Week 4
control flow

STAT 198/298 Fall 2020

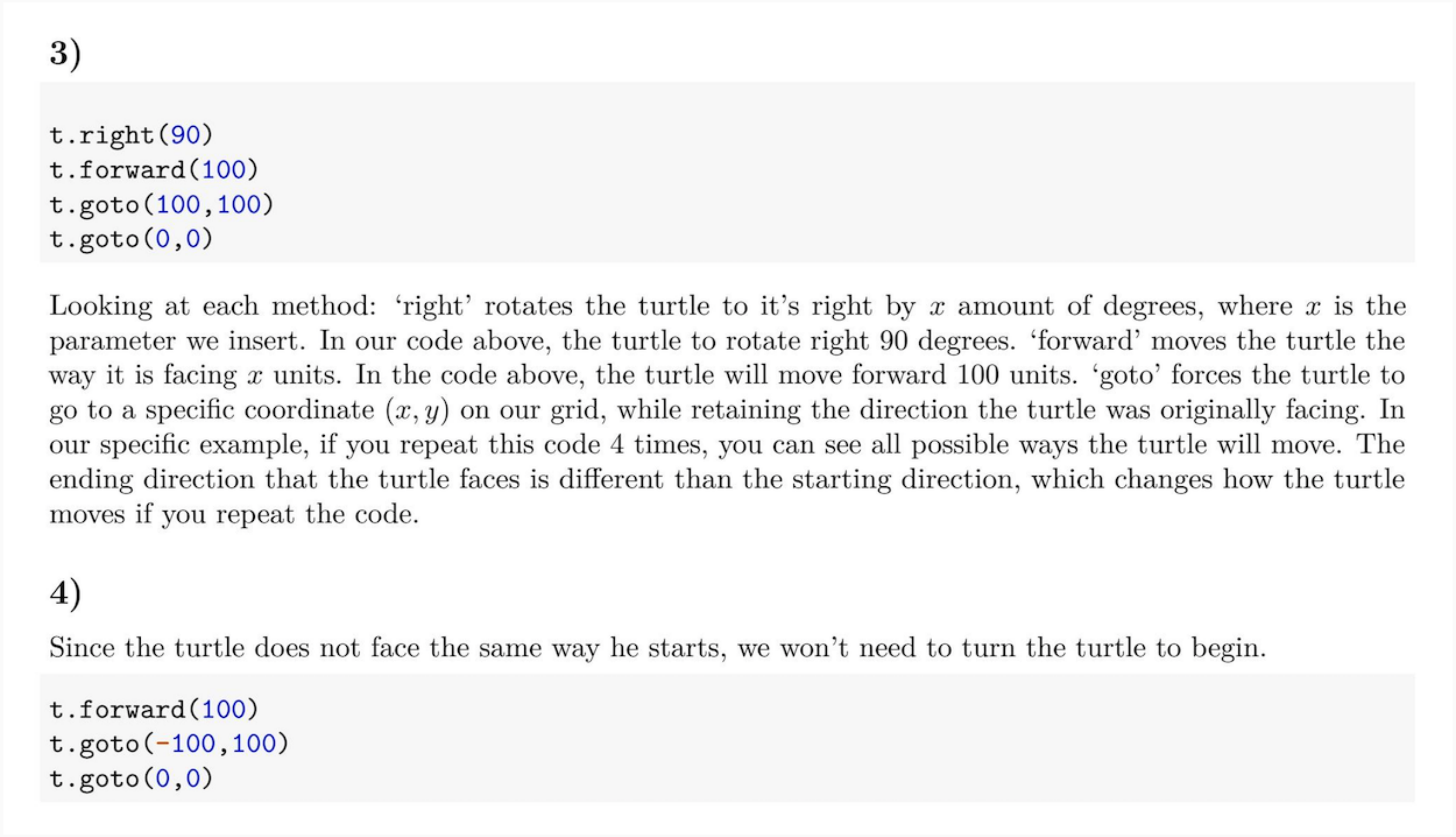
Send your phone here

Or send a browser to `slido.com`, event `#Z837`.

Agenda

1. Tips for assignments
2. Control Flow
    - if - else
    - for loops
3. What is a dataframe?

Writing Code, Writing English



3)

```
t.right(90)
t.forward(100)
t.goto(100,100)
t.goto(0,0)
```

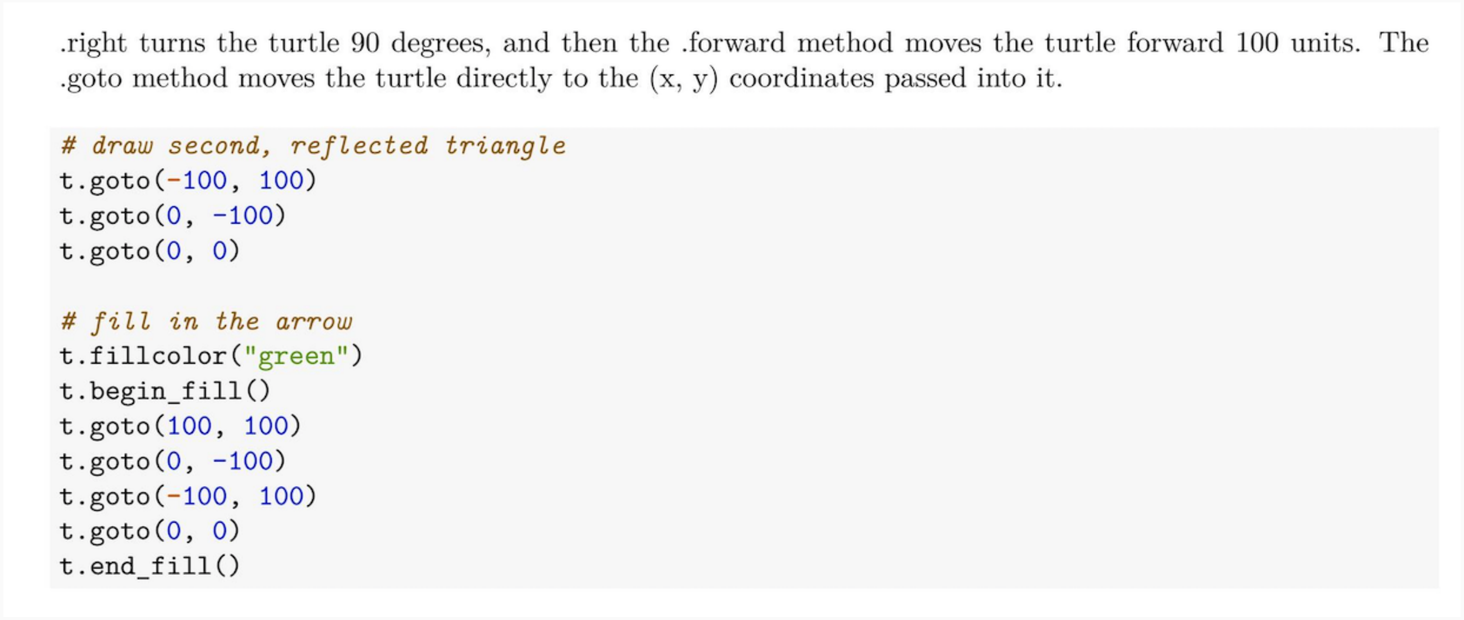
Looking at each method: ‘right’ rotates the turtle to it’s right by x amount of degrees, where x is the parameter we insert. In our code above, the turtle to rotate right 90 degrees. ‘forward’ moves the turtle the way it is facing x units. In the code above, the turtle will move forward 100 units. ‘goto’ forces the turtle to go to a specific coordinate (x, y) on our grid, while retaining the direction the turtle was originally facing. In our specific example, if you repeat this code 4 times, you can see all possible ways the turtle will move. The ending direction that the turtle faces is different than the starting direction, which changes how the turtle moves if you repeat the code.

4)

Since the turtle does not face the same way he starts, we won’t need to turn the turtle to begin.

```
t.forward(100)
t.goto(-100,100)
t.goto(0,0)
```

Markdown vs Comments



.right turns the turtle 90 degrees, and then the .forward method moves the turtle forward 100 units. The .goto method moves the turtle directly to the (x, y) coordinates passed into it.

```
# draw second, reflected triangle
t.goto(-100, 100)
t.goto(0, -100)
t.goto(0, 0)

# fill in the arrow
t.fillcolor("green")
t.begin_fill()
t.goto(100, 100)
t.goto(0, -100)
t.goto(-100, 100)
t.goto(0, 0)
t.end_fill()
```

`CODE`

Assignment tips

- Use `output: pdf_document`
- Suppress unneeded output with chunk options
    - `message = FALSE`: suppress package loading messages
    - `warning = FALSE`: suppress warnings
    - `echo = FALSE`: suppress your code showing up
    - `eval = FALSE`: suppress your output showing up
- Answers questions in full / thoughtful sentences
- Scan your pdf before you submit to be sure it looks right

Control Flow in Python

Elements of a script that redirect the flow of running commands

- If - then
- For loops
- While loops, etc.

if

```
if <condition>:
    <expression>
```

- keyword: `if`
- condition can be single or compound
- use the `:` to start expression
- expression can be multi-line but be sure to indent

if example

Example 1

What do you think will happen when this is run?

```
x = 15
if x == 0:
    print(x, "is zero.")
```

Nothing! (well, we now have a new object `x = 15`)

Example 2

```
x = 0
if x == 0:
    print(x, "is zero.")
```

```
## 0 is zero.
```

else

Example 3

```python
x = 15
if x == 0:
    print(x, "is zero.")
else:
    print(x, "is non-zero.")
```

```
## 15 is non-zero.
```

elif

Example 4

```
x = 15
if x == 0:
    print(x, "is zero.")
elif x > 0:
    print(x, "is positive.")
elif x < 0:
    print(x, "is negative")
else:
    print(x, "confuses me.")
```

```
## 15 is positive.
```

for loop

```
for <variable> in <iterator>:
    <expression>
```

- keyword: `for` and `in`
- variable can be any legal name that you'll refer to in the loop
- iterator is a generalized sequence
- start loop with `:`
- expression can be multi-line and must be indented

for loop example

Example 1

What do you think will happen when this is run?

```
for i in range(10):
    print(i, end = " ")
```

```
## 0 1 2 3 4 5 6 7 8 9
```

for loop example

Example 2

```
l = [1, 3, 99]
for i in range(3):
    print(l[i], end = " ")
```

```
## 1 3 99
```

Example 3

```
for i in [1, 3, 99]:
    print(i, end = " ")
```

```
## 1 3 99
```

- Loop on value or index - which is better?

`CODE`

Iterators

Python has a class of objects called *iterators* that behave like a list in terms of iteration but never actually create the full list.

POLL: What is a dataframe?

What is a dataframe?

`CODE`

What is a dataframe?

... it's a list!

A compound data structure that

- is *type heterogeneous*
- can contain elements of different sizes
- is subsetted using `[[]]`
- can also use key:value pairings (named elements)
- can be index with `$`
- can be nested

Whole smokes, lists are VERY flexible.

What's the difference between an R list and a Python dictionary?

R list vs Python Dictionary

Unlike the R dataframe, the Python dictionary:

- is unordered; can't index by position
- every element must be named

R dataframes combine elements of the Python list (index by position) and the Python dictionary (index by key)

`CODE`

What is a dataframe?

... it's a dataframe!

- a list with elements (vectors/lists) of equal length
- has specific methods
- can add `row.names`
- can be subsetted like a matrix `[row, column]` with indexing done by position or name

... so what's a tibble?

`CODE`

What is a tibble?

... it's an "opinionated dataframe" for the `tidyverse`.

- a dataframe with a refined print method
  - shows `dim()` and `typeof()`
  - limits number of rows printed
- doesn't change characters to factors
- can be created by coersion or by passing through `dplyr`

So what's a ggplot?

```
p1 <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point()
```

CODE