CS 61λ

Discussion 1

Link: cs61a.org/disc/disc01.pdf

Survey: [redacted]

Song Rec:

I'm Still Standing
~~Here Comes The Sun~~

by ~~The Beatles~~
Elton John. ♥

Announcements

Note On Preparing For Exams

cs61a.amks.me/practisetests

Drop me a hi :)

arushisomani@berkeley.edu

Zoom Link:

amks.me/zoom

My 61A Stuff:

cs61a.amks.me

Discussion 1

Control and Environments

Expressions vs Statements

Cannot print statements
print(a=3) will error

Expressions

Evaluate to values

```
>>> 3 + 4 + 2 + 1
10
>>> "hello"
"hello"
```

>>>3
3
>>> 1<2
True

Statements

Do not evaluate to values, instead determine some kind of change in control flow.

an assignment/binding statement

```
>>> a = 3
>>>
```

>>> a = 3+4+2+1 = 10

inside memory:
A location → 10
given name ← a

Control Flow

Control flows downwards in a program

Things are evaluated one after another.

```
a = 3

print(a)
```

⇒ print(3)

Inside a statement, control flows left to right.

Control Flow with IF

Conditional execution of statements.

```
① if (3 > 4):
      print (1)
② else:
③ print (2)
```

false

skipped ← is not evaluated →

could be print(1/0) and wouldn't error

is evaluated →

Control Flow with AND/OR

(short circuiting)

AND returns the first False value or last True value.

```
3 and 4 and 0 and 1
```

Truthy Truthy Falsy Truthy

evaluation stops

0

Truthy >>> 1 and 1/0 and False

Error

Error

OR returns the first True value or last False value.

```
0 or False or 1 or 0
```

Falsy Falsy Truthy Falsy

evaluation stops

1

Truthy >>> 1 or 1/0 or False

eval stops

Control Flow with WHILE

While loops create iterative control flow.

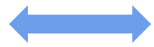
```
n = 1

while (n > 3):
    print (n)
    n = n +1

print("End")
```

initialisation

condition

body

updation



```
n = 1

print (n)
n = n +1
print (n)
n = n +1

print("End")
```

initialisation
assigning starting value

condition
till when does it loop? [the opposite of condition is when it terminates]

body
whatever we want to do repeatedly

updation
changing the initialized value in a way that gets us closer and closer to making the condition false.

Question 1.1

jacket if below 60 or raining

```
def wears_jacket_with_if(temp, raining):
    """
    >>> wears_jacket_with_if(90, False)
    False
    >>> wears_jacket_with_if(40, False)
    True
    >>> wears_jacket_with_if(100, True)
    True
    """
```

type? Integer type? Boolean.

return type? boolean

```
    if temp < 60 or raining:
        return True  # jacket
    else:
        return False  # no jacket
```

Question 1.1.2

In one line...

```
def wears_jacket(temp, raining):
    return temp < 60 or raining
```

(Try to convince yourself that this is correct)

Question 1.2

1.2 What is the result of evaluating the following code?

```
def square(x):
    print("here!")
    return x * x

def so_slow(num):
    x = num
    while x > 0:
        x = x + 1
    return x / 0

square(so_slow(5))
```

=5

x=5

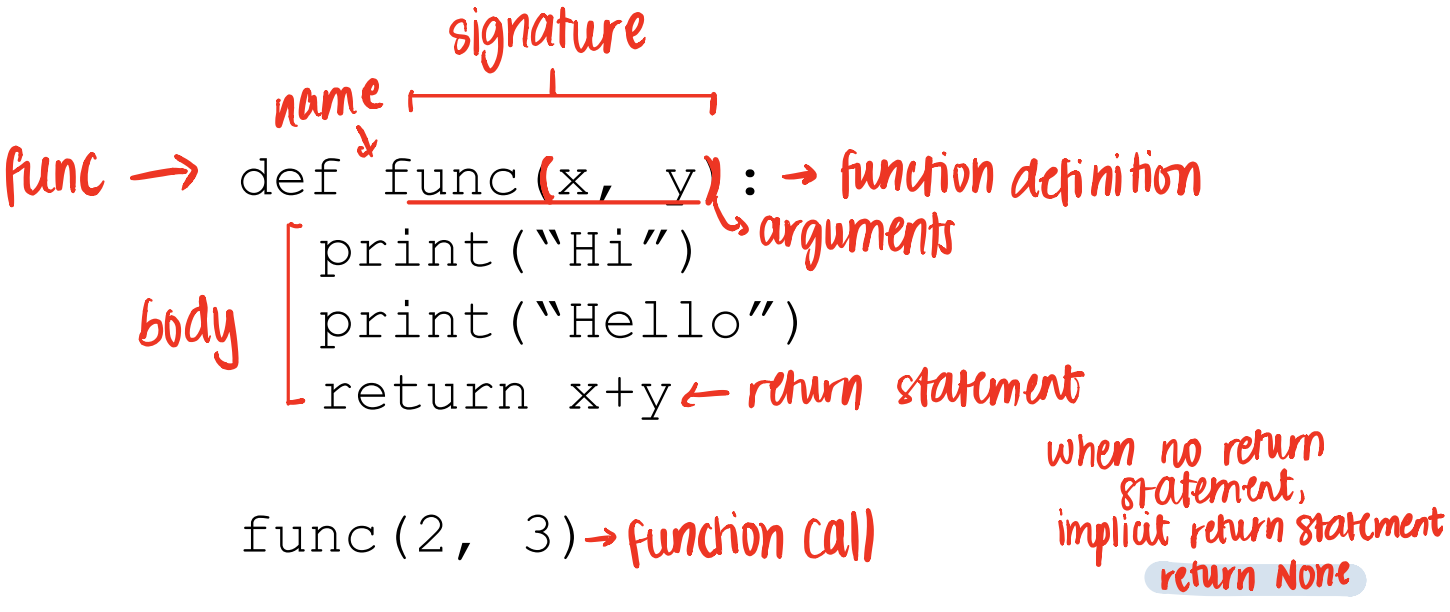
→ loops infinitely ☐

Question 1.3

```
def is_prime(n):
    """
    >>> is_prime(10)
    False
    >>> is_prime(7)
    True
    """
    if n <= 1:
        return False
    i = 2
    while( i < n ):
        if n % i == 0:
            return False
        i += 1
    return True
```

→ can also be upto sqrt(n) for same results

Anatomy of a Function



Functions are outside of Control Flow

Output:
Outside
<function f at --->

```
def f(x):
    print (“Inside”)
    return 1/0


print(“Outside”)
print(f)
```

How Functions Are Evaluated

Output:
Inside

x=3

```
def f(x):
    print ("Inside")
    return x + 1
```

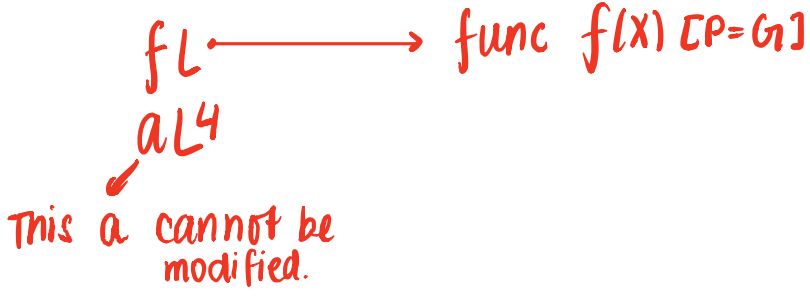
3+1=4

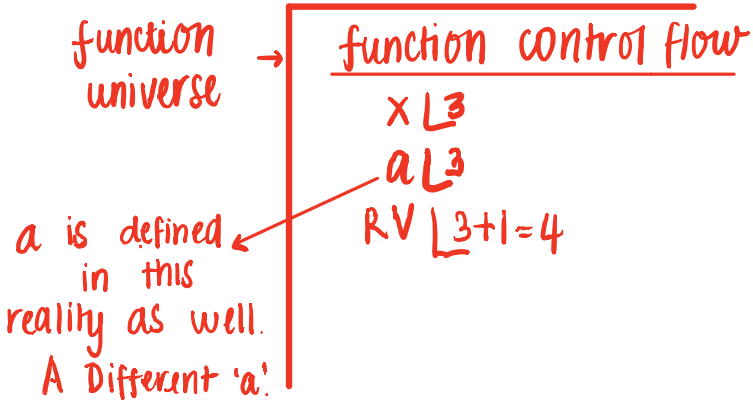
f(3)=4

Function Frames Are Independent

"scope"

```
def f(x):
    a = 3
    return x + 1

a = 4
f(3) = 4
print(a) = 4
print(x) = Error
```





Function Frames Can Access Parent Frame Bindings

```
def f(x):
    b = a + 3
    return x + 1

a = 4
f(3)
print(a)
print(b)
```

f L ⟶ func f(x) [P=G]

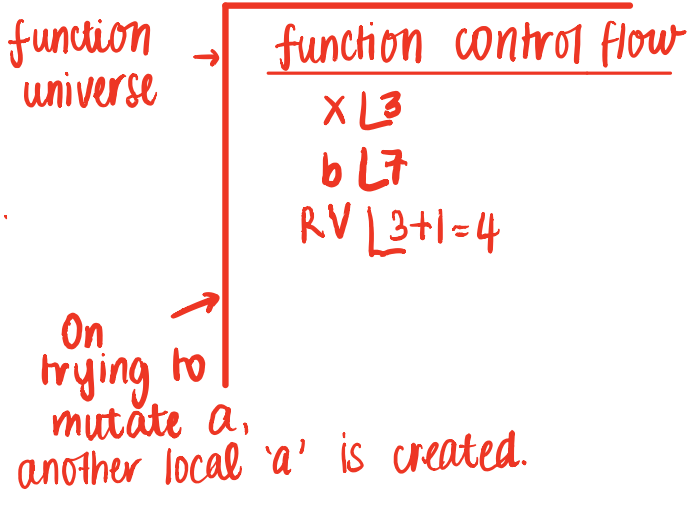
a L4

↘ This can be accessed but not mutation.

f(3) = 4

print(a) = 4

print(b) = Error



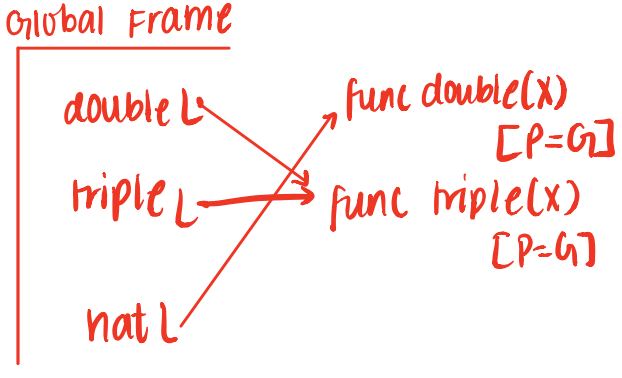
Question 2.2

2.2 Use these rules and the rules for assignment statements to draw a diagram for the code below.

```
def double(x):
    return x * 2
def triple(x):
    return x * 3
hat = double
double = triple
```

never executed

never executed

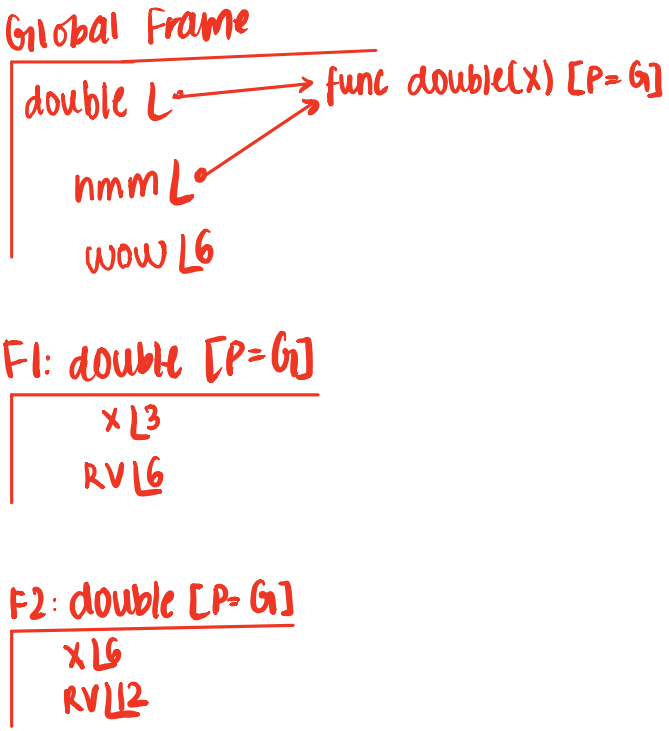


Question 2.3

```
def double(x):
    return x * 2

hmmm = double
wow = double(3)
hmmm(wow)
```

=12

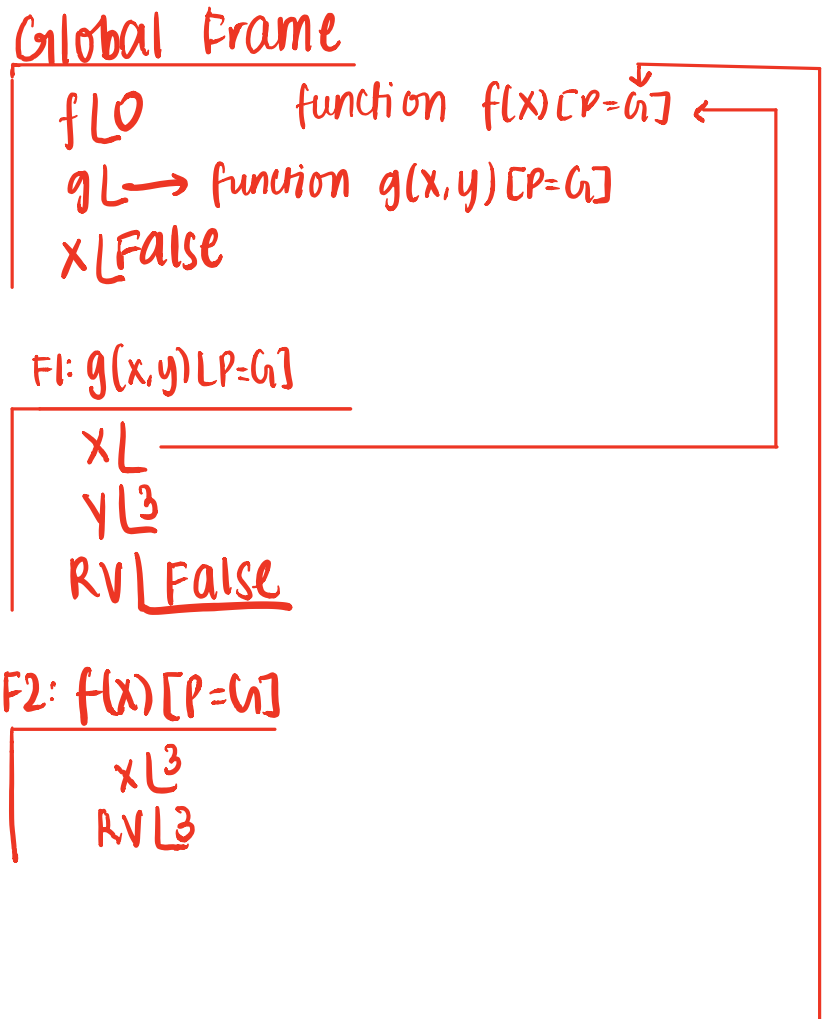
double (6)



Question 2.4

```
def f(x):
        return x

def g(x, y):
        if x(y):
                return not y
        return y

x = 3
x = g(f, x)
f = g(f, 0)
```

one func, multiple pointers



F3: g(x, y) [P=G]

x 1

y 0

RV 0

F4: f(x) [P=G]

x 0

RV 0

Prints In Prints

Assignment is not an expression. Function arguments only accept expressions.

```
print(a=3)
```

→ Errors

```
print(print(2))
```

evaluates to None

2
None.

Preview Of Next Week

Higher Order Functions

1. Functions accept/return python values
2. Functions are python values
3. Functions can accept/return functions

Higher Order Functions

```
def make_adder(add_value):
    def adder(input_val):
        return add_value + input_val
    return adder


make_adder(1)(3)
```

Try it in Python Tutor.

Higher Order Functions

```
def compose(func1, func2):
    def f(x):
        return func1(func2(x))
    return f

def add_one(x):
   return x+1
def mul_two(x):
   return 2*x

compose(add_one, mul_two)(3)
```

Try it on Python Tutor.

Quiz!

solutions on cs61a.org

```
x = 3

def p(rint):
        print(rint)

def g(x, y):
        if x:
                print("one")
        elif x:
                print(True, x) # Does x being truth-y affect the printed value?
        if y:
                print(True, y) # Does y being truth-y affect the printed value?
        else:
                print(False, y) # Does y being false-y affect the printed value?
        return print(p(y)) + x
```

Quiz!

Expression	Interactive Output
`print(4, 5) + 1`	`4 5` `Error`
`2 * 2 * 1 + x * x`	
`print(3 * 3 * 1)`	
`print(x + 1 * x + 1)`	
`print(print(x + 1 * x + 1))`	
`print(print(x + 1 * x + 1) + 1)`	
`print(p("rint"))`	
`x, y = 2, x` `g(y, x)`	
`g(y, p("rint"))`	

Feedback!

cs61a.amks.me/feedback