# CMPSC 100 SPRING 2021

for what it’s worth…

# WELCOME & COURSE INFORMATION

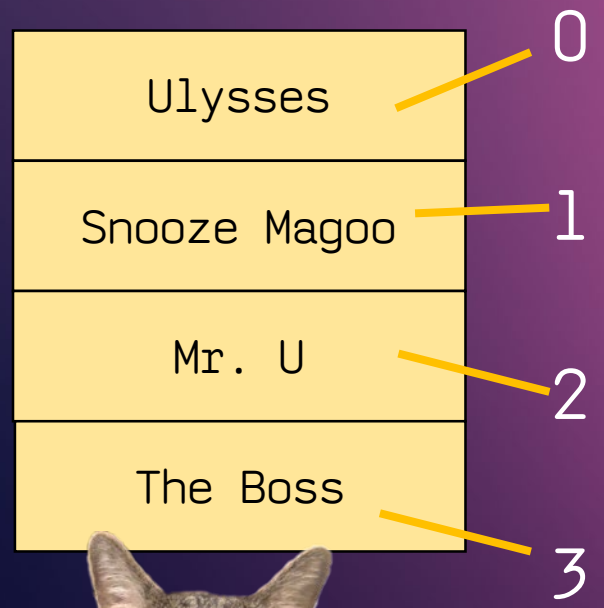
- Quiz has been posted to the course schedule
  - Recall that it’s due by 8:00a on Friday!
- Grading continues on Week 01 assignments
  - Final grades will be posted by Wednesday
- Some of you have expertly been using TL assistance
  - Don’t forget that we’re (TLs and me) are here to help!

LISSSSSTS

```
int_list = [0, 1, 2, 3
            4, 5, 6, 7]

str_list = [“This”, “is”, “a”
            “list”, “of”, “strings.”]
```

```
cat_names = [“Ulysses”, “Snooze Magoo”, “Mr. U”, “The Boss”]
```




```
cat_names[0]
```

I am soooo gonna shred your couch later…

# SELECTING PARTS OF LISTS ("SLICING")

end (uninclusive)

```
cat_names[   :   :   ]
```

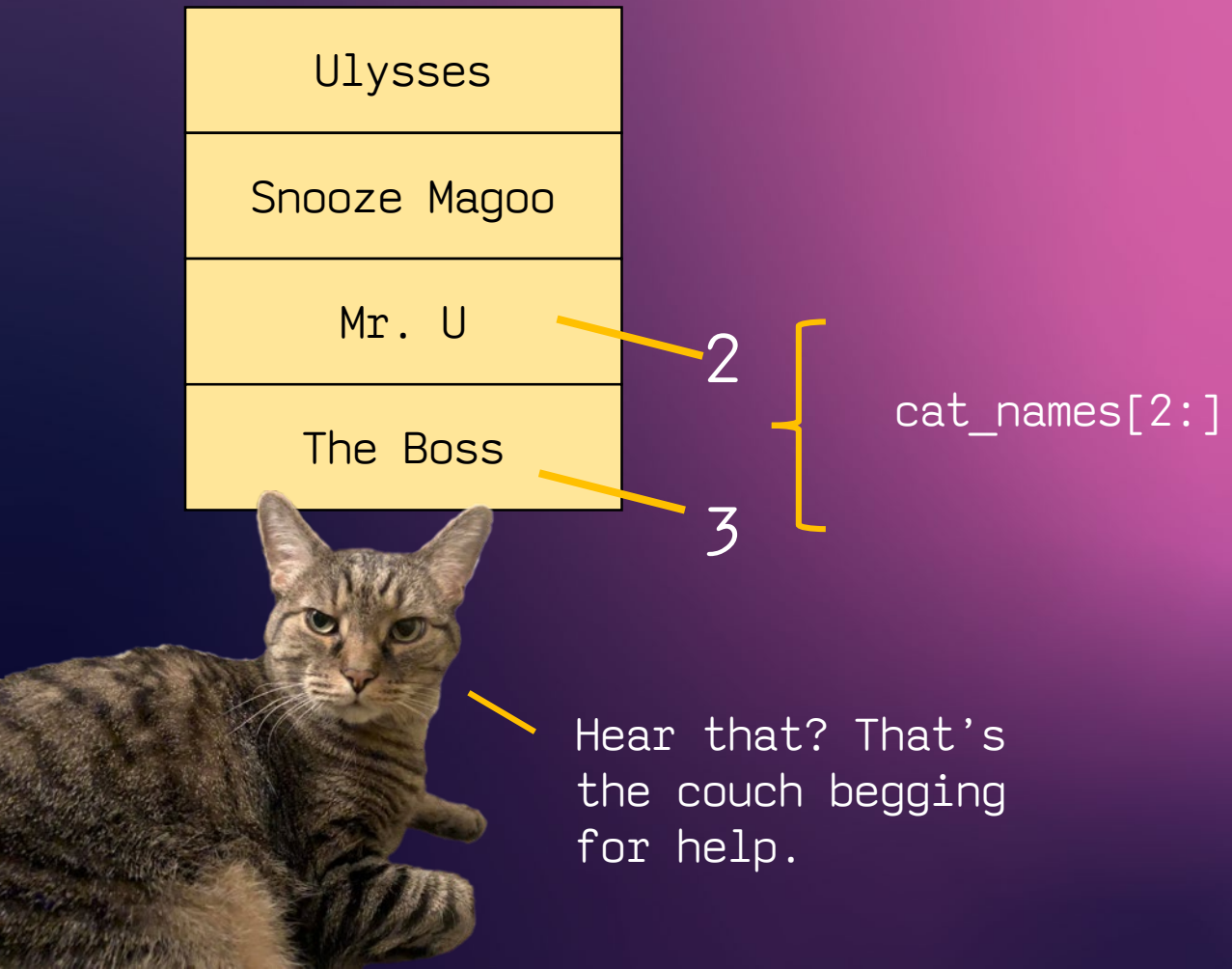
skip/ "jump"

start

| | |
|---|---|
| The Boss | 0 |
| Snooze Magoo | 1 |
| Mr. U | |
| Ulysses | |

cat_names[0:2]

At least these students know that I am, in fact, The Boss


Ulysses
Snooze Magoo
Mr. U
The Boss
2
3
cat_names[2:]
Hear that? That's the couch begging for help.

# LIST VS. TUPLES

- Lists:
  - are defined by square brackets []
  - can be modified
  - Ideal for values that change
- Tuples:
  - are defined by parenthesis
  - Cannot be modified
  - Ideal for <u>constants</u>
  - Sounds like a breakfast cereal

```
cat_names = ("Ulysses", "Snooze Magoo", "Mr. U", "The Boss")
```

| | |
|---|---|
| Ulysses | 0 |
| Snooze Magoo | 1 |
| Mr. U | 2 |
| The Boss | 3 |

Tuple is a funny name, tho - good one, Prof.

method name

```
cat_names.index("Snooze Magoo")
```

dot operator

argument

| Regular Assignments | Data Structures |
| --- | --- |
| number_of_people = 28 | names_of_students = [“Prof. Luman”,…] |
| Single values only, of any data type | Multiple values of any data type |
| By nature can only be one type | Can “mix-and-match” types |
| Treated as a single entity (“thing”) | Has indexes that represent “things” |
| Can’t be “sliced” | Can be “sliced” |
| If a “primitive” (integer, floating point) no methods (“powers”) | Has methods (“powers”) that it can use to perform special operations |

```
while CONDITION:
    # Do
    # all
    # these
    # things
# done
```

## while

Model behaviors -- operations to conduct while a condition is true

Executes all “member” statements until a condition is no longer true

Conditions can be simple (while light_switch == True) or complex (while light_switch == True and power == on)

Can be used to “count” by setting up a “sentinel variable”:

```
t = 10

While t > 0:
    print(t)
    t -= 1
```

Variable is created right here

```
for IDENTIFIER in DATA STRUCTURE:
    # do something with IDENTIFIER
# Variable still exists here
```

| while | for |
| --- | --- |
| Model behaviors -- operations to conduct while a condition is true | Iterate over items in a data structure |
| Executes all “member” statements until a condition is no longer true | Executes all “member” statements until a some data structure’s elements are exhausted |
| Conditions can be simple (while light_switch == True) or complex (while light_switch == True and power == on) | Cannot use conditions; must be used to, effectively, “count” things |
| Can be used to iterate over data structures, but it’s impractical:<br><br>`nums = [1,2,3,4]`<br>`n = 0`<br><br>`while n < len(nums) - 1:`<br>`  print(nums[n])`<br>`  n += 1` | Is more suited to counting:<br><br>`nums = [1,2,3,4]`<br><br>`for n in nums:`<br>`  print n` |

```
for name in cat_names:
    print(name) # prints each element of cat_names

print(name) # prints the last name “seen”
```