# [220] Functions as Objects

Meena Syamkumar
Mike Doescher

**Cheaters caught: 0**
**(Through P5)**
**12 suspicious works for P7**
**(email Mike to confess)**

# 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

```
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?*

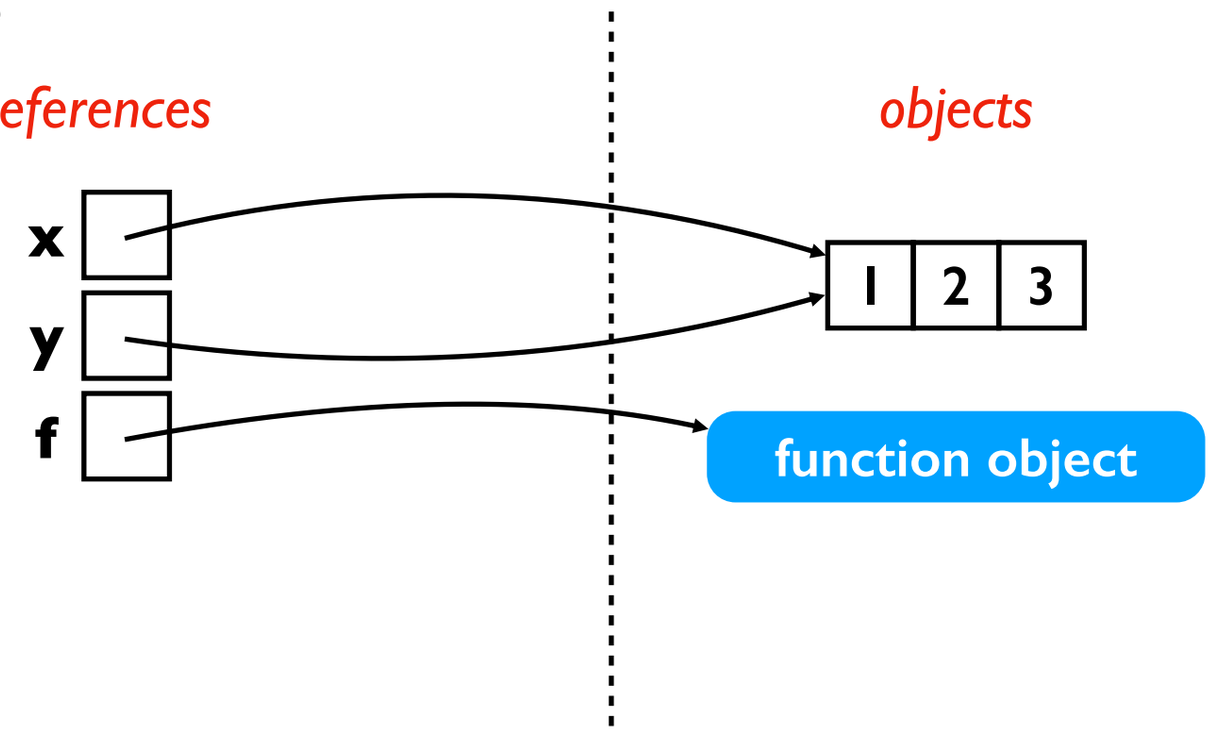
```
x = [1,2,3]
y = x

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

**State:**



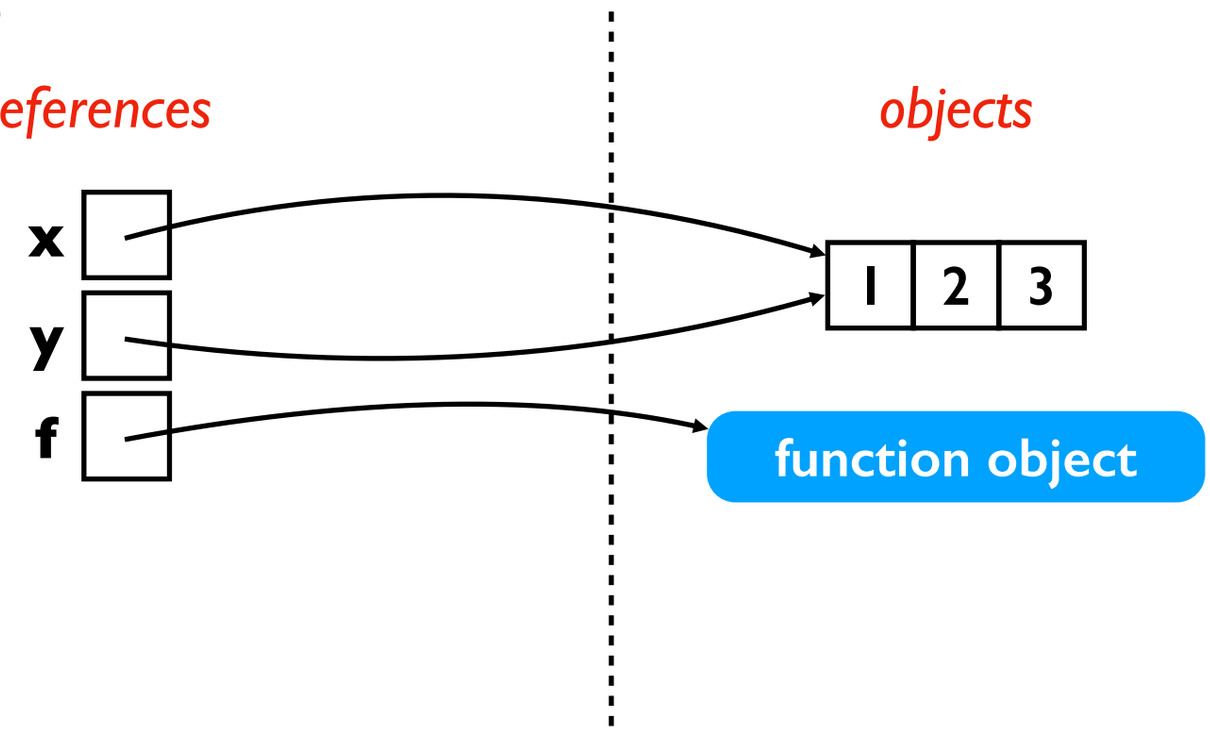
```
x = [1,2,3]
y = x

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

**State:**



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

def f():
    return "hi"

g = f

z = f()
```

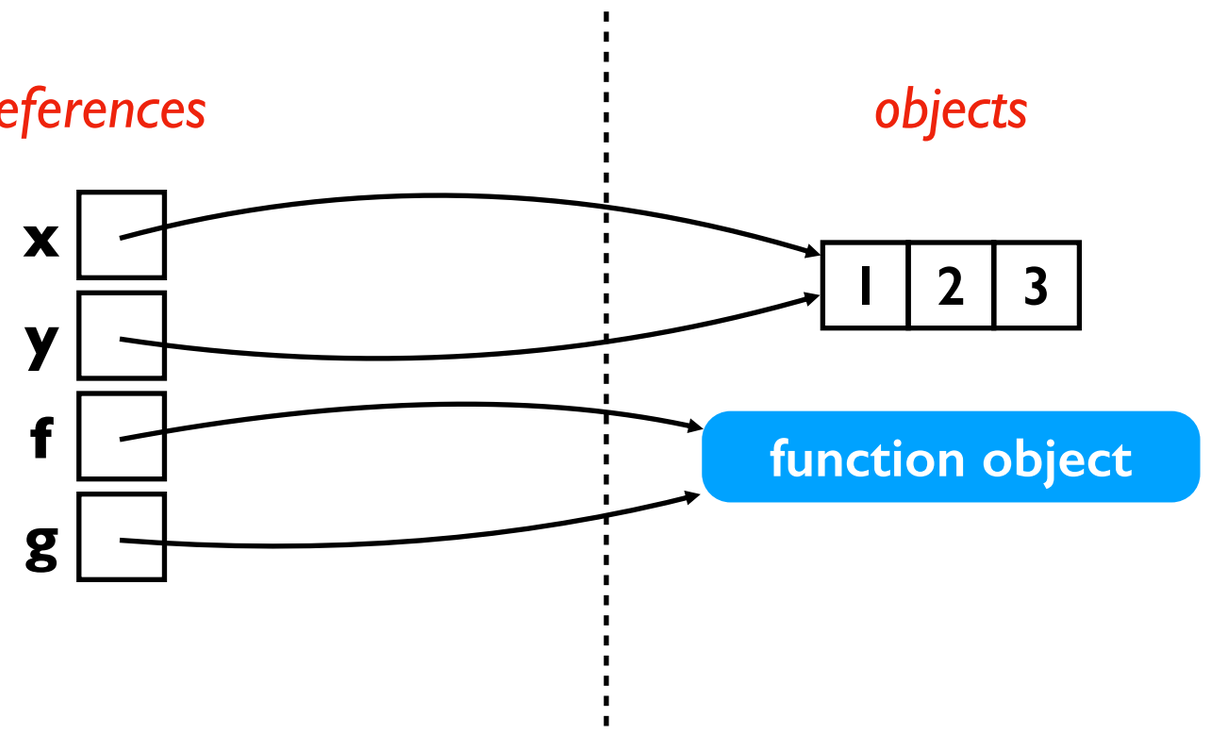
**Explanation:** x should reference a new list object

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

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

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

**State:**



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

def f():
    return "hi"

g = f

z = f()
```

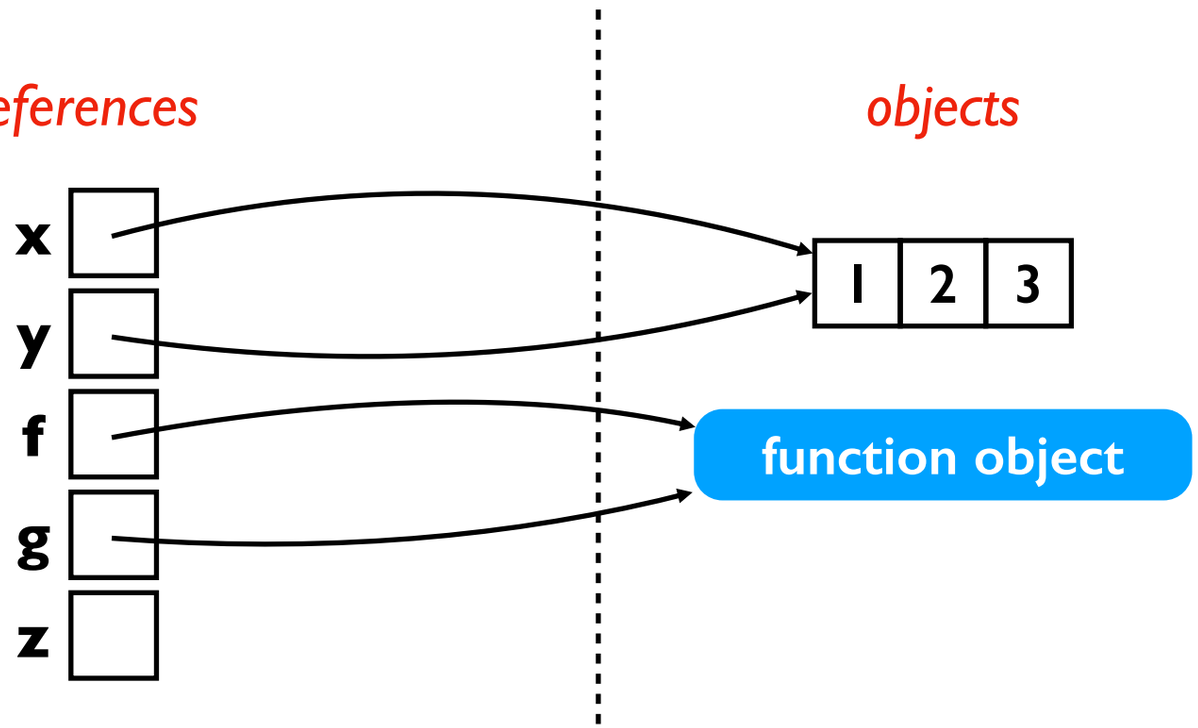
**Explanation:** x should reference a new list object

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

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

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

**State:**



```
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

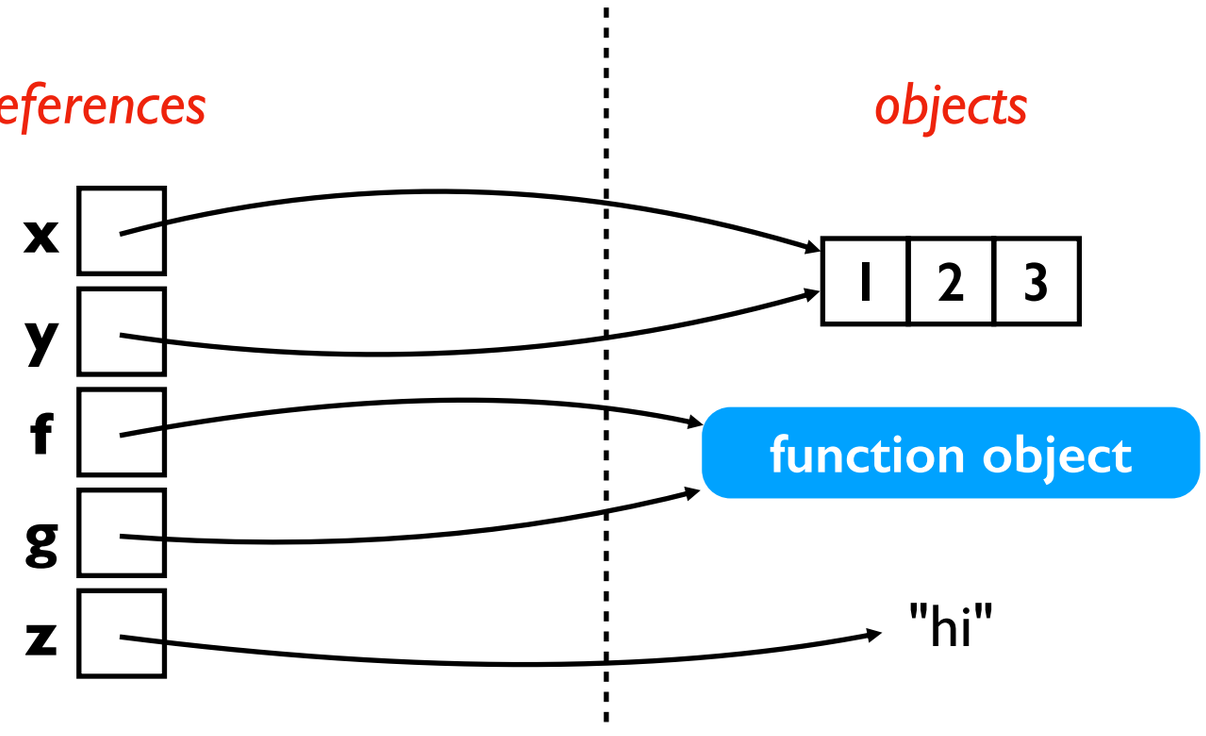
**Explanation:** g should reference whatever f references

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

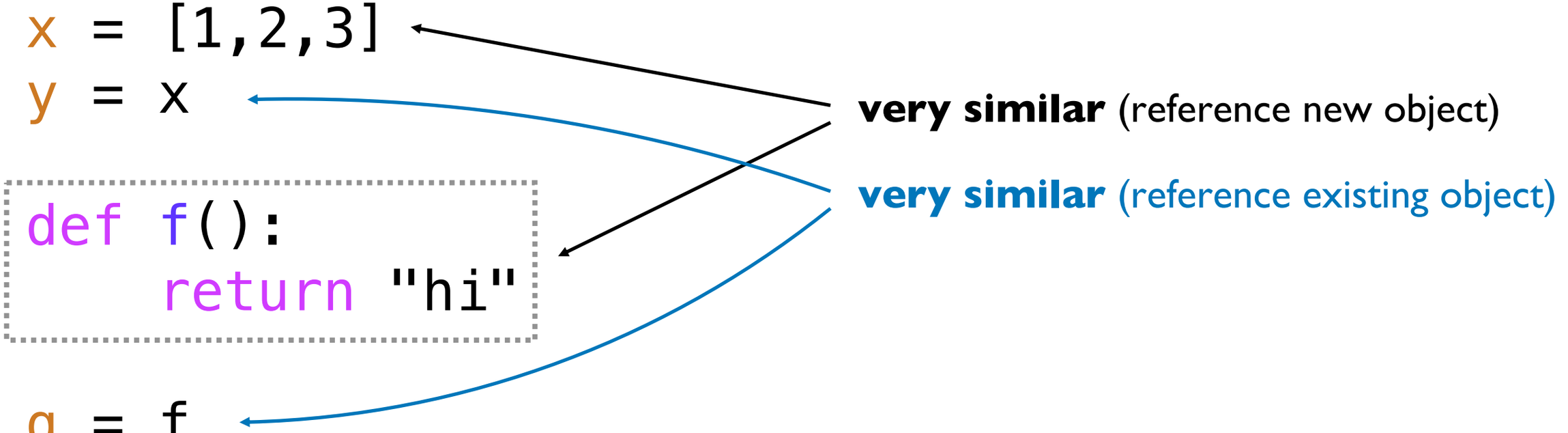
both of these calls would have run the same code, returning the same result:

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

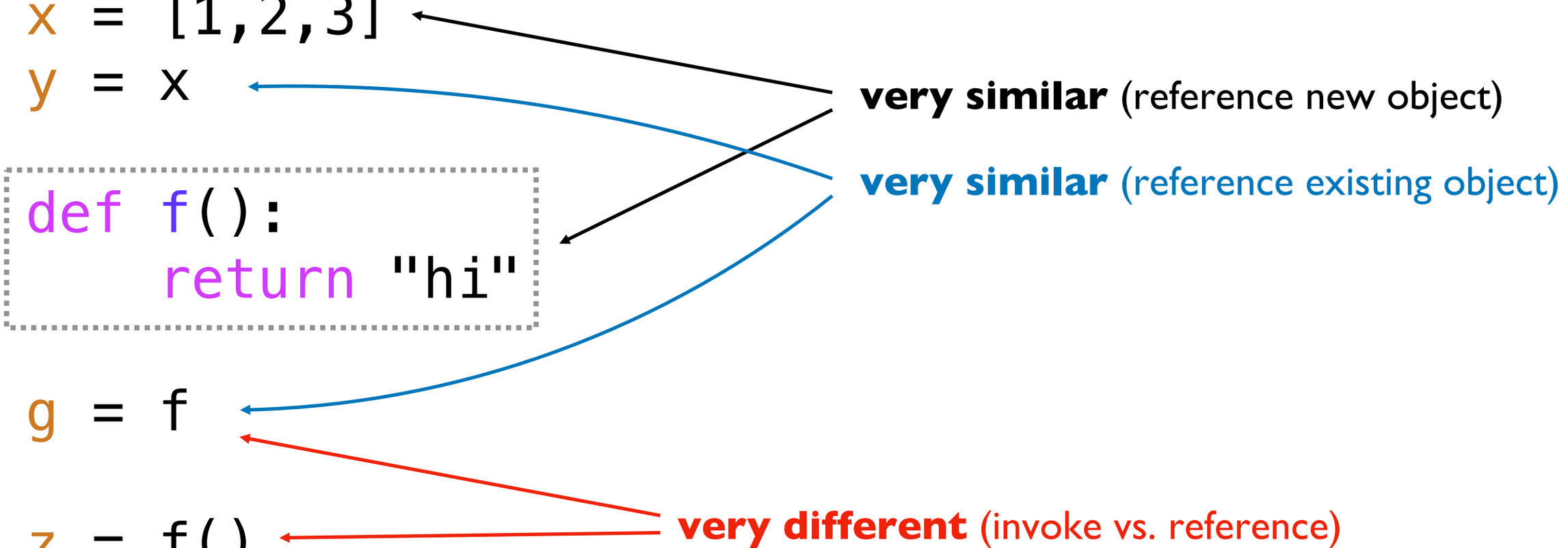
**State:**



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

def f():
    return "hi"

g = f

z = f()
```



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

def f():
    return "hi"

g = f

z = f()
```



# CODING DEMOS

# [Python Tutor]

# Function References (Part 1)

Outline

- functions as objects
- sort

# Example: Sorting Names

List of tuples:

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

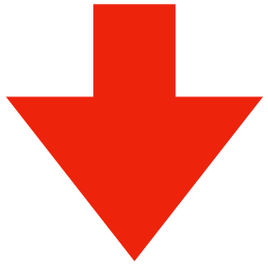
| Catherine | Baker |
|---|---|
| Bob | Adams |
| Alice | Clark |

# Example: Sorting Names

List of tuples:

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

| Catherine | Baker |
|---|---|
| Bob | Adams |
| Alice | Clark |



```
names.sort()
```

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

| Alice | Clark |
|---|---|
| Bob | Adams |
| Catherine | Baker |

# Example: Sorting Names

List of tuples:

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

| Catherine | Baker |
|---|---|
| Bob | Adams |
| Alice | Clark |

```
names.sort()
```

**what if we want to sort by the last name?**

**or by the length of the name?**

| 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 |

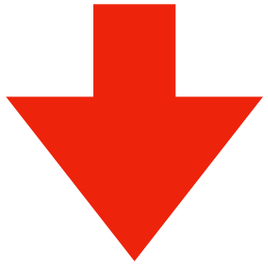
| Bob | Adams |
|---|---|
| Catherine | Baker |
| Alice | Clark |

# Example: Sorting Names

List of tuples:

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

| Catherine | Baker |
|---|---|
| Bob | Adams |
| Alice | Clark |



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

names.sort(key=extract)
```

| Bob | Adams |
|---|---|
| Alice | Clark |
| Catherine | Baker |