COL1000: Introduction to Programming

Functions

Subodh Sharma | Lec 20 | Sept 26

Quiz

Quiz

```
x = 10

def foo():

    print(x)

foo()
```

Quiz

```
x = 10

def foo():

    print(x)

foo()
```

- Is this correct ?

Quiz

```
x = 10

def foo():

    print(x)

foo()   # Yes — reading & printing the global value
```

- Is this correct ?

Quiz

```
x = 10

def foo():

    print(x)

foo()
```

- Is this correct ?

Quiz

```
x = 10

def foo():

    print(x)

foo()
```

- Is this correct ?

```
x = 10

def foo():

    print(x)

    x = 15

foo()
```

Quiz

```
x = 10
def foo():
    print(x)
foo()
```

- Is this correct ?

```
x = 10
def foo():
    print(x)
    x = 15
foo()
```

- Is this correct ?

Quiz

```
x = 10

def foo():

    print(x)

foo()
```

- Is this correct ?

```
x = 10

def foo():

    print(x)    # UnboundLocalError

    x = 15

foo()
```

- Is this correct ?

Quiz

```
x = 10
def foo():
    print(x)
foo()
```

- Is this correct ?

```
x = 10
def foo():
    print(x)
    x = 15
foo()
```

- Is this correct ?

Functions: Closures

Functions: Closures

- A **closure** is a function object that **remembers values from its enclosing scope even after that scope has finished its execution**

Functions: Closures

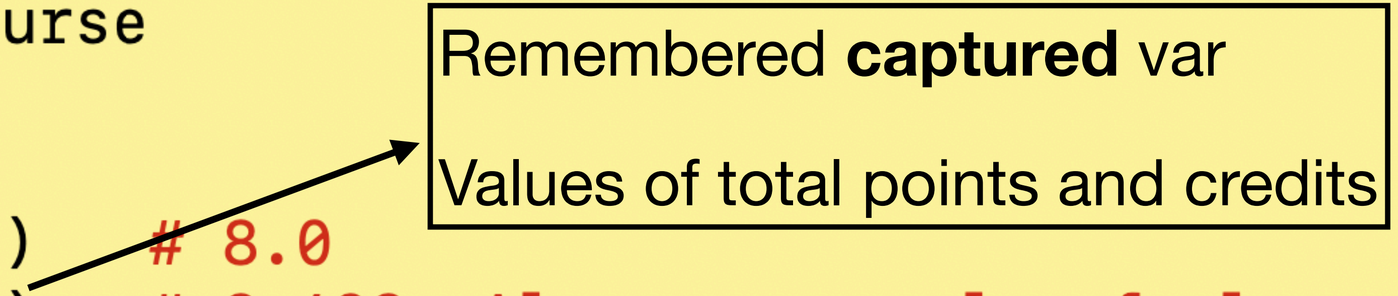
- A **closure** is a function object that **remembers values from its enclosing scope even after that scope has finished its execution**

```
def make_gpa():
    total_points = 0.0
    total_credits = 0.0
    def add_course(grade_point, credits):
        nonlocal total_points, total_credits
        total_points  += grade_point * credits
        total_credits += credits
        return total_points / total_credits
    return add_course

gpa = make_gpa()
print(gpa(8.0, 4))   # 8.0
print(gpa(9.0, 3))   # 8.428. Also an example of closure
```

Functions: Closures

- A **closure** is a function object that **remembers values from its enclosing scope even after that scope has finished its execution**

```
def make_gpa():
    total_points = 0.0
    total_credits = 0.0
    def add_course(grade_point, credits):
        nonlocal total_points, total_credits
        total_points  += grade_point * credits
        total_credits += credits
        return total_points / total_credits
    return add_course

gpa = make_gpa()
print(gpa(8.0, 4))    # 8.0
print(gpa(9.0, 3))    # 8.428. Also an example of closure
```



Functions: Lambda

- **Lambda functions**: They are anonymous functions (i.e. don't have a user-specified name)
  - Syntax: `lambda arguments: expression`
    - Arguments can be 0 or more, but **have a single expression** whose **results are returned**!
    - E.g., `add_five = lambda x: x+5; print(add_five(7)) #12`
    - E.g., `lst = list(map(lambda x: ?, [1,2,3]))`

Functions: Exceptions

- Exceptions in functions **denote errors or unexpected conditions**
- There are two ways of handling exceptions in the code:
  - Explicitly **raising the errors with an appropriated exception message**
  - **Catching** them to handle them gracefully
  - Let us see examples of each!