Immutable types

strings, integers, floats, and booleans are **immutable types**

A **mutable** object can be changed after it is created, and an **immutable** object can't

Case Study: if you reassign an immutable type in a function, the change doesn’t last beyond the lifetime of the function (see **add3.py** for an example)

Lists

Lists are an example of a **mutable** data types

Lists support

+ concatenation

* repetition

len()

[] indexing

The same operations as strings! But one important difference: [] can be used to *both inspect and change* values in the list. For strings, [] can only inspect

append() # adds an element to the end of the list (changes the list!!!)

Example: addVal.py

You can change the values in a list using indexing, unlike strings

```
>>> word = "test"
>>> word[2] = "t"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> wordList = list(word)
>>> wordList[2] = "t"
>>> wordList
['t', 'e', 't', 't']
>>>
```

addVal.py — ~/CS21/cs21-devel/examples/inclass/w05 — Atom

File Edit View Selection Find Packages Help

add.py | add2.py | add3.py | tip.py | fancify.py | addVal.py

```python
"""
Write a program that adds 10 to every item in
a list of integers
"""

def main():
  values = list(range(10))
  print(values)

  # Example: code which iterates through every element in
  # the list and prints it on its own line
  for i in range(len(values)):
      print(values[i])

  # Add 10 to each item in values using a for loop
  for i in range(len(values)):
      values[i] = values[i] + 10

  # Print the result
  print(values)

main()
```

addVal.py 7:2 LF N UTF-8 Python 0 files