

CS 220 / CS319

Functions as Objects

Department of Computer Sciences
University of Wisconsin-Madison

Radical Claim:

Functions are Objects

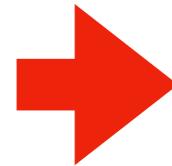
implications:

- variables can reference functions
- lists/dicts can reference functions
- we can pass function references to other functions
- we can pass lists of function references to other functions
- ...

Function References (Part 1)

Outline

- functions as objects
- sort
- lambda



```
l1 = [1, 2, 3]
```

```
l2 = l1
```

```
def f(l):  
    return l[-1]
```

```
g = f
```

```
num = f(l2)
```

which line of code is most novel for us?

State:

references

objects

`l1 = [1, 2, 3]`

`l2 = l1`

Explanation: `l1` should reference a new list object

Explanation: `l2` should reference whatever `l1` references

`def f(l):`

`return l[-1]`

Explanation: `f` should reference a new function object

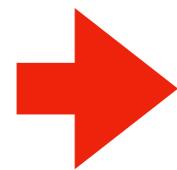
`g = f`

Explanation: `g` should reference whatever `f` references

`num = f(l2)`

Explanation: `l` should reference whatever `l2` references

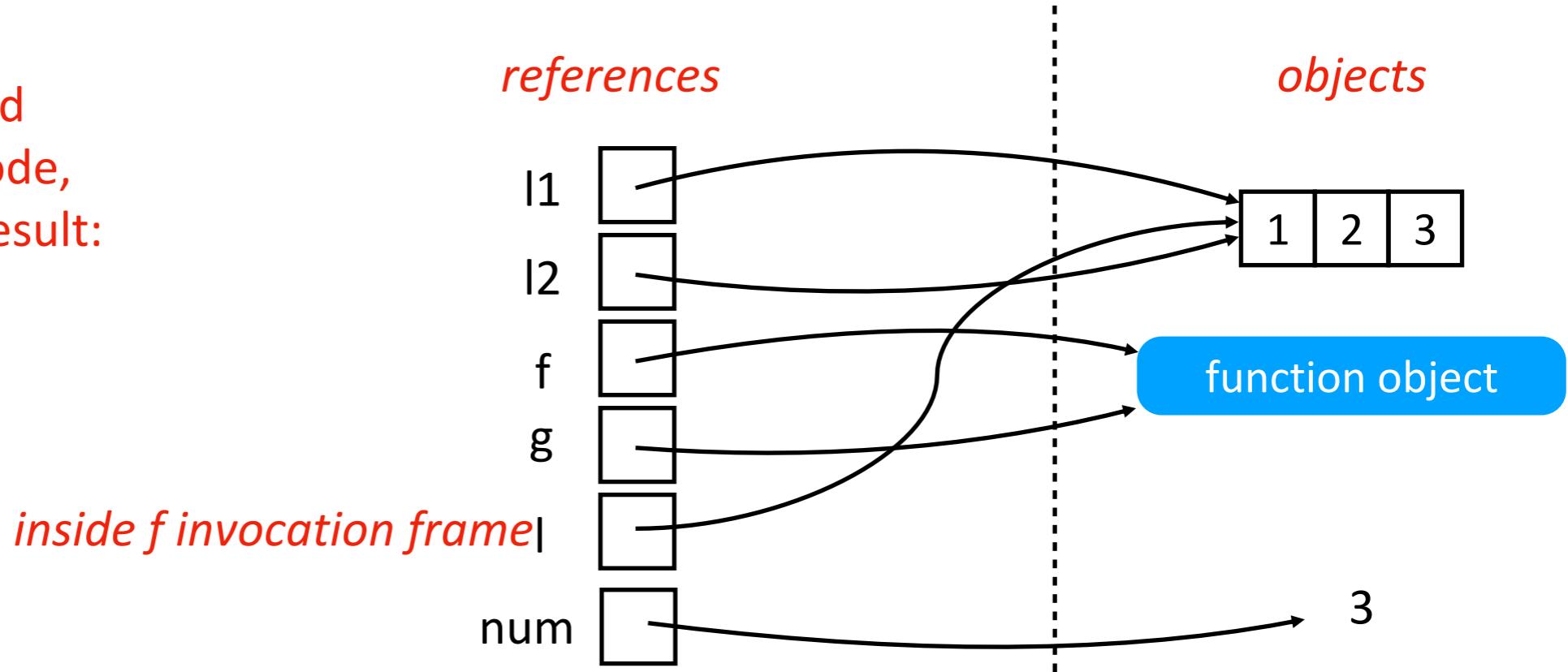
Explanation: `num` should reference whatever `f` returns



both these calls would have run the same code, returning the same result:

- `num = f(l1)`
- `num = g(l2)`

State:



```
|1 = [1, 2, 3]
```

```
|2 = |1
```

```
def f(l):  
    return l[-1]
```

```
g = f
```

```
num = f(|2)
```

very similar (reference new object)

very similar (reference existing object)

very different (invoke vs. reference)

CODING DEMOS

[Python Tutor]

Function References (Part 1)

Outline

- functions as objects
- `sort`
- lambda

Example: Sorting Names

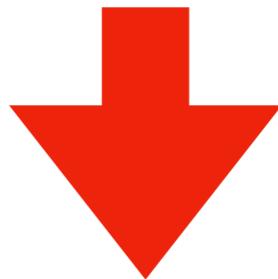
List of tuples:

```
names = [  
    ("Catherine", "Baker"),  
    ("Alice", "Clark"),  
    ("Bob", "Adams"),  
]
```

```
names.sort()
```

sorting tuples is done
on first element
(ties go to 2nd element)

Catherine	Baker
Bob	Adams
Alice	Clark



Alice	Clark
Bob	Adams
Catherine	Baker

Example: Sorting Names

List of tuples:

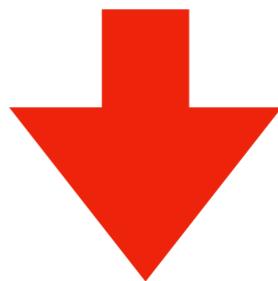
```
names = [  
    ("Catherine", "Baker"),  
    ("Alice", "Clark"),  
    ("Bob", "Adams"),  
]
```

```
names.sort()
```

what if we want to
sort by the last name?

or by the length of the name?

Catherine	Baker
Bob	Adams
Alice	Clark



Alice	Clark
Bob	Adams
Catherine	Baker

Example: Sorting Names

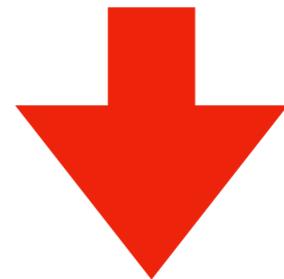
List of tuples:

```
names = [  
    ("Catherine", "Baker"),  
    ("Alice", "Clark"),  
    ("Bob", "Adams"),  
]
```

```
def extract(name_tuple):  
    return name_tuple[1]
```

```
names.sort(key=extract)
```

Catherine	Baker
Bob	Adams
Alice	Clark



Bob	Adams
Catherine	Baker
Alice	Clark

Example: Sorting Names

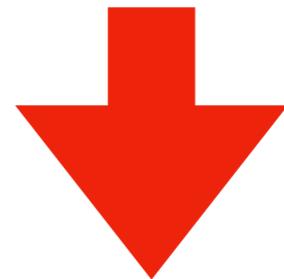
List of tuples:

```
names = [  
    ("Catherine", "Baker"),  
    ("Alice", "Clark"),  
    ("Bob", "Adams"),  
]
```

```
def extract(name_tuple):  
    return len(name_tuple[0])
```

```
names.sort(key=extract)
```

Catherine	Baker
Bob	Adams
Alice	Clark



Bob	Adams
Alice	Clark
Catherine	Baker

CODING DEMOS

[Jupyter notebook]

Function References (Part 1)

Outline

- functions as objects
- sort
- lambda

Example: Sorting Dictionary by keys using lambdas

- lambda functions are a way to abstract a function reference
- multiple possible parameters and single expression as function body

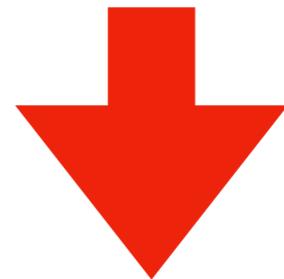
bob	20
alice	8
alex	9

lambda parameters: expression

Dictionary:

```
players = {"bob": 20, "alice": 8, "alex": 9}
```

```
dict(sorted(players.items(),  
key = lambda item: item[0]))
```



alex	9
alice	8
bob	20

Example: Sorting Dictionary by keys using lambdas

- lambda functions are a way to abstract a function reference
- multiple possible parameters and single expression as function body

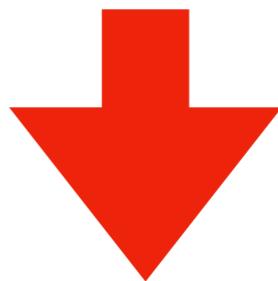
bob	20
alice	8
alex	9

lambda parameters: expression

Dictionary:

```
players = {"bob": 20, "alice": 8, "alex": 9}
```

```
dict(sorted(players.items(),  
key = lambda item: item[1]))
```



alice	8
alex	9
bob	20