LECTURE 10

LOCAL VARIABLES; DICTIONARIES

MCS 260 Fall 2021
Emily Dumas

REMINDERS

- Homework 2 scores and solutions posted
- Project 1 due Friday, 6pm central
- Worksheet 4 solutions to be posted tomorrow
- Homework 4 to be posted tomorrow, due Tue Sep 21 at 10am
- Never submit anything after the deadline unless you have an extension

MAX()

In Python, the function `max` takes an iterable and returns a largest item in it.

```
max([2,6,0,2,0,2,1]) # returns 6
max("2775") # returns "7"
max(range(10_000)) # very slow way to get the number 9999
```

There is also `min()`.

LOCAL VARIABLES

Variables changed or created inside a function don't affect anything outside of the function.

Such variables are **local** to the function.

```
>>> def f():
...     x = 10  # local variable
...     print("x is",x)
...
>>> x=3 # danger! same name as local var
>>> f()
x is 10
>>> x
3
```

PARAMETERS ARE LOCAL, TOO

```
def g(x):
    x = x + 1
    print(x)

a = 5 # global var
g(a)  # prints 6
print(a) # prints 5
```

Equivalent to:

```
a = 5 # global var
# ---------------------------
x_local_to_g = a
x_local_to_g = x_local_to_g + 1
print(x_local_to_g)
# ---------------------------
print(a)
```

REASONS TO USE FUNCTIONS

- **Don't repeat yourself** (DRY). Capture often-used code in a function to make programs smaller and easier to maintain.
- Well-named functions make code more **readable**, expressing intent and deferring details.
- Local variables provide **isolation**, avoid accidental modification or reuse of variables.

DICTIONARIES

A **dictionary** or **dict** in Python is an *unordered* collection of *key* → *value* pairs.

Dictionaries can be indexed using keys, to get the associated values.

Other languages call this a **map** or **associative array**.

Dictionaries in Python are mutable.

Example of syntax for working with dictionaries:

```
>>> # define a new dict
>>> tuition = { "UIC": 10584,    # key "UIC" -> value 10584
...             "Stanford": 50703,
...             "Harvard": 46340 }
>>> # Get the value associated to a given key
>>> tuition["UIC"]
10584
>>> # Add or change the value at a key
>>> tuition["PSU"] = 18454
>>> tuition
{'UIC': 10584,
 'Stanford': 50703,
 'Harvard': 46340,
 'PSU': 18454}
>>> # Remove a key and value
>>> del tuition["Harvard"]
>>> tuition
{'UIC': 10584, 'Stanford': 50703, 'PSU': 18454}
```

Mixed types are ok for keys or values.

```
d = { 1: "fish", "two": "fish", "x": [7,6,5] }
```

Useful dict methods:

```
>>> d.values() # All values, without the keys
dict_values(['fish', 'fish', [7, 6, 5]])
>>> d.items()  # All key-value pairs (like enumerate(L))
dict_items([(1, 'fish'), ('two', 'fish'), ('x', [7, 6, 5])])
```

dict_keys, dict_items, dict_values types behave a lot like list, and can be converted to a list with **`list()`**.

MEMBERSHIP TESTING

Membership in a dictionary means being a *key*!

```
>>> d = { 1: "fish", "two": "fish", "x": [7,6,5] }
>>> "fish" in d
False
>>> 1 in d
True
```

Forgetting this is a very common source of programming errors.

ITERATION OVER DICTS

dicts are iterable, but iterate over the *keys*.

```
for k in d:  # loop over keys
    print(k,"is one of the keys")

for k in d:  # loop over keys (index to get value)
    print("Key",k,"has value",d[k])
```

It is common for the values in a dict to be dicts themselves. This is the usual way to make a collection of labeled data indexed by a key.

```
schooldata = {
    "UIC": {
        "fullname": "University of Illinois at Chicago",
        "tuition": 10584,
        "undergrad_students": 21641,
        },
    "Stanford": {
        "fullname": "Leland Stanford Junior University",
        "tuition": 50703,
        "undergrad_students": 7083
        },
    "Harvard": {
        "fullname": "Harvard University",
        "tuition": 46340,
        "undergrad_students": 6755
```

DICTIONARIES AS RULES

```
pr_replacements = {
    "accident": "unplanned event",
    "escape": "departure",
    "laser-sharks": "fish"
}
original = "an accident involving the escape of laser-sharks"
words = original.split()  # [ "an", "accident", ... ]
for w in words:
    if w in pr_replacements:
        w = pr_replacements[w]
    print(w,end=" ")
print()
```

Output:

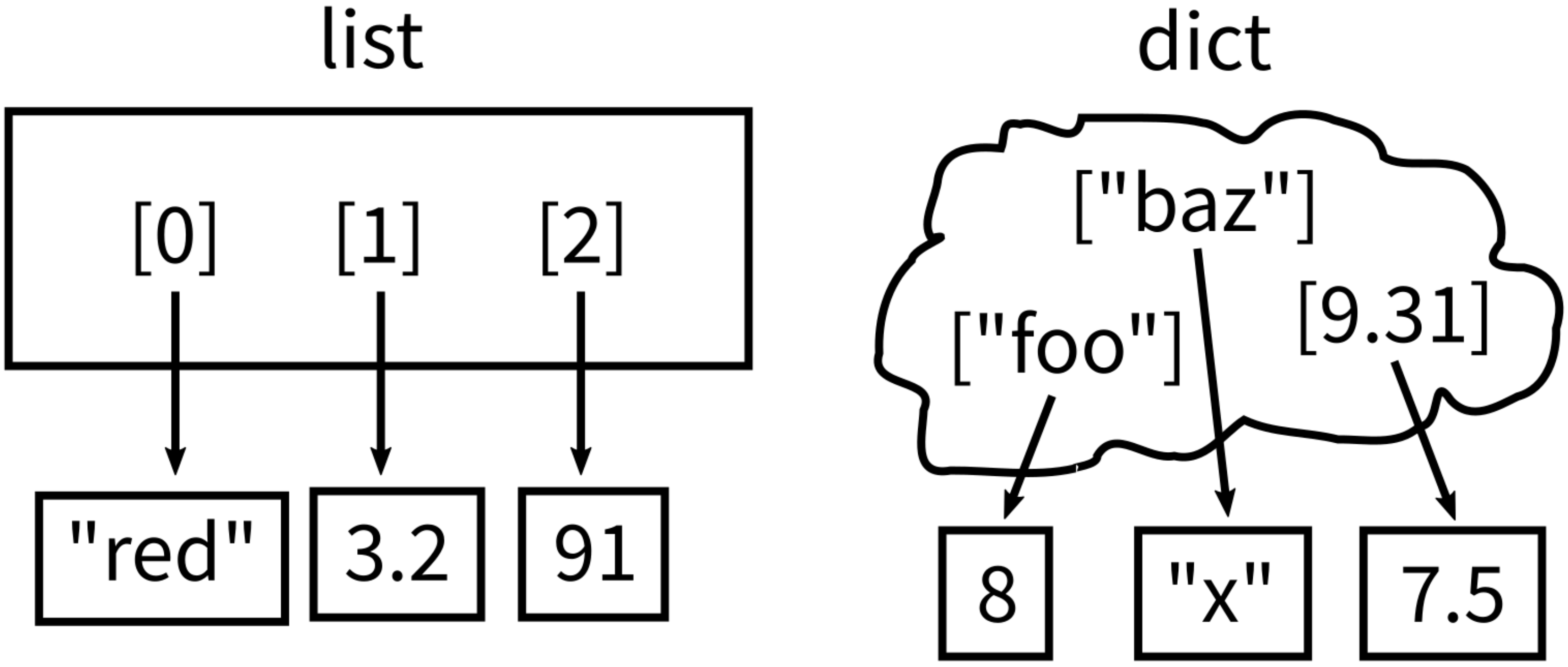
```
an unplanned event involving the departure of fish
```

HASHABLE TYPES

Not every type is allowable as a key in a dictionary.

Strings and numeric types are allowed. Lists and dicts are not.

In most cases, allowable keys are immutable types.



index to value
ordered

key to value
unordered

REFERENCES

- In *Downey*:
  - Chapter 11 covers dictionaries

ACKNOWLEDGEMENT

- Some of today's lecture was based on teaching materials developed for MCS 260 by Jan Verschelde.

REVISION HISTORY

- 2021-09-14 Initial publication