An artificially intelligent virtual gamer created by computer scientists at The University of Texas at Austin has won the BotPrize by convincing a panel of judges that it was more human-like than half the humans it competed against.

The bots face off in a tournament against one another and about an equal number of humans, with each player trying to score points by eliminating its opponents. Each player also has a "judging gun" in addition to its usual complement of weapons. That gun is used to tag opponents as human or bot.

The winning bots both achieved a humanness rating of 52 percent. Human players received an average humanness rating of only 40 percent.

http://www.sciencedaily.com/releases/2012/09/120926133235.htm
• Questions?
• Class: a prototype for an object
• Instance/Object: a unique thing, containing data and methods, defined by its class
• Constructor: a special method in a class that creates an instance of a class --- it must have the same name as the class
Psudocode: Simple paint program

- **DrawButtons( win )**
  - Called during program setup
  - Draws buttons to screen
  - No return value

- **ButtonClicked( x1, y1, x2, y2, p ) --- need if statement for**
  - Input = location of click
  - returns true iff button defined by (x1,y1), (x2,y2) has been clicked

- **ChangeColor( p, color )**
  - Called when user clicks at point p
  - Input = location of click, current color
  - Returns new color value if button was clicked, or old value if no color was clicked

- **HandleClick( p, pOld, color )**
  - Called when user clicks at point p (pOld was last place clicked)
  - Handle a button press, if it was clicked
  - Otherwise, draw a line from pOld to p using current color
Using Graphical Objects

• It’s possible for two different variables to refer to the same object
• Changes made to object through one variable will be visible to the other

```python
>>> leftEye = Circle(Point(80,50), 5)
>>> leftEye.setFill('yellow')
>>> leftEye.setOutline('red')
>>> rightEye = leftEye
>>> rightEye.move(20,0)
```

• Idea:
  1. create the left eye,
  2. copy that to the right eye ,
  3. which gets moved 20 units
Using Graphical Objects

- Assignment `rightEye = leftEye` makes `rightEye` and `leftEye` refer to the same circle.

- When two variables refer to the same object: called **aliasing**.
Aside

- Perceptual Aliasing
- Common problem in robotics
- The “robot kidnapping problem”
Using Graphical Objects
Using Graphical Objects

• There are two ways to get around this
• We could make two separate circles, one for each eye:

```python
>>> leftEye = Circle(Point(80, 50), 5)
>>> leftEye.setFill('yellow')
>>> leftEye.setOutline('red')
>>> rightEye = Circle(Point(100, 50), 5)
>>> rightEye.setFill('yellow')
>>> rightEye.setOutline('red')
```
Using Graphical Objects

• The graphics library has a better solution. Graphical objects have a clone method that will make a copy of the object

```python
>>> # Correct way to create two circles, using clone
>>> leftEye = Circle(Point(80, 50), 5)
>>> leftEye.setFill('yellow')
>>> leftEye.setOutline('red')
>>> rightEye = leftEye.clone() # rightEye is an exact copy of the left
>>> rightEye.move(20, 0)
```
Carol never wore her safety goggles.

She said, YOLO instead...

STOP YOLO-ing

WE DISAPPROVE OF YOUR POST

AND YOUR LYNX
The String Data Type

```python
>>> str1="Hello"
>>> str2='spam'
>>> print(str1, str2)
Hello spam
>>> type(str1)
<class 'str'>
>>> type(str2)
<class 'str'>
```
## The String Data Type

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+</code></td>
<td>Concatenation</td>
</tr>
<tr>
<td><code>*</code></td>
<td>Repetition</td>
</tr>
<tr>
<td><code>&lt;string&gt;[[]]</code></td>
<td>Indexing</td>
</tr>
<tr>
<td><code>&lt;string&gt;[:</code></td>
<td>Slicing</td>
</tr>
<tr>
<td><code>len(&lt;string&gt;)</code></td>
<td>Length</td>
</tr>
<tr>
<td><code>for &lt;var&gt; in &lt;string&gt;</code></td>
<td>Iteration through characters</td>
</tr>
</tbody>
</table>
The String Data Type

In a string of $n$ characters, the last character is at position $n-1$ since we start counting with 0.

We can index from the right side using negative indexes.

```python
>>> greet[-1]
'b'

>>> greet[-3]
'B'
```
The String Data Type

• Slicing:
  <string>[<start>:<end>]

• start and end should both be ints

• The slice contains the substring beginning at position start and runs up to but doesn’t include the position end
Create a program that accepts a first and last name, then creates a Lafayette User Id
Simple String Processing

• Usernames on a computer system
  – First initial, first seven characters of last name

```python
# get user’s first and last names
first = input("Please enter your first name (all lowercase): ")
last = input("Please enter your last name (all lowercase): ")

# concatenate first initial with 7 chars of last name
uname = first[0] + last[:7]
```
Simple String Processing

>>> 
Please enter your first name (all lowercase): john
Please enter your last name (all lowercase): doe
uname = jdoe

>>> 
Please enter your first name (all lowercase): donna
Please enter your last name (all lowercase): rostenkowski
uname = drostenk
Simple String Processing

• Another use – converting an int that stands for the month into the three letter abbreviation for that month

• Store all the names in one big string:
  “JanFebMarAprMayJunJulAugSepOctNovDec”

• Use the month number as an index for slicing this string:
  monthAbbrev = months[pos:pos+3]
Simple String Processing

# month.py
# A program to print the abbreviation of a month, given its number

def main():
    # months is used as a lookup table
    months = "JanFebMarAprMayJunJulAugSepOctNovDec"

    n = eval(input("Enter a month number (1-12): "))

    # compute starting position of month n in months
    pos = (n-1) * 3

    # Grab the appropriate slice from months
    monthAbbrev = months[pos:pos+3]

    # print the result
    print ("The month abbreviation is", monthAbbrev + ").")

main()
Strings, Lists, and Sequences

- It turns out that strings are really a special kind of sequence, so these operations also apply to sequences!

```python
>>> [1,2] + [3,4]
[1, 2, 3, 4]

>>> [1,2]*3
[1, 2, 1, 2, 1, 2]

>>> grades = ['A', 'B', 'C', 'D', 'F']

>>> grades[0]
'A'

>>> grades[2:4]
['C', 'D']

>>> len(grades)
5
```
Strings, Lists, and Sequences

• Strings are always sequences of characters, but *lists* can be sequences of arbitrary values.
• Lists can have numbers, strings, or both.

```python
myList = [1, "Spam ", 4, "U"]
```
Strings, Lists, and Sequences

• We can use the idea of a list to make our previous month program even simpler

• We change the lookup table for months to a list:

```python
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```
Strings, Lists, and Sequences

• To get the months out of the sequence, do this:
  \[
  \text{monthAbbrev} = \text{months}[n-1]
  \]

Rather than this:
  \[
  \text{monthAbbrev} = \text{months}[\text{pos:pos+3}]
  \]
Strings, Lists, and Sequences

• This version of the program is easy to extend to print out the whole month name rather than an abbreviation

```python
months = ["January", "February", "March", "April", "May", "June",
         "July", "August", "September", "October", "November", "December"]
```
Strings, Lists, and Sequences

- Lists are *mutable*, meaning they can be changed.
- Strings can **not** be changed.

```python
>>> myList = [34, 26, 15, 10]
>>> myList[2]
15
>>> myList[2] = 0
>>> myList
[34, 26, 0, 10]

>>> myString = "Hello World"
>>> myString[2]
'l'
>>> myString[2] = "p"
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in -toplevel-
    myString[2] = "p"
TypeError: object doesn't support item assignment
```
Strings and Secret Codes

• Inside the computer, strings are represented as sequences of 1’s and 0’s, just like numbers
• A string is stored as a sequence of binary numbers, one number per character
• It doesn’t matter what value is assigned, as long as it’s done consistently
Strings and Secret Codes

• Originally, each manufacturer used their own encoding of numbers for characters

• ASCII system (American Standard Code for Information Interchange) uses 127 bit codes
  – http://www.asciitable.com/

• Python supports Unicode (100,000+ characters)
Strings and Secret Codes

• The `ord` function returns the numeric (ordinal) code of a single character.

• The `chr` function converts a numeric code to the corresponding character.

```python
>>> ord("A")
65
>>> ord("a")
97
>>> chr(97)
'a'
>>> chr(65)
'A'
```
Strings and Secret Codes

• Using `ord` and `char` we can convert a string into and out of numeric form

• The encoding algorithm is simple: get the message to encode
  for each character in the message:
    print the letter number of the character

• A for loop iterates over a sequence of objects, so the for loop looks like:
  for `ch` in `<string>`
Strings and Secret Codes

• One of these methods is *split*. This will split a string into substrings based on spaces

```python
>>> "Hello string methods!".split()
['Hello', 'string', 'methods!']
```
Strings and Secret Codes

• Split can be used on characters other than space, by supplying the character as a parameter.

```python
>>> "32,24,25,57".split(",")
["32", "24", "25", "57"]
>>> 
```
Strings and Secret Codes

- How can we convert a string containing digits into a number?
- Use `eval`

```python
>>> numStr = "500"
>>> eval(numStr)
500
>>> x = eval(input("Enter a number "))
Enter a number 3.14
>>> print x
3.14
>>> type (x)
<type 'float'>
```
Other String Methods

• There are a number of other string methods. Try them all! Great party trick!
  – `s.capitalize()` – Copy of `s` with only the first character capitalized
  – `s.title()` – Copy of `s`; first character of each word capitalized
  – `s.center(width)` – Center `s` in a field of given width
Other String Operations

- `s.count(sub)` – Count the number of occurrences of sub in s
- `s.find(sub)` – Find the first position where sub occurs in s
- `s.join(list)` – Concatenate list of strings into one large string using s as separator.
- `s.ljust(width)` – Like center, but s is left-justified
Other String Operations

– `s.lower()` – Copy of `s` in all lowercase letters
– `s.lstrip()` – Copy of `s` with leading whitespace removed
– `s.replace(oldsusb, newsub)` – Replace occurrences of `oldsusb` in `s` with `newsub`
– `s.rfind(sub)` – Like `find`, but returns the right-most position
– `s.rjust(width)` – Like `center`, but `s` is right-justified
Other String Operations

- `s.rstrip()` – Copy of s with trailing whitespace removed
- `s.split()` – Split s into a list of substrings
- `s.upper()` – Copy of s; all characters converted to uppercase
Input/Output as String Manipulation

- We now have a complete set of type conversion operations:

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>float(&lt;expr&gt;)</td>
<td>Convert expr to a floating point value</td>
</tr>
<tr>
<td>int(&lt;expr&gt;)</td>
<td>Convert expr to an integer value</td>
</tr>
<tr>
<td>str(&lt;expr&gt;)</td>
<td>Return a string representation of expr</td>
</tr>
<tr>
<td>eval(&lt;string&gt;)</td>
<td>Evaluate string as an expression</td>
</tr>
</tbody>
</table>
Input/Output as String Manipulation

- Often we will need to do some string operations to prepare our string data for output ("prettify it")
  - Yes, prettify is actually a word

- Let’s say we want to enter a date in the format "05/24/2003" and output "May 24, 2003." How could we do that?
1. Input the date in \texttt{mm/dd/yyyy} format (dateStr)
2. Split \texttt{dateStr} into month, day, and year strings
3. Convert the \texttt{month string} into a \texttt{month number}
4. Use the \texttt{month number} to lookup the \texttt{month name}
5. Create a new date string in the form “\texttt{Month Day, Year}”
6. Output the new date string
Input/Output as String Manipulation

dateStr = input("Enter a date (mm/dd/yyyy): ")

monthStr, dayStr, yearStr = dateStr.split("/")
Input/Output as String Manipulation

• Next step: Convert monthStr into a number
• You might think eval!
• We can use the \textit{int} function on monthStr to convert "05", for example, into the integer 5. \(\text{int("05") = 5}\)
Input/Output as String Manipulation

months = [“January”, “February”, …, “December”]
monthStr = months[int(monthStr) – 1]