

# [220 / 319] Functions as Objects

Meena Syamkumar  
Andy Kuemmel

# 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 I)

## Outline

- functions as objects
- sort
- lambda

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

```
y = x
```

```
def f():  
    return "hi"
```

```
g = f
```

```
z = f()
```

your notes should probably include this example, with an explanation of what each of the 5 steps do!

*which line of code is most novel for us?*

x = [1, 2, 3]

y = x

def f():  
 return "hi"

g = f

➔ z = f()

**Explanation:** x should reference a new list object

**Explanation:** y should reference whatever x references

**Explanation:** f should reference a new function object

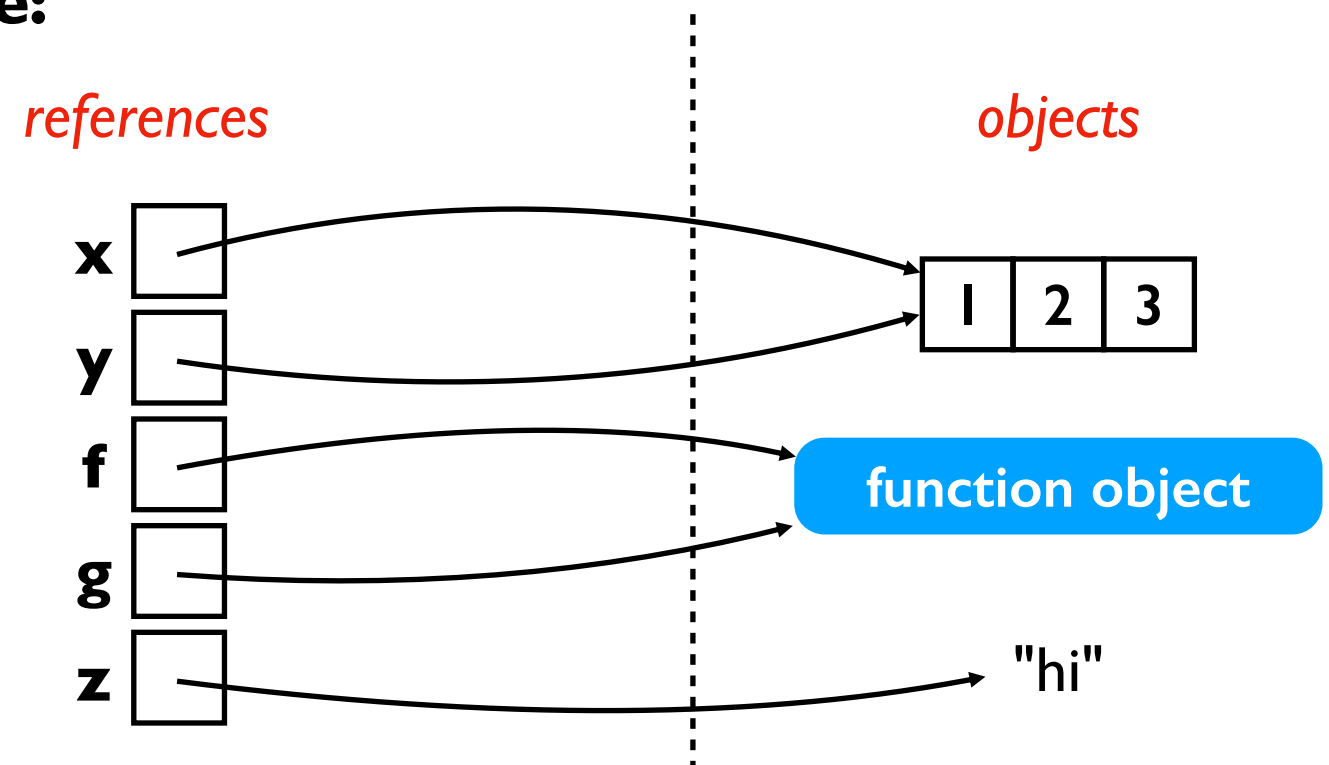
**Explanation:** g should reference whatever f references

**Explanation:** z should reference whatever f returns

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

- z = f()
- z = g()

**State:**



x = [1,2,3]

y = x

```
def f():  
    return "hi"
```

g = f

z = f()

**very similar** (reference new object)

**very similar** (reference existing object)

**very different** (invoke vs. reference)

# **CODING DEMOS**

## [Python Tutor]

# Function References (Part I)

## Outline

- functions as objects
- `sort`
- `lambda`



# Example: Sorting Names

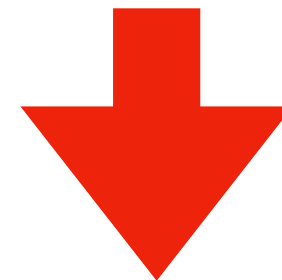
List of tuples:

```
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:

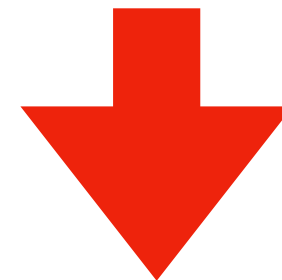
```
names = [  
    ("Catherine", "Baker"),  
    ("Alice", "Clark"),  
    ("Bob", "Adams"),  
]
```

```
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

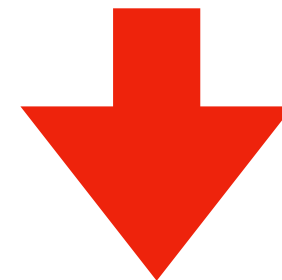
List of tuples:

```
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

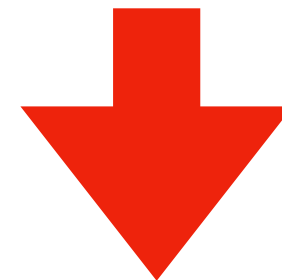
List of tuples:

```
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 I)

## Outline

- functions as objects
- sort
- **lambda**

# Example: Sorting Dictionary by keys using lambdas

*lambda functions are a way to abstract a function reference*

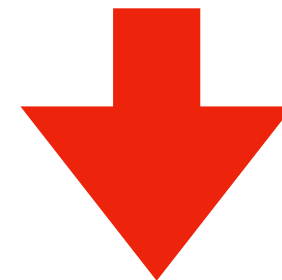
***lambda arguments: expression***

Dictionary:

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

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

bob	20
alice	8
alex	9



alex	9
alice	8
bob	20

# Example: Sorting Dictionary by values using lambdas

*lambda functions are a way to abstract a function reference*

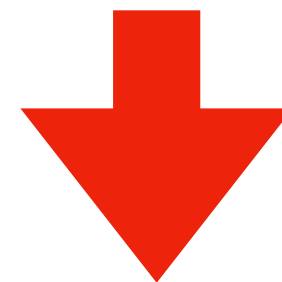
***lambda arguments: expression***

Dictionary:

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

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

bob	20
alice	8
alex	9



alice	8
alex	9
bob	20