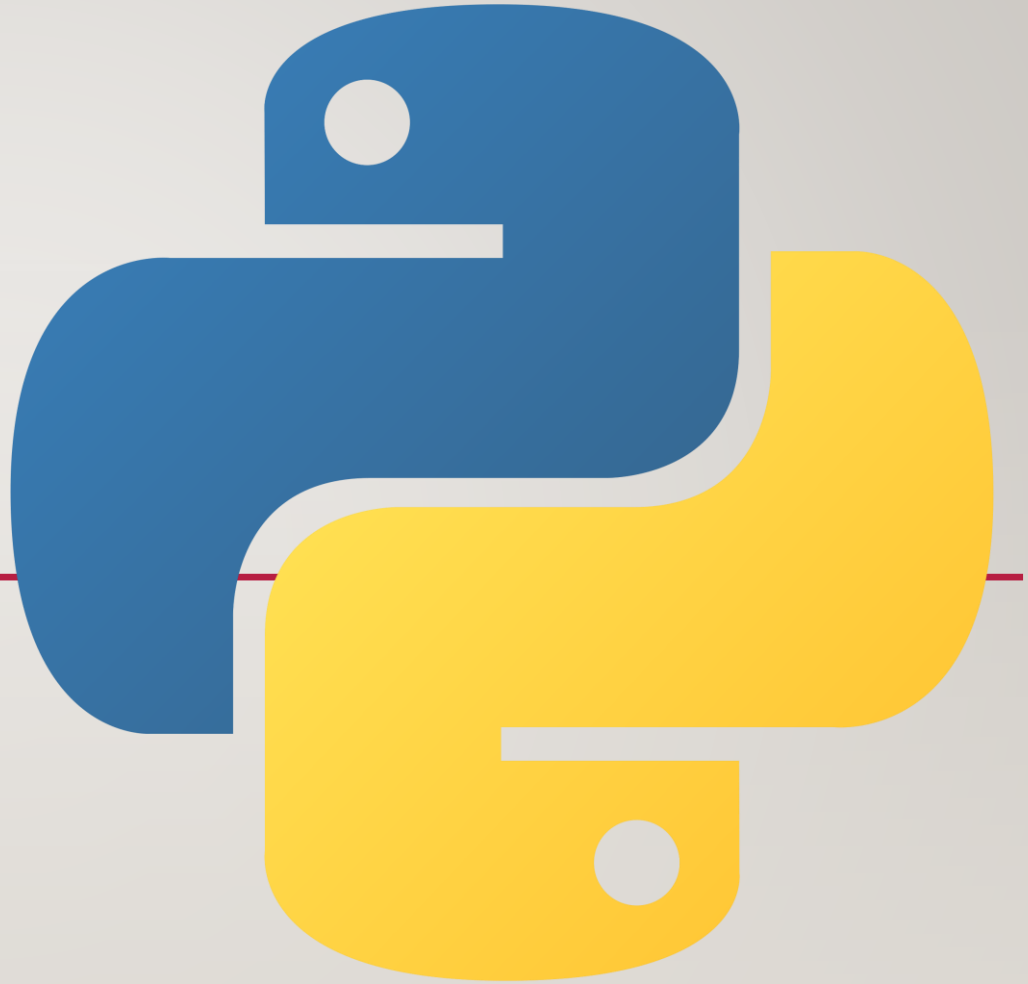
AND MANY MANY MANY MANY MORE

- As you get more experience as a programmer, you will encounter lots of other bugs and errors while you program.
- Also, you will become faster at finding them.
- You will never write perfect code, but that is OK!
- In life, we are all on a journey and we can try to get better as we go.





LOOPS AND LOOPING



REPETITION

- 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.

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

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

LOOPS

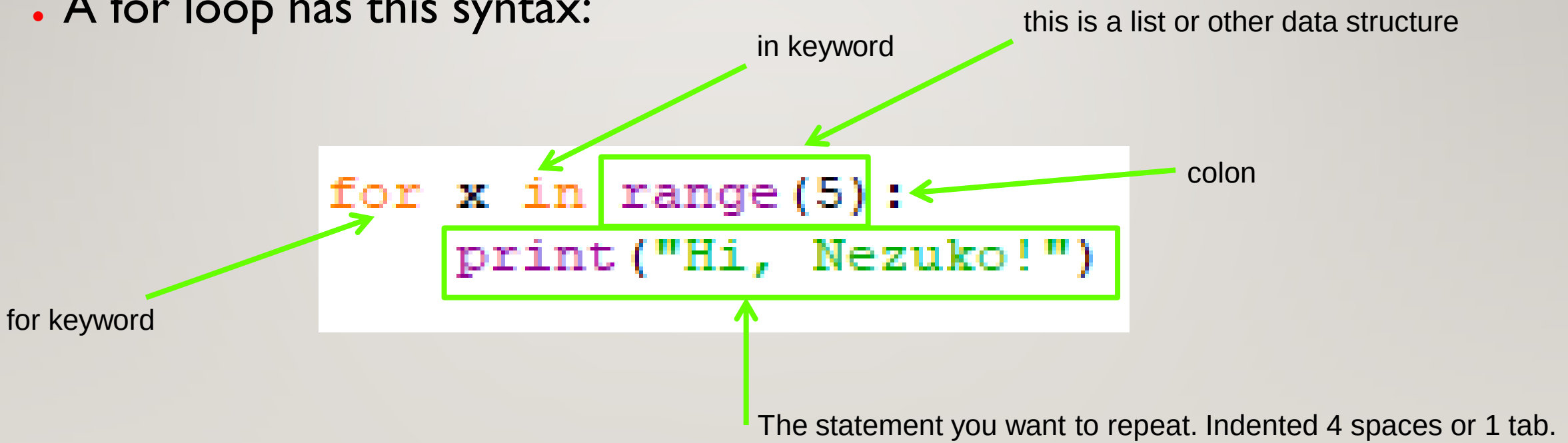
```
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
print("Hi, Nezuko!")
```

```
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:



SPACES VERSUS TABS

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

LOOPS AND LISTS

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

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

WAYS TO CONTROL A FOR-LOOP

- Use range(a)
- Use range(a,b)
- Use in + a list (or dictionary, or other data structure)

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
```

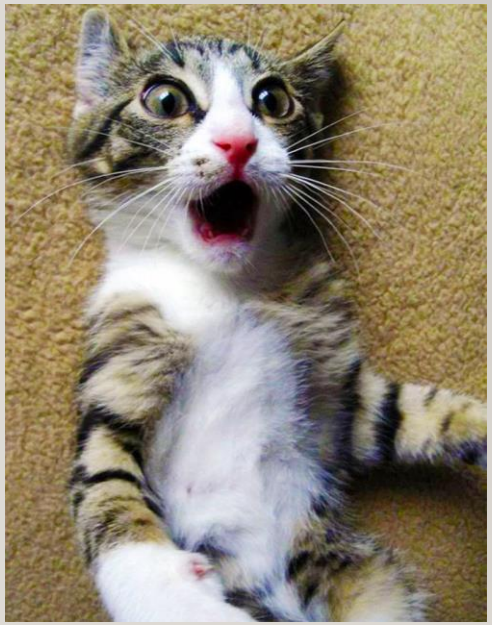
```
for x in range(10):
    a = 2 * x
    print(a)

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