CMPSC 100 JANUARY 2021

Dictionaries, Control Structures (Loops)

COURSE INFORMATION

- Dylan’s office hours
  - Thursdays 1 - 3
    - That’s today!
- Video from yesterday is “chapterized”
- We’ll go a Google Meet again today
  - Today, I’d like to use this for Q & A as well
- Tomorrow, it’s only the Google Meet, starting at 9:30A

FEELING LIKE YOU'RE BEHIND?

- It's easy to say "don't," but here's what I know:

  We're at day 3, being able to ***read*** and understand code is enough

  Based on quizzes, I'm actually seeing that's true for most of us

TWO QUESTIONS FROM YESTERDAY

- How do we know when spacing is important?
  - The text answer: where intent/meaning is ambigious
  - Let’s look at an example
- What’s up with that “truth table” in day-1, worksheet 0.1.0?
  - Let’s talk about “logical” and “relational” operators

RELATIONAL OPERATORS

- We already know about == (equality)
- Here, we see:
  - > Greater than
  - < Less than
  - >= Greater/equal to
  - <= Less/equal to
  - != NOT equal

LOGICAL OPERATORS

- This is more complex:
  - and inclusive
  - or exclusive

I am awake and I am here:
class is likely happening

If I am awake or I am here:
the above not nece. true

SIGNIFICANT DIFFERENCES

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

method name

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

dot operator

argument

```
power_on = True

while power_on:
    choice = input("Turn the switch off? [Y]/[N]? ")
    if choice == "Y":
        power_on = False
```

Again, notice the indent:
it demonstrates what
"belongs" a while or if
"block"

```
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
```

FLOW OF CONTROL

```
power_on = True

while power_on:
    choice = input("Turn the switch off? [Y]/[N]? ")
    if choice == "Y":
        power_on = False
```

braces

```
a_dict = {}
```

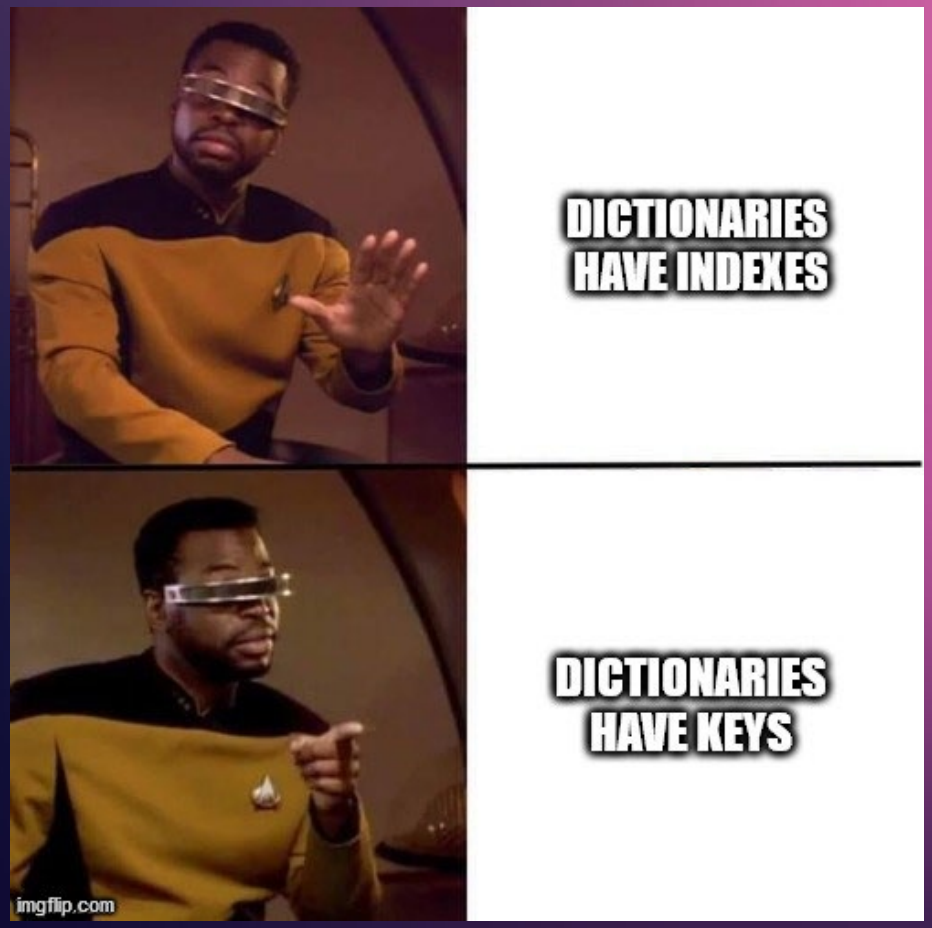
Fun fact: official name is the “Scottish brace”

(I don’t know why.)

```
students_are_awake = {
    True: 0,
    False: 16,
}
```

Keys

```
students_are_awake[True]
```


DICTIONARIES HAVE INDEXES
DICTIONARIES HAVE KEYS
imgflip.com

dict: ← “dictionary”

- Use brackets
- Are key: value pairs
- Do not use indexes
- Accommodate all data types

```
cat_names = {
      “Ulysses”: 3, # <- I was mad at him
      “Snooze Magoo”: 4,
      “The Boss”: 10,
      “Mane Man”: 3
}

print(cat_names[“Ulysses”])
```

```
cat_names[“The Bug”] = 1

cat_names = {
      “Ulysses”: 3,
      “Snooze Magoo”: 4,
      “The Boss”: 10,
      “Mane Man”: 3,
      “The Bug”: 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:
 	
`nums = [1,2,3,4]`
`n = 0`
 	
`while n < len(nums) - 1:`
`  print(nums[n])`
`  n += 1`
	Is more suited to counting:
	
	`nums = [1,2,3,4]`
	
	`for n in nums:`
	`  print n`

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

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

```
if “staples” in stock.keys():
    stock[“staples”] += 1
else:
    stock[“staples”] = 1

try:
    stock[“staples”] += 1
except KeyError:
    stock[“staples”] = 1
```