Tutorial03-lists+for

May 30, 2022

Due Saturday, June 9 at 11:59pm.

1 Make a new notebook

Answer all tutorial questions directly in a Jupyter notebook. In the Jupyter file browser, navigate to the folder in which you would like to save your tutorial. Select 'New/Python 3' to create a new notebook and edit its name from "`Untitled`" to "`tutorial03_{identikey}`", replacing "`{identikey}`" with your identikey (get rid of the squiggly brackets).

2 think+pair+code: `for` vs `while` loops (8 pts)

*In this **think+pair+code** section, you will need to work together with your neighbor(s)/zoommates. Introduce yourself, and work together. It can sometimes be way easier to solve coding problems if you talk through them with someone else!*

Please examine and run the following code snippet. Working together with your partner(s), use a "Code" cell to rewrite code that prints out *exactly the same outputs*, except using a `while` loop instead of the `for` loop. You may need to add some extra lines to the code.

[ ]:
```
for x in range(10):
    print(3*x)
```

3 think+type: accessing list elements (6 pts)

*In this **think+type** section, please type and test these commands. Think carefully about what you think each one will print, and feel free to play around!*

In Python we can access elements via their indices from the beginning of the list (starting at zero) or backwards from the end (starting at -1). Try these commands:

[ ]:
```
a = [2, 5, 3, 2, 11, 7] # create a list
```

[ ]:
```
a[0] # access first element of the list
```

[ ]:
```
a[1] # access second element of the list
```

```
[ ]: a[6] # access seventh(?) element of the list
```

Negative indicies count backwards from the end of the list.

```
[ ]: a[-1] # access the last element of the list
```

```
[ ]: a[-2] # access the second to last element of the list
```

4 think+code: accessing list elements (8 pts)

In this ***think+code****, please use the skills you just practice above to write your own simple code.*

Please write Python code in a “Code” cell that does the following tasks:

- Create a list that contains at least 5 different numbers and store it as a Python variable.
- Print out one element of the list. Use a positive integer as your index, so you’re counting forward from the start of the list.
- Print out that same element of the list, but this time use an index that counts backward from the end of the list.

5 think+type: modifying lists (11 pts)

In this ***think+type*** *section, please type and test these commands. Think carefully about what you think each one will print, and feel free to play around!*

We can modify the list as well, inserting new elements, and deleting elements we no longer want. ***Note that because we are modifying the list as we go, if you make a mistake (or run a cell more than once), you will need to re-run all the cells in this section to reinitialize your variables.***

```
[ ]: a = [2, 5, 3, 2, 11, 7] # create a list (same as before, just recreating it␣
      ↪here)
```

```
[ ]: a.append(10) # include "10" as a new element at the end of the list
     print(a)
```

```
[ ]: a.insert(-1, -2) # insert the element "-2" before index -1
     print(a)
```

```
[ ]: a.insert(2, 13) # insert the element "13" at index 2
     print(a)
```

```
[ ]: a.remove(2) # remove the *first* occurrence of "2"
     print(a)
```

```
[ ]: b = a.pop() # remove and return the last element
     print(b)
     print(a) # How did a change?
```

```
[ ]: c = a.pop(1)
     print(c)
     print(a)
```

```
[ ]: d = [5, 17, 19]
     a.extend(d) # like append, but for multiple elements at once
     print(a)
```

```
[ ]: a.sort() # rearrange elements in ascending order
     print(a)
```

```
[ ]: len(a)  # print the length (number of elements)
```

```
[ ]: # lists can contain any data types
     a.insert(1, 'abc')
     print(a)
```

6 think+pair+code: making a list (8 pts)

*In this **think+pair+code** section, you will need to work together with your neighbor(s)/zoommates. Introduce yourself, and work together.*

Open a new Code cell, and include a comment on the first line that says "`# making lists with {name(s) of partner(s)}`", then do the following:

- Ask your partner(s) to share at least their three favorite planets (minor planets, exoplanets, and imaginary planets are allowed!)
- Define the variable `planets` to be a Python `list` containing these planets, each one as a separate string.
- Add one (or more) of your favorite planets to the list, using either the `append` or `extend` functions you saw above.
- Use the `planets.sort()` method to rearrange the order of the list elements.
- Print the list.

7 think+type: `for` loops (3 pts)

*In this **think+type** section, please type and test these commands. Think carefully about what you think each one will print, and feel free to play around!*

A simple `for` loop iterates over the elements of a Python list.

[ ]:
```
numbers = [0, 1, 2, 3] # make a list
for n in numbers: # loop over elements in the list
    print(n, n**2) # in each iteration, n is a different element in the list
```

Note that you don't have to tell python to increment `n` to the next element in the list. It does it automatically at the start of each loop.

The same result can be achieved using the Python built-in function `range`, which creates a `list` containing a sequence of numbers:

[ ]:
```
for n in range(4):
    print(n, n**2)
```

If only one argument is supplied, `range` will create a list from 0 up to (but not including!) the supplied number. But, it can also be used to create lists with any start, stop, or (integer) step-size you want as `range(start, stop, step)`. So, if we want to print all even numbers between 10 and 20:

[ ]:
```
for n in range(10, 22, 2):
    print(n)
```

8 think+code: the Nth closest star (8 pts)

In this ***think+code*** *section, you will need to write some code to do an astronomical calculation. As with all* ***think+code*** *sections, you should feel free to work together with your neighbors, even if it's not explicitly a* ***think+pair+code****!*

Imagine we wanted to know how far away the 100th, 1000th, or 10,000th closest star is. We could use some calculations inside a `for` loop to give us a sense of this. We'll start by first estimating the local density of stars in the Milky Way:

- There are 322 hydrogen-burning stars within 10 parsecs (pc) of the Sun.
- The volume of a 10 pc sphere is $\frac{4\pi}{3}(10\text{ pc})^3 = 4188\text{ pc}^3$.
- Therefore, the number density of stars in the local Milky Way is, therefore, about $\frac{322\text{ stars}}{4188\text{ pc}^3} = 0.077\frac{\text{stars}}{\text{pc}^3}$.

Let's imagine we make spherical shells around the Sun, and ask how many stars we would expect to fall inside that shell. Please write code that uses a loop to print an estimate for the average number of stars we would expect inside shells of increasing radii, from 2pc in radius out to (and including) 50pc in radius, in 2pc steps. You can assume the density of stars is approximately constant in our local region of the Milky Way. *Sanity Check: What should your calculation give at r=10pc?*

9 (bonus)+think+type: indices, elements, `enumerate`

You can skip this final bonus section if you're out of time.

If we want to loop over a list, sometimes we want to access "elements" in the list (the actual values), sometimes we want the indices (which indicate the location of the elements), and sometimes we want both. Let's loop through a list several different ways to demonstrate (note: once you type it in once, you may want to copy, paste, and edit the code):

[ ]:
```
darks = ['chocolate', 'matter', 'energy']
```

[ ]:
```
index = 0
print("Order of discovery:")
for element in darks:
    print(f" {index}, dark {element} was discovered")
    index += 1
```

[ ]:
```
print("Order of discovery:")
for index in range(len(darks)):
    element = darks[index]
    print(f" {index}, dark {element} was discovered")
```

[ ]:
```
print("Order of discovery:")
for index, element in enumerate(darks):
    print(f" {index}, dark {element} was discovered")
```

10 save+upload

Be sure to save the final version of your notebook. To submit, click "File|Download As >|HTML (html)" *inside* your jupyter notebook window to convert your notebook into an HTML web page file ***(Be sure not to chose "Save As…")***. This will save the html file to whatever local directory on your computer your web browser saves to. It should have a name like `tutorial03_{identikey}.html`. Please upload this HTML file as your homework submission to "Tutorial04" on Canvas.