Learn to Program With Python

Day 2: Building a Text Adventure

This course uses unofficial curriculum created for Girl Develop It! Ottawa by Gail Carmichael.

http://www.gailcarmichael.com

Today's class is inspired by content from:

Adventures in Raspberry Pi
by Carrie Anne Philbin

The Python Game Book – Adventure Game
(http://thepythongamebook.com/en:resources:people:jens_horst:part1step002)

How To Make Your Own Text Adventure On A Computer
(http://www.bluzeandmuse.com/final_site/how_to.html)

Access these slides online:

http://gailcarmichael.com/learn-python

TEXT ADVENTURES

Zork

The Game:
http://www.web-adventures.org/cgi-bin/webfrotz?s=ZorkDungeon

Command list:
http://zork.wikia.com/wiki/Command_List

BASIC USER INPUT

Strings

Text is stored in a program as a “string”, which is anything inside quotes.

```
"I'm a string!"
'Me too!'

'I'll cause an error'
"So will I'
"Don't forget "me"!"
```

Strings

Text is stored in a program as a “string”, which is anything inside quotes.

```
"""Triple quotes means
    all my white  space and new lines
-
  no matter how weird
-
will
  be
    kept."""
```

Input

Python 2:

```
answer = raw_input("What did you eat today? ")
print(answer)
```

Python 3:

```
answer = input("What did you eat today? ")
print(answer)
```

Printing Answers Nicely

```
answer = input("What did you eat today? ")
print("Today I ate " + answer)
```

Adding a Dramatic Pause

```
import time

answer = input("What did you eat today? ")

time.sleep(1)

print("Today I ate " + answer)
```

Adventure Example

```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

Adventure Example

Just like we needed the turtle module to draw, we need the time module so we can sleep

```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

Adventure Example

Print some introductory text

```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

Adventure Example

```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

Cause the program to sleep for the number of seconds provided as a parameter

Adventure Example

```
import time

print("You have entere                              me.")
print("You need to arm

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

Ask the user to choose a 'weapon' (notice how the string is written on multiple lines)

Adventure Example

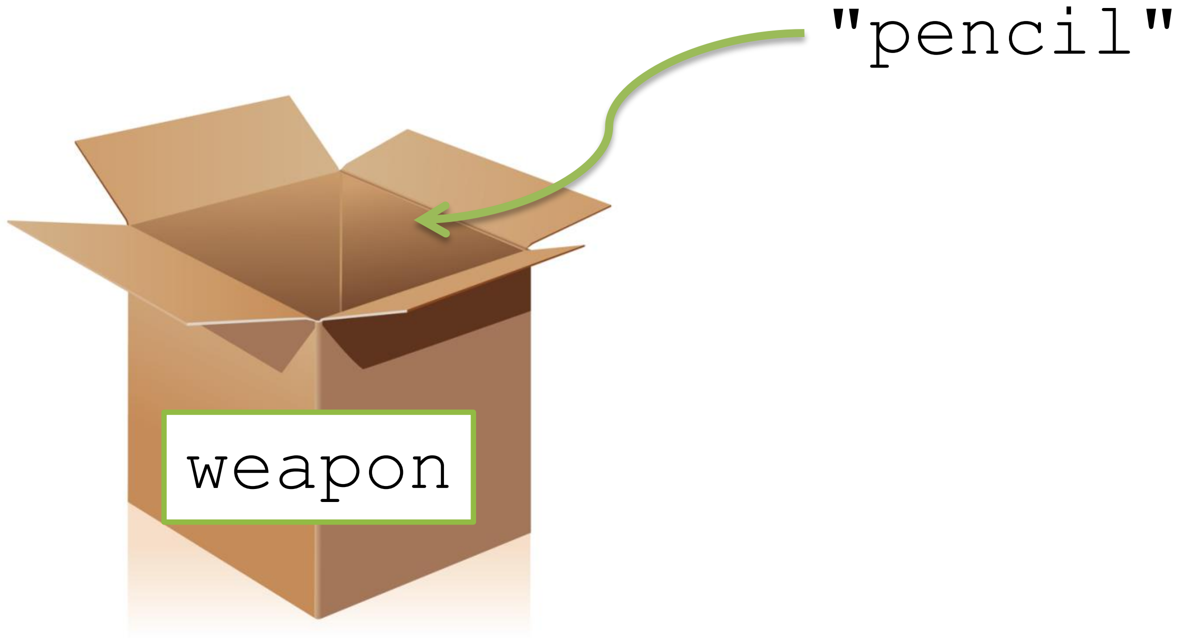
```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

This means add a new line at the end of the string

Adventure Example

```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

           n your backpack for " + weapon)




           not find " + weapon)
```

The user's response is saved in the box labelled weapon (i.e. the weapon variable)

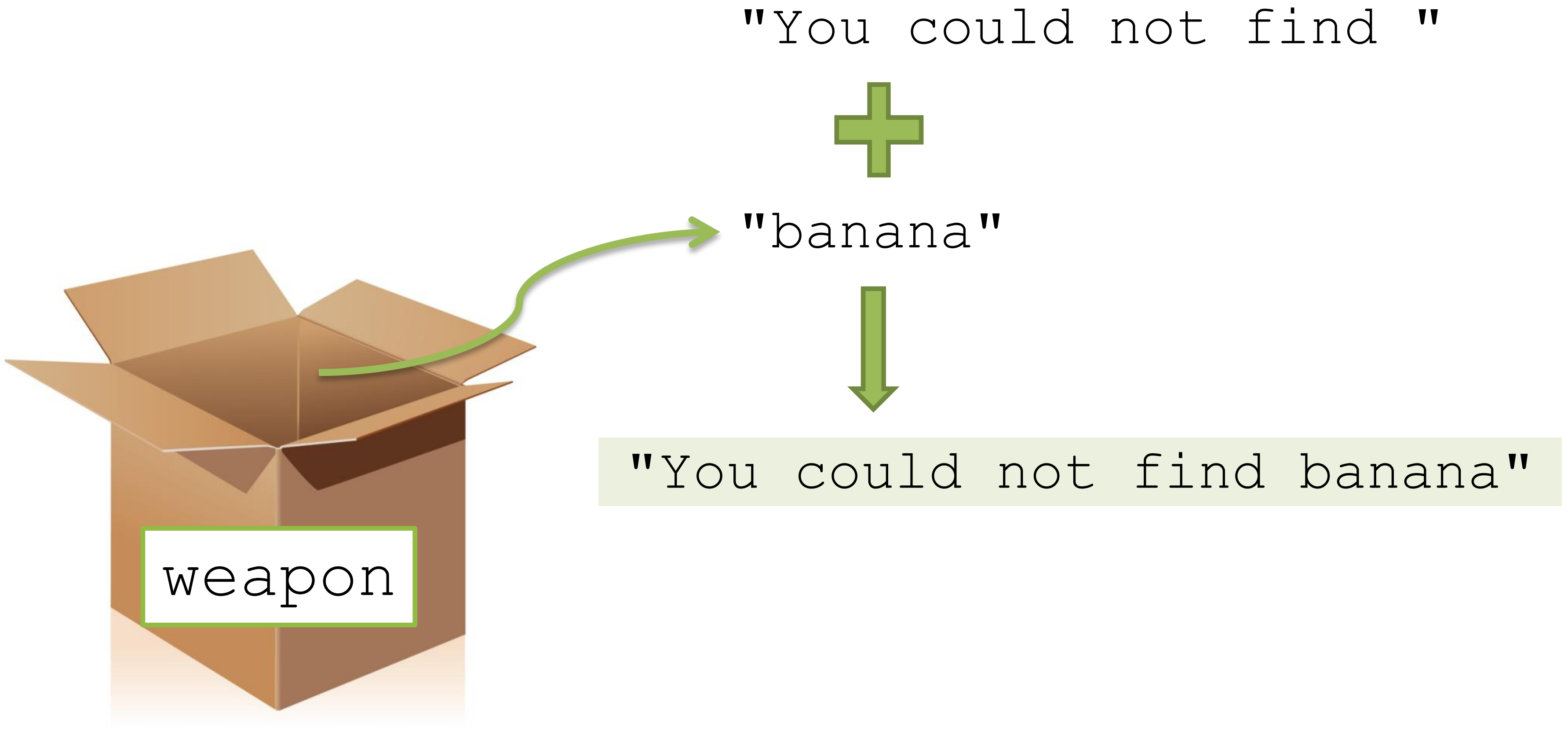


```
weapon = input("…")
```

Adventure Example

```
import time

print("You have entered the classroom for the first time.")
print("You need to arm yourself for learning.")

time.sleep(1)

weapon = input("What weapon of mass learning will you "
               "choose?\n")

print("You look in your backpack for " + weapon)

time.sleep(2)

print("You could not find " + weapon)
```

The user's input is stored as a string, too, so we can stick the two strings together to print them nicely.


"You could not find "
"banana"
"You could not find banana"
weapon

LISTS

Where did we use a list before?

Where did we use a list before?

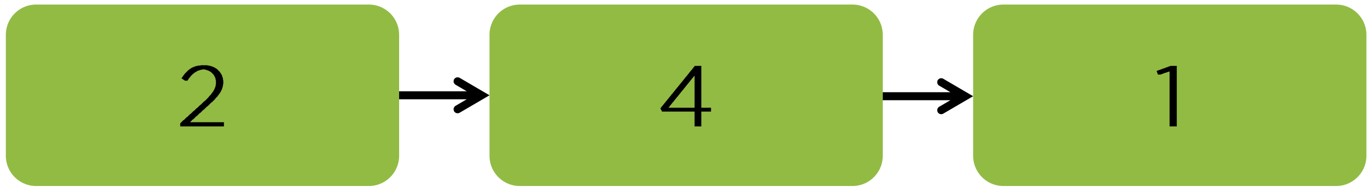
```
for sideNum in [1, 2, 3, 4, 5]:
    alex.forward(100)
    alex.left(72)
```

Where did we use a list before?

```
for sideNum in [1, 2, 3, 4, 5]:
    alex.forward(100)
    alex.left(72)
```

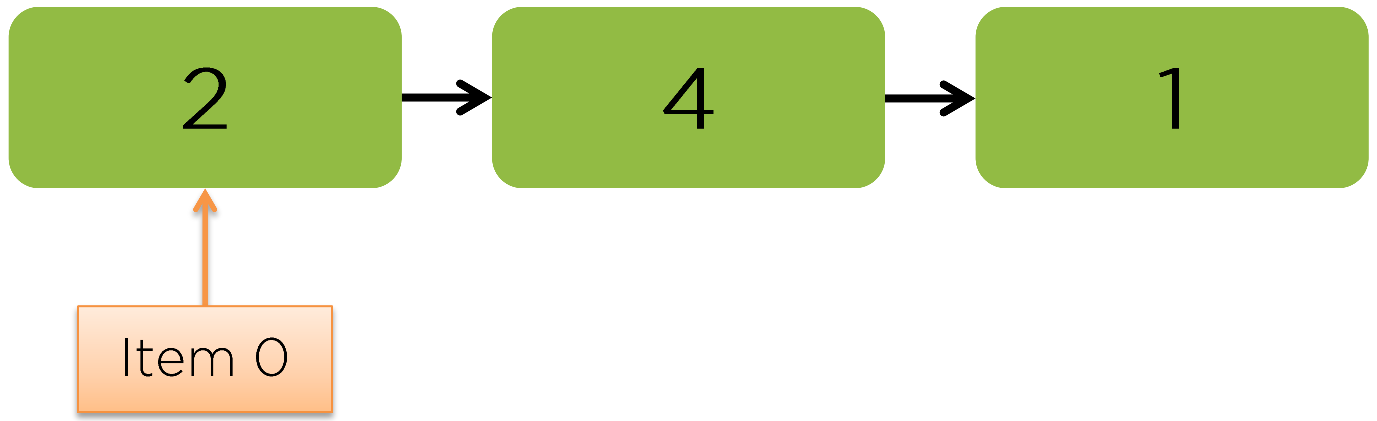
```
for aColor in ["red", "blue", "yellow",
               "green", "purple"]:
    alex.color(aColor)
    alex.forward(100)
    alex.left(72)
```

Creating a List



```
myList = [2, 4, 1]
```

Using a List

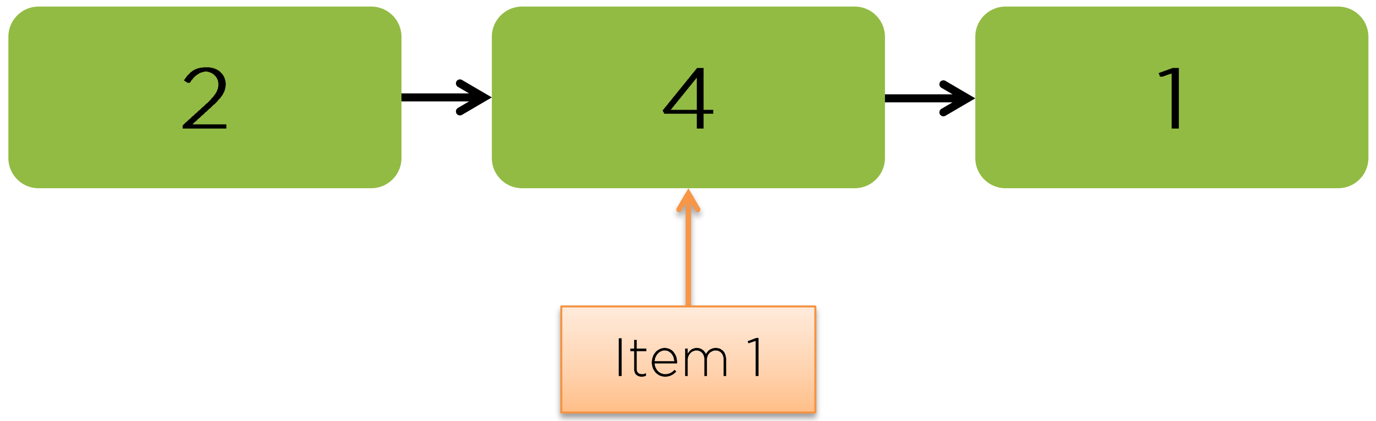


```
print(myList[0])
```

↑

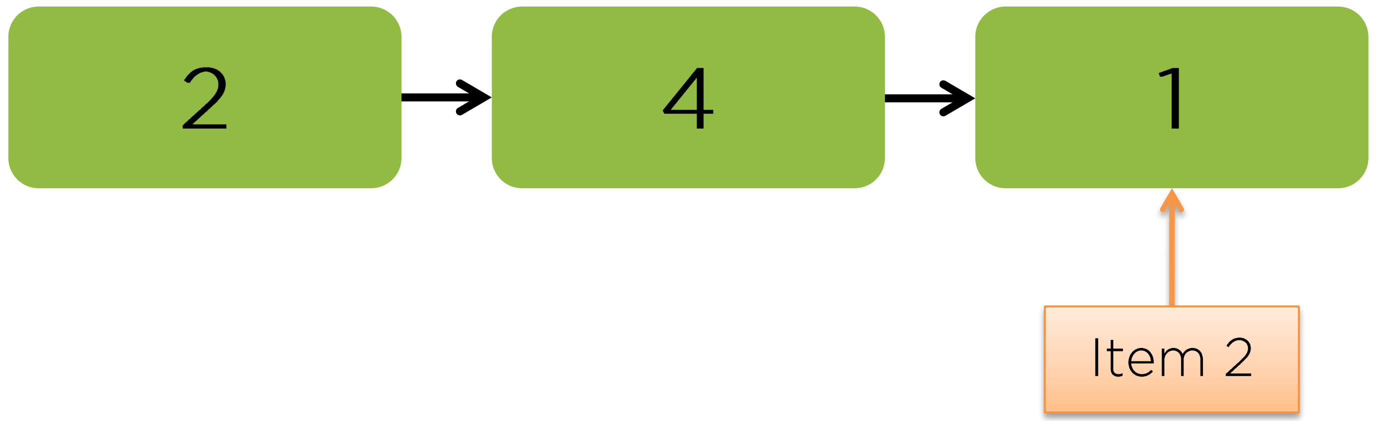
known as the *index*

Using a List



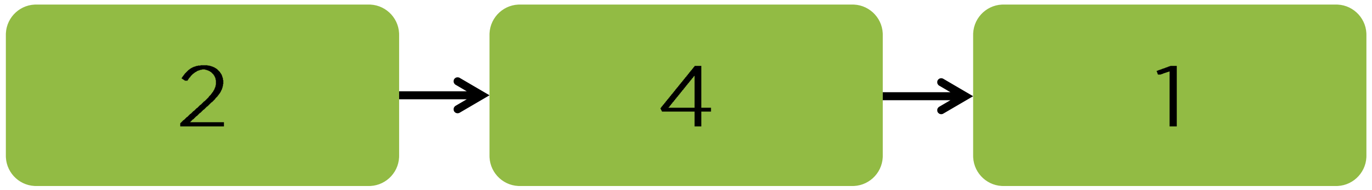
```
print(myList[1])
```

Using a List



```
print(myList[2])
```

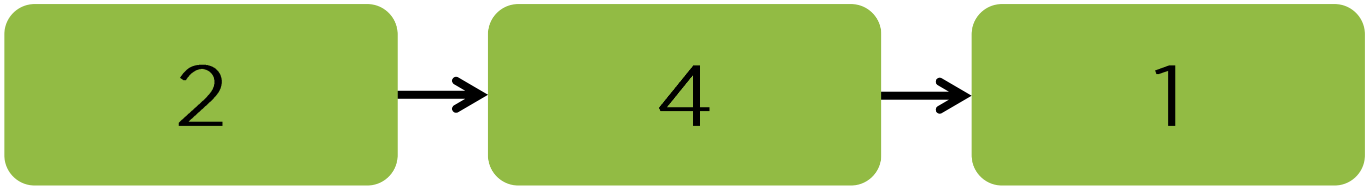
Using a List



```
print(len(myList))
```

Prints 3

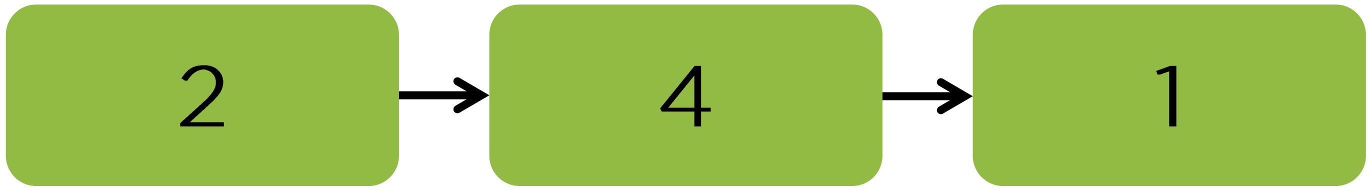
Using a List



```
print(myList[len(myList)-1])
```

Prints ?

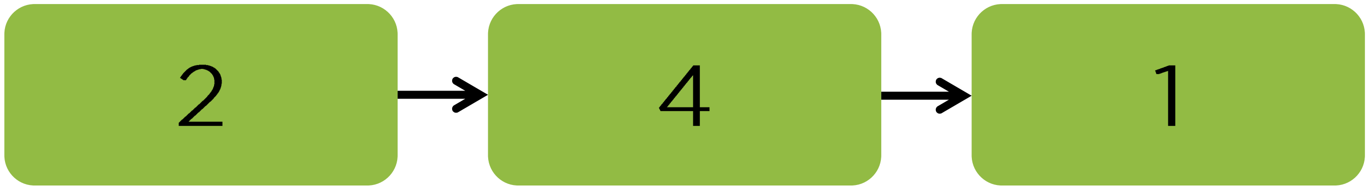
Using a List



```
print(myList[len(myList)-1])
```

Prints 1

Using a List



```
print(myList[-1])
```

Watch out for negative numbers as they may not do what you expect.

The above prints `1`.

Adventure Example

```
import random
import time
...
```

Add a new import at the top of the file so we can use the `random` module.

Adventure Example

```
inventory = ["pen", "pencil", "tablet",
             "textbook", "banana"]
```

Below the imports, create a list called inventory and add some "weapons of mass learning" to it.

Adventure Example

```
print("You pick something randomly from "
      "your backpack instead: ")
time.sleep(3)

print(random.choice(inventory))
```

Finally, at the end of the program, tell the user they chose something randomly from their inventory, then actually pick something to print.

Adventure Example

What could happen if you type something that is actually in your inventory when asked for a "weapon"?

CHECKING USER INPUT

Checking whether the user's choice is already in the list

`weapon` is in the **`inventory`** list

Yes:
Tell the user they found the item.

No:
Tell the user they picked something random instead.

Checking whether the user's choice is already in the list

```
if weapon in inventory:
    print("You found " + weapon + " in your backpack.")
else:
    print("You did not find " + weapon + " in your "
          "backpack.")
    print("You pick something randomly from your "
          "backpack instead: ")
    print(random.choice(inventory))
```

Checking whether the user's choice is already in the list

```
if weapon in inventory:
    print("You found " +                    backpack.")
else:
    print("You did not find " + weapon + " in your "
          "backpack.")
    print("You pick something randomly from your "
          "backpack instead: ")
    print(random.choice(inventory))
```

Checks whether the contents of the `weapon` box (variable) are in the list `inventory` list

Checking whether the user's choice is already in the list

```
if weapon in inventory:
    print("You found " +
                                   backpack.")
else:
    print("You did not find " + weapon + " in your "
          "backpack.")
    print("You pick something randomly from your "
          "backpack instead: ")
    print(random.choice(inventory))
```

This statement is either true or false. If it is true...

Checking whether the user's choice is already in the list

```
if weapon in inventory:
    print("You found " + weapon + " in your backpack.")
else:
    print("You did not find " +                             "
          "backpack.")
    print("You pick something randomly from your "
          "backpack instead: ")
    print(random.choice(inventory))
```

...this will be printed.

Checking whether the user's choice is already in the list

If it is false...

```
if weapon in inventory:
    print("You found " +                your backpack.")
else:
    print("You did not find " + weapon + " in your "
          "backpack.")
    print("You pick something randomly from your "
          "backpack instead: ")
    print(random.choice(inventory))
```

Checking whether the user's choice is already in the list

```
if weapon in inventory:
    print("You found " + weapon + " in your backpack.")
else:
    print("You did not find " + weapon + " in your "
          "backpack.")
    print("You pick something randomly from your "
          "backpack instead: ")
    print(random.choice(inventory))
```

...this code will run.

Taking Action Based on Input

An adventure game needs a set of commands the user will interact with.

How can we start implementing our own?

Taking Action Based on Input

```
inventory = ["pen", "pencil", "tablet",
              "textbook", "banana"]

print("You are sitting in your chair. "
      "What would you like to do next?")

answer = input("> ")
```

Taking Action Based on Input

```
inventory = ["pen", "pencil", "tablet",
              "textbook", "banana"]
```

```
print("You are sitting in your chair. "
      "What would you like to do next?")

answer = input("> ")
```

This is a nicer way to ask for input: set things up, then give the user a prompt

Taking Action Based on Input

```
if answer == "ask question":
    print("You decide to ask the teacher "
          "a question.")

elif answer == "check inventory":
    print("Your inventory: " +
          ", ".join(inventory))

else:
    print("I don't know that command.")
```

Taking Action Based on Input

```
if answer == "ask question":
    print("You decide to ask the teacher "
          "a question.")

elif a                           ventory":
    pr                           : " +
          ", ".join(inventory))

else:
    print("I don't know that command.")
```

We have a new structure here that includes the keyword `elif`

```
if <something true or false>:
    <code>
else:
    if <something true or false>:
        <code>
    else:
        if <something true or false>:
            <code>
        else:
           <code>
```

We could chain together a bunch of if statements together...

```
if <something true or false>:
    <code>
elif <something true or false>:
    <code>
elif <something true or false>:
    <code>
else:
    <code>
```

...or we can use `elif`, which is short for "else if"

```
if 4 < 3:
    print(1)
elif 4 < 5:
    print(2)
elif 4 < 6:
    print(3)
else:
    print(4)
```

Important: In a chain of an `if`, some number of `elif`(s), and an optional `else`, only the first condition that is True will be used, or else if nothing before it was True.

Taking Action Based on Input

```
if answer == "ask question":
    print("You decide to ask the teacher "
          "a question.")

elif answer == "check inventory":
    print("Your inventory: " +
          ", ".join(inventory))

else:
    print("I don't know that command.")
```

Taking Action Based on Input

```
if answer == "ask question":
    print("You decide to ask the teacher "
          "a question.")

elif answer == "check inventory":
    print("Your inventory: " +
          ", ".join(inventory))

else:
    pri                    at command.")
```

Create a temporary string that will be used as a separator when printing the list...

Taking Action Based on Input

```
if answer == "ask question":
    print("You decide to ask the teacher "
          "a question.")

elif answer == "check inventory":
    print("Your inventory: " +
          ", ".join(inventory))

else:
    print("I                    mand.")
```

...then use `join` to take all the items in the given list and stick them together into a single string using the separator

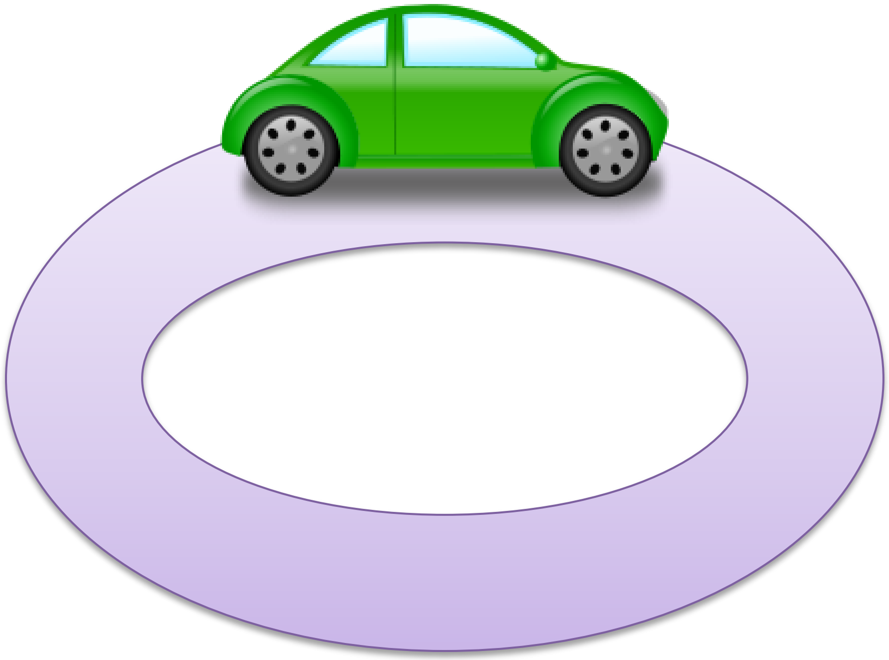
Taking Action Based on Input

```
if answer == "ask question":
    print("You decide to ask the teacher "
          "a question.")

elif answer == "check inventory":
    print("Your in
          ", ".joi

else:
    print("I don't know that command.")
```

What if we don't want to leave it at that? What if we want to keep asking until we get good input?

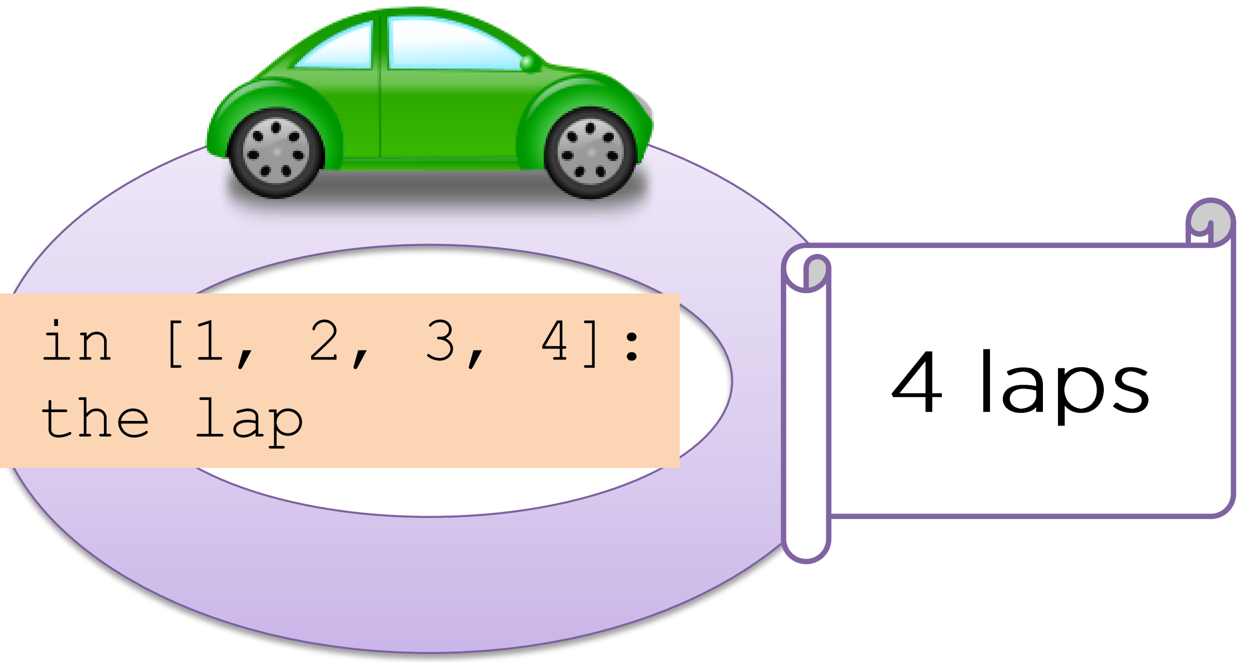
LOOPS AND USER INPUT

Loops



Drive the same track multiple times

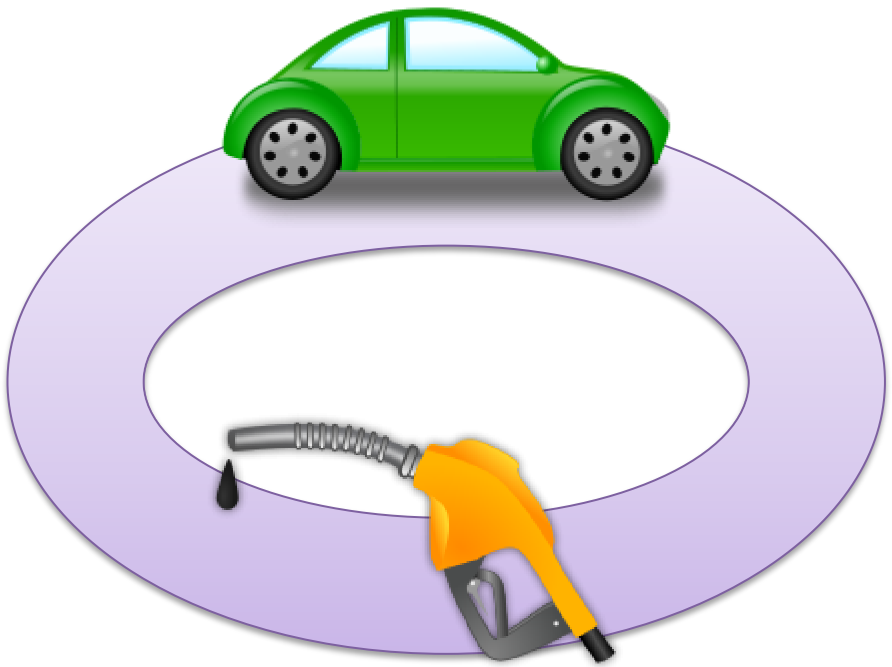
for loop



```
for lapNum in [1, 2, 3, 4]:
    # drive the lap
```

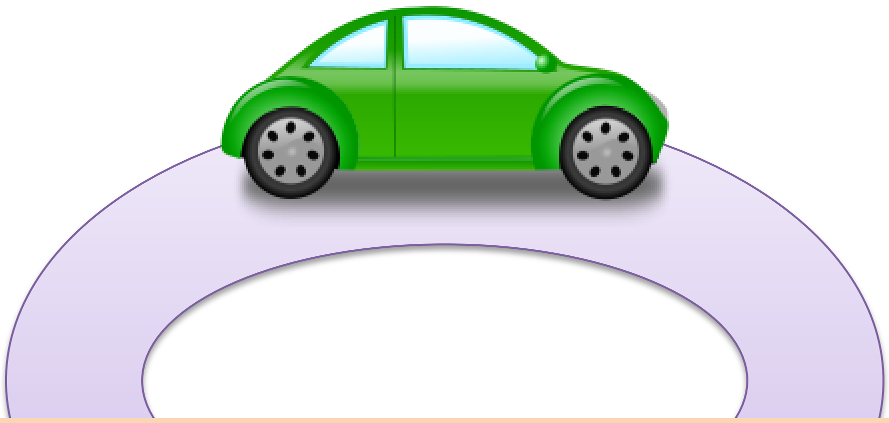
Drive the same track exactly four times

while loop



Keep driving the track until we run out of gas (but always go back to the starting point before stopping).

while loop



```
answer = input("> ")
while answer != "1":
    print("Try again")
    answer = input("> ")
```

Keep driving the track until we run out of gas (but always go back to the starting point before stopping).

```
number = 5
while number < 10:
      number = number + 1
      print(number)
```

Important: We always drive the whole track (i.e. run all the code inside the loop) before coming back to the beginning. *Then* we check whether the loop's statement is true.

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == False:
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting

answerIsGood = False

while answerIsGood ==
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

Store a variable called `answerIsGood` to keep track of when we have to ask for input again

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting

answerIsGood = False

while answerIsGood ==
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

Start with the answer being bad so we have to loop at least once

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == False:
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

We want to keep driving the track, asking for new input, while the answer is not good

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == Fal
    answerIsGood = True

    print("What would you
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

Assume the answer will be good until it's proven otherwise

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == False:
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

Each lap around the track should start by asking the user what they want to do.

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == False:
    answerIsGood = True

    print("What would you like
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inventory":
        print("Your inventory: " + ", ".join(inventory))
    else:
        answerIsGood = False
        print("I don't know that command.")
```

Once we have an answer, we can decide what to do with it

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == False:
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question'
        print("You decide to as
    elif answer == "check inver
        print("Your inventory:
    else:
        answerIsGood = False
        print("I don't know that command.")
```

The `else` is like a catch-all – if we haven't handled the answer yet, we know it couldn't have been good

Looping Until Input is Good

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]
print("You are sitting in your chair.")

answerIsGood = False

while answerIsGood == False:
    answerIsGood = True

    print("What would you like to do next?")
    answer = input("> ")

    if answer == "ask question":
        print("You decide to ask the teacher a question.")
    elif answer == "check inve
        print("Your inventory:
    else:
        answerIsGood = False
        print("I don't know that command.")
```

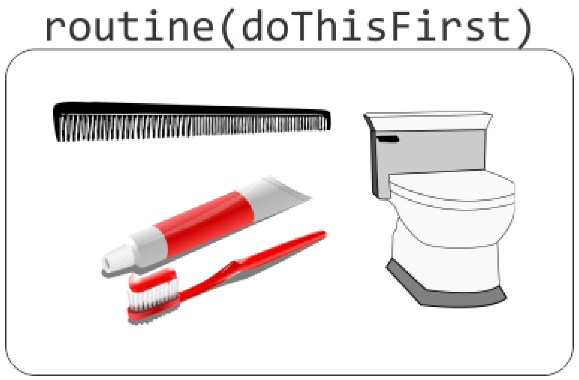
Setting this variable to `False` again ensures we drive around the track once more

Generalizing Getting Input

Suppose we have a set number of commands we want to give users access to. Can we separate getting good user input from actually taking action with it?

ROUTINES TO GET INPUT

Creating Customized Routines



```
def drawSquare(x, y):
    alex.penup()
    alex.goto(x, y)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)

drawSquare(50, 50)
drawSquare(200, 200)
```

Defining a User Input Routine

```
def getUserInput(allowedCommands):
    while True:
        print("What do you want to do?")
        answer = input("> ")
        if answer in allowedCommands:
            return answer
        else:
            print("I don't know that command.")
```

Defining a User Input Routine

```
def getUserInput(allowedCommands):
```

We are defining a routine called `getUserInput`

```
                              want to do?")
                        ")
        if answer in allowedCommands:
            return answer
        else:
            print("I don't know that command.")
```

Defining a User Input Routine

```
def getUserInput(allowedCommands):
    while T
        pri
        ans
        if
            return answer
        else:
            print("I don't know that command.")
```

We are setting up a parameter that should be a list of commands the user can enter when asked

Defining a User Input Routine

```
def getUserInput
    while True:
        print("W
        answer = input("> ")
        if answer in allowedCommands:
            return answer
        else:
            print("I don't know that command.")
```

This seems to mean we drive around the track... forever!!

Defining a User Input Routine

```
def getUserInput(allowedCommands):
    while True:
        print("What do you want to do?")
        answer = input("> ")
        if answer in allowe
            return answer
        else:
            print("I don't
```

Although a routine doesn't have to have a result, it can – this routine "returns" the user's answer as a result

Defining a User Input Routine

```
def getUserInput(allowedCommands):
    while True:
        print("What do you want to do?")
        answer = input("> ")
        if answer in allowe
            return answer
        else:
            print("I don't
```

Returning a result causes the routine to exit immediately; therefore, the loop also ends.

Generalizing the Question Asked

```
def getUserInput(question, allowedCommands):
    while True:
        print( question )
        answer = input("> ")
        if answer in allowedCommands:
            return answer
        else:
            print("I don't know that command.")
```

Using the Input Routine

```
command = getUserInput("What do you want to do?",
                       ["ask question", "check inventory"])

if command == "ask question":
    print("You decide to ask the teacher a question.")
elif command == "check inventory":
    print("Your inventory: " + ", ".join(inventory))
```

Using the Input Routine

```
command = getUserInput("What do you want to do?",
                          sk question", "check inventory"])

                          n":
                          sk the teacher a question.")
elif command == "check inventory":
    print("Your inventory: " + ", ".join(inventory))
```

Anything a routine returns can be saved into a variable

Using the Input Routine

```
command = getUserInput("What do you want to do?",
                       ["ask question", "check inventory"])

if command == "ask question":
    print("You decide to ask the teacher a question.")
elif command == "check inventory":
    print("Your inventory: " + ", ".join(inventory))
```

What is the advantage of creating and using the routine?

ROUTINES TO HANDLE PLACES TO GO

Adding Places to Go

We know how to add commands to our game, but what about places to go?

A good technique is to write a routine (aka function) for each area the player can visit.

Adding Places to Go

```
def handleClassroom():
    print("You walk into the classroom just as class begins.")
    print("You can ask the teacher about birds or pencils.")

    answer = getUserInput("Which would you like to ask about?",
                          ["birds", "pencils"])

    if answer == "birds":
        print("The teacher explains how birds are different"
              " from mammals.")

    elif answer == "pencils":
        print("The teacher sends you to the principal's office"
              " for distracting the class...again.")
```

Going Places

```
inventory = ["pen", "pencil", "tablet", "textbook", "banana"]

command = getUserInput("What do you want to do?",
                       ["go to class", "check inventory"])

if command == "go to class":
    handleClassroom()
elif command == "check inventory":
    print("Your inventory: " + ", ".join(inventory))
```

PUTTING IT ALL TOGETHER

Game Loop

We can look at our text adventure game as some valid path through all of our locations until we reach the "end."

```
while we are not at the end:
    call the routine for the current location
```

Locations Should Return New Locations

```
def handleClassroom():
    print("You walk into the classroom just as class begins.")
    print("You can ask the teacher about birds or pencils.")

    answer = getUserInput("Which would you like to ask about?",
                          ["birds", "pencils"])

    if answer == "birds":
        print("The teacher explains how birds are different"
              " from mammals.")
        return "classroom"

    elif answer == "pencils":
        print("The teacher sends you to the principal's office"
              " for distracting the class...again.")
        return "principal"
```

The Game Loop Drives Going Through Locations

```
inventory = ["pen", "pencil", "tablet", "textbook",
             "banana"]
location = "classroom"

while location != "end":
    if location == "classroom":
        location = handleClassroom()
    elif location == "principal":
        print("To do: make principal")
        location = "end"

print("The end!")
```

The Game Loop Drives Going Through Locations

```
inventory = ["pen", "pencil", "tablet", "textbook",
             "banana"]
location = "classroom"

while location != "end":
    if location == "class
        location = handleClassroom()
    elif location == "principal":
        print("To do: make principal")
        location = "end"

print("The end!")
```

Keep going until the last location is reached

The Game Loop Drives Going Through Locations

```
inventory = ["pen", "pencil", "tablet", "textbook",
             "banana"]
location = "classroom"

while location != "end":
    if location == "classroom":
        location = handleClassroom()
    elif location == "principal":
        print("To do: make principal")
        location = "end"

print("The end!")
```

Check the current location…

The Game Loop Drives Going Through Locations

```
inventory = ["pen", "pencil", "tablet", "textbook",
             "banana"]
location = "classroom"

while location != "end":
    if location == "classroom":
        location = handleClassroom()
    elif location == "principal":
        print("To do: make principal")
        location = "end"

print("The end!")
```

…and call the appropriate routine.

The Game Loop Drives Going Through Locations

```
inventory = ["pen", "pencil", "tablet", "textbook",
             "banana"]
location = "classroom"

while location != "end":
    if location == "classroom":
        location = handleClassroom()
    el                            l":
                                  ncipal")

print("The end!")
```

Save the next location for use when we drive around the track next time.

Updating getUserInput

```
def getUserInput(question, allowedCommands):
    while True:
        print(question)
        answer = input("> ")
        if answer in allowedCommands:
            return answer
        elif answer == "check inventory":
            print("Your inventory: "
                  + ", ".join(inventory))
        else:
            print("I don't know that command.")
```

Updating getUserInput

```
def getUserInput(question, allowedCommands):
    while True:
        print(question)
        answer = input("> ")
        if answer in allowedCommands:
            return answer
        elif answer == "check inventory":
            print("Your inventory: "
                  + ", ".join(inventory))
                                                      nd.")
```

Now that we aren't asking for commands outside location routines, we need somewhere to allow users to check their inventory (and any other general commands)