>>> inc = lambda x : x + 1
>>> p = [1, 3, 5, 7]
>>> [inc(num) for num in p]
[2, 4, 6, 8]
R is for …

- Random
  - .choice, .shuffle, .seed, .randint
- R
  - Programming language of choice in stats
- Refactoring
  - Better, but not different
• Sorting in Python and sorting in general
  • How to use .sort and sorted, differences
  • Key function – change how sorting works
  • Lambda – create anonymous functions

• Stable sorting
  • How to leverage when solving problems
  • Why Timsort is the sort-of-choice (! quicksort)

• Clever Hangman –
  • How does it work?
Go over Last WOTO from last time
WOTO last time – 1st question

Showing the list and the list sorted

```
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

What's the list returned by sorted(a, reverse=True)? *

- [yellow', 'violet', 'red', 'orange', 'indigo', 'green', 'blue']
- [violet', 'indigo', 'blue', 'green', 'yellow', 'orange', 'red']
WOTO last time – 1st question

Showing the list and the list sorted

```
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

What's the list returned by sorted(a, reverse=True)? *

- [yellow', 'violet', 'red', 'orange', 'indigo', 'green', 'blue']
- [violet', 'indigo', 'blue', 'green', 'yellow', 'orange', 'red]
WOTO last time – 2cd question

Showing the list and the list sorted

```python
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

What's the list returned by sorted(a, key=len)? *

- [red', 'blue', 'green', 'orange', 'yellow', 'indigo', 'violet']
- ['red', 'blue', 'orange', 'green', 'yellow', 'indigo', 'violet']
WOTO last time – 2cd question

Showing the list and the list sorted

```
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

What's the list returned by sorted(a, key=len)? *

- [red', 'blue', 'green', 'orange', 'yellow', 'indigo', 'violet']
- ['red', 'blue', 'orange', 'green', 'yellow', 'indigo', 'violet']
Showing the list and the list sorted

```python
In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']
```

The function max applied to a string returns the alphabetically greatest character in the string, so `max('indigo') == 'o'` and `max('yellow') == 'y'`. What's the list returned by `sorted(a, key=max)`? *

- [ ] ['indigo', 'orange', 'green', 'red', 'blue', 'violet', 'yellow']
- [ ] ['indigo', 'red', 'orange', 'green', 'blue', 'violet', 'yellow']
WOTO last time – 3rd question

Showing the list and the list sorted

In[14]: a = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
In[15]: sorted(a)
Out[15]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet', 'yellow']

The function max applied to a string returns the alphabetically greatest character in the string, so max('indigo') == 'o' and max('yellow') == 'y'. What's the list returned by sorted(a, key=max)? *

- ['indigo', 'orange', 'green', 'red', 'blue', 'violet', 'yellow']
- ['indigo', 'red', 'orange', 'green', 'blue', 'violet', 'yellow']
Turing Award 2019
Pat Hanrahan, Ed Catmull

- Pixar, RenderMan, Computer Generated Imagery

Catmull: You are not your idea, and if you identify too closely with your ideas, you will take offense when they are challenged.

Catmull: If you aren’t experiencing failure, then you are making a far worse mistake: You are being driven by the desire to avoid it.
Compsci 101
Stable Sorting, Lambda, Clever Hangman
Part 2 of 3

Susan Rodger
October 20, 2020

```python
>>> inc = lambda x : x + 1
>>> p = [1, 3, 5, 7]
>>> [inc(num) for num in p]
[2, 4, 6, 8]
```
Review: CSV and Sort for top artists

- Using two-sorts to get top artists

```python
print('Top 5 artists:
sortbycount = sorted([(a[1], a[0]) for a in counts.items()])
sortedArtists = [(a[1], a[0]) for a in sortbycount]
for artist in sortedArtists[-5:]:
    print(artist)
```

- Reverse tuples to sort
- Reverse tuples to print

Top 5 artists:
('John, Elton', 21)
('Who', 24)
('Rolling Stones', 36)
('Led Zeppelin', 38)
('Beatles', 51)
Top 5 Artists

• Instead of intermediary list, use `lambda`
• Instead of `[-5:]`, use `reverse=True`

```python
print("Top 5 artists:")
sortbycount = sorted([(a[1], a[0]) for a in counts.items()])
sortedArtists = [(a[1], a[0]) for a in sortbycount]
for artist in sortedArtists[-5:]:
    print(artist)

print("repeat it")
sortedArtists = sorted(counts.items(), key=lambda item: item[1], reverse=True)
for tup in sortedArtists[:5]:
    print(tup)
```

Output slightly different. Why?

repeat it
('Beatles', 51)
('Led Zeppelin', 38)
('Rolling Stones', 36)
('Who', 24)
('Eagles', 21)
The power of lambda

• We want to create a function "on-the-fly"
  • aka anonymous function
  • aka "throw-away" function

```python
In[7]: a
Out[7]: ['red', 'orange', 'green', 'blue', 'indigo', 'violet']
In[8]: sorted(a, key=lambda x: x.count("e"))
Out[8]: ['indigo', 'red', 'orange', 'blue', 'violet', 'green']
```

• Why 'indigo' first and 'green' last?
  • What about order of ties? Later today! Stable
Sorting Examples

- Use `key=function argument and reverse=True`
- What if we want to write our own function?

```
In[2]: a = ['red', 'orange', 'green', 'blue', 'indigo', 'violet']
In[3]: sorted(a)
Out[3]: ['blue', 'green', 'indigo', 'orange', 'red', 'violet']
In[4]: sorted(a,key=len)
Out[4]: ['red', 'blue', 'green', 'orange', 'indigo', 'violet']
In[5]: sorted(a,key=len,reverse=True)
Out[5]: ['orange', 'indigo', 'violet', 'green', 'blue', 'red']
```
Anonymous Functions

• Useful when want “throw-away” function
  • Our case mainly sort

• Syntax: `lambda PARAMETERS: EXPRESSION`
  • PARAMETERS – 0 or more comma separated
  • EXPRESSION – evaluates to something
Why is lambda used?

• It doesn't matter at all could use zeta? iota? …
  • https://en.wikipedia.org/wiki/Alonzo_Church

• Lisp and Scheme have lambda expressions
• Guido van Rossum, learned to live with lambda
What is a lambda expression?

• It's a function object, treat like expression/variable
  • Like list comprehensions, access variables

```python
>>> inc = lambda x: x + 1
>>> p = [1, 3, 5, 7]
>>> [inc(num) for num in p]
[2, 4, 6, 8]
```
Syntactic sugar
(makes the medicine go down)

- Syntactic sugar for a normal function definition

```python
def f(x):
    return x[1]
sorted(lst, key=f)
```

```python
f = lambda x: x[1]
sorted(lst, key=f)
```

```python
sorted(lst, key=lambda x: x[1])
```

```python
>>> d.items()
dict_items([("a", [1, 2, 3]), ("b", [4, 7]), ("c", [1, 1, 5, 8])])
>>> sorted(d.items(), key=lambda x: len(x[1]))
[("b", [4, 7]), ("a", [1, 2, 3]), ("c", [1, 1, 5, 8])]
>>> sorted(d.items(), key=lambda sparky : len(sparky[1]))
[("b", [4, 7]), ("a", [1, 2, 3]), ("c", [1, 1, 5, 8])]
```

Parameter name does not matter
Syntax and Semantics of Lambda

• Major use: single variable function as key

```python
>>> fruits = ['banana', 'apple', 'lemon', 'kiwi', 'pineapple']
>>> sorted(fruits)
['apple', 'banana', 'kiwi', 'lemon', 'pineapple']
>>> min(fruits)
'apple'
>>> max(fruits)
'pineapple'
>>> min(fruits, key=lambda f: len(f))
'kiwi'
>>> max(fruits, key=lambda z: z.count("e"))
'pineapple'
>>> sorted(fruits, key=lambda z: z.count("e"))
['banana', 'kiwi', 'apple', 'lemon', 'pineapple']
```
Compsci 101
Stable Sorting, Lambda, Clever Hangman
Part 3 of 3

Susan Rodger
October 20, 2020

>>> inc = lambda x: x + 1
>>> p = [1, 3, 5, 7]
>>> [inc(num) for num in p]
[2, 4, 6, 8]
How is the sorting happening?

```python
>>> d
{'a': [1, 2, 3], 'b': [4, 7], 'c': [1, 1, 5, 8]}
>>> sorted(d.items())
[['a', [1, 2, 3]], ['b', [4, 7]], ['c', [1, 1, 5, 8]]]
>>> sorted(d.items(), key=lambda x: x[1])
[ ['c', [1, 1, 5, 8]], ['a', [1, 2, 3]], ['b', [4, 7]]]
>>> sorted(d.items(), key=lambda x: x[1][-1])
[ ['a', [1, 2, 3]], ['b', [4, 7]], ['c', [1, 1, 5, 8]]]
```
How to do some “fancy” sorting

• lambda PARAMETER : EXPRESSION

• Given data: list of tuples: (first name, last name, age)
  [('Percival’, ‘Avram’,  51),
   (‘Melete’,   ‘Sandip’, 24), …]

• Think: What is the lambda key to sort the following?
  sorted(data, key=lambda z : (z[0],z[1],z[2]))
  • Sort by last name, break ties with first name
  • Sort by last name, break ties with age
  • Alphabetical by last name, then first name, then reverse age order
Creating Tuples with lambda

• Sort by last name, break ties with first name
  • key = lambda x: (x[1], x[0])

• Sort by last name, break ties with age
  • key = lambda x: (x[1], x[2])

• Alphabetical by last name, then first name, then reverse age order
  • key = lambda x: (x[1], x[0], -x[2])

• What if wanted something really different?
  • Sort alphabetical by last name, break ties by reverse alphabetical using first name
Leveraging the Algorithm

- Can’t sort by creating a tuple with lambda, use:
  - Pattern: Multiple-pass **stable** sort – first sort with last tie breaker, then next to last tie breaker, etc. until at main criteria

- Sort by index 0, break tie in reverse order with index 1
  
  ```
  [ (‘b’, ‘z’), (‘c’, ‘x’), (‘b’, ‘x’), (‘a’, ‘z’) ]
  [ (‘b’, ‘z’), (‘a’, ‘z’), (‘c’, ‘x’), (‘b’, ‘x’) ]
  [ (‘a’, ‘z’), (‘b’, ‘z’), (‘b’, ‘x’), (‘c’, ‘x’) ]
  ```

- **Stable** sort respects original order of "equal" keys
Stable sorting: respect "equal" items

- Female before male, each group height-sorted
  - First sort by height
Stable sorting: respect "equal" items

- Female before male, each group height-sorted
  - First sort by height
  - Then sort by gender
Understanding Multiple-Pass Sorting

```python
> a0 = sorted(data, key = lambda x: x[0])
> a1 = sorted(a0, key = lambda x: x[2])
> a2 = sorted(a1, key = lambda x: x[1])
> a0
[('a', 2, 0), ('b', 3, 0), ('c', 2, 5),
 ('d', 2, 4), ('e', 1, 4), ('f', 2, 0)]
> a1
[('a', 2, 0), ('b', 3, 0), ('f', 2, 0),
 ('d', 2, 4), ('e', 1, 4), ('c', 2, 5)]
> a2
[('e', 1, 4), ('a', 2, 0), ('f', 2, 0),
 ('d', 2, 4), ('c', 2, 5), ('b', 3, 0)]
```