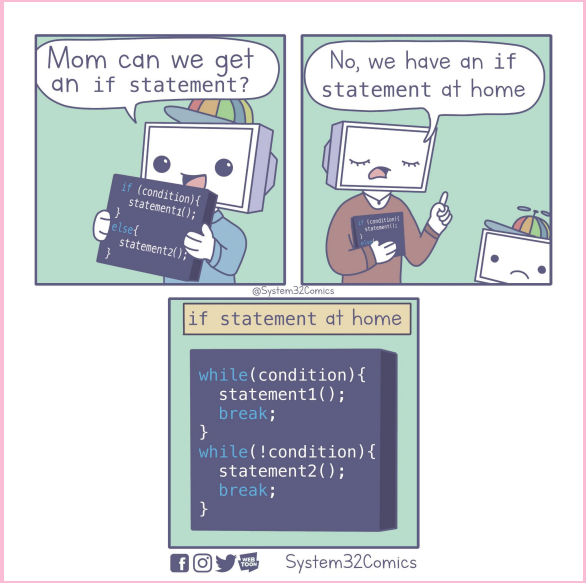


Hi! Welcome to 61A Discussion :)

We will begin at **5:10**!

Slides: **cs61a.bencuan.me**

Attendance: **go.cs61a.org/ben-disc**
(secret word ™ will be released shortly)

Announcements

- HW1 due tonight
- CSM signups opening soon!
- Office hours exist, you should try some!
  - oh.cs61a.org
  - Zoom and in-person both available
- Slides available at cs61a.bencuan.me

Agenda

- Attendance
- Expressions, values, statements
- Control (if, while)
- Environment diagrams!!

Attendance

(**go.cs61a.org/ben-disc**)

Today's Secret Word ™ : biscotti

Expressions, Values, Statements

Values vs Expressions vs Statements

Let’s get some examples!

Values	Expressions	Statements

Values vs Expressions vs Statements

More formal definitions:

Values: simple building blocks

- Evaluate to themselves (if you type into python, you get the same thing back)

Expressions: evaluate into values

- Kind of like a question (“what is 1+1?”) where the answer is a value (2)

Statements: change the flow of the program

- Does not directly evaluate into a value (main difference to expressions)

The purpose of an interpreter is to **turn expressions and statements into values.**

More on short circuiting

“Truthy” values:

- True, 1, -5000, ‘hello’

“Falsy” values:

- False, 0, ‘’ (empty string)

More on short circuiting

What will the following expression evaluate:

```
0 or 1 and 2 or 3
```

More on short circuiting

Short circuiting rules summary:

- Left to right
- Stop when you're 100% sure of the answer
- Return whatever's there

Control

- if statements: do something if condition is true

```
if condition:
    do stuff
elif other condition:
    do other stuff
else:
    do other other stuff
```

Q1: Case Conundrum

In this question, we will explore the difference between `if` and `elif`.

What is the result of evaluating the following code?

```
def special_case():
    x = 10
    if x > 0:
        x += 2
    elif x < 13:
        x += 3
    elif x % 2 == 1:
        x += 4
    return x

special_case()
```

```
def just_in_case():
    x = 10
    if x > 0:
        x += 2
    if x < 13:
        x += 3
    if x % 2 == 1:
        x += 4
    return x

just_in_case()
```

```
def case_in_point():
    x = 10
    if x > 0:
        return x + 2
    if x < 13:
        return x + 3
    if x % 2 == 1:
        return x + 4
    return x

case_in_point()
```

If Statements Summary

- If statements are processed from top to bottom
- Elif only runs when the if condition is false
- Return immediately makes function exit

Q2: Jacket Weather?

Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining.

Write a function that takes in the current temperature and a boolean value telling if it is raining. This function should return `True` if Alfonso will wear a jacket and `False` otherwise.

Try solving this problem using an `if` statement.

```
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
    """
    "*** YOUR CODE HERE ***"
```

Check data types!!!!!

Q3: If Function vs Statement

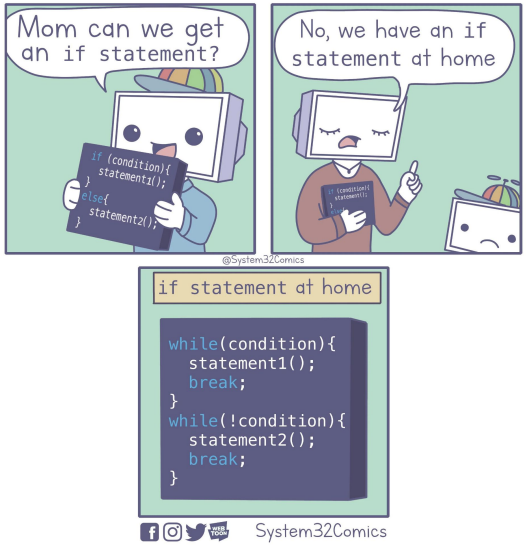
Now that we've learned about how `if` statements work, let's see if we can write a function that behaves the same as an `if` statement.

```
def if_function(condition, true_result, false_result):
    """Return true_result if condition is a true value, and
    false_result otherwise.

    >>> if_function(True, 2, 3)
    2
    >>> if_function(False, 2, 3)
    3
    >>> if_function(3==2, 'equal', 'not equal')
    'not equal'
    >>> if_function(3>2, 'bigger', 'smaller')
    'bigger'
    """
    if condition:
        return true_result
    else:
        return false_result
```

While

- while statements: do something until condition (boolean) is false

```
while condition:
    # stuff


# stuff to do after while
```



Q4: Is Prime?

Write a function that returns `True` if a positive integer `n` is a prime number and `False` otherwise.

A prime number n is a number that is not divisible by any numbers other than 1 and n itself. For example, 13 is prime, since it is only divisible by 1 and 13, but 14 is not, since it is divisible by 1, 2, 7, and 14.

> Hint: Use the `%` operator: `x % y` returns the remainder of `x` when divided by `y`.

```
def is_prime(n):
    """
    >>> is_prime(10)
    False
    >>> is_prime(7)
    True
    """
    "*** YOUR CODE HERE ***"
```

More hints:

- How do we check if x is divisible by y?
- Which numbers do we need to check? How do we go through them?

Environment Diagrams

...why are we doing this?

- Really, really good for understanding how computers interpret code
- Helpful for debugging programs
- Project 4: you'll make your own interpreter

make your own ED's

tutor.cs61a.org

Basic ED rules

- Boxes hold **values**
  - Evaluate all expressions fully!
- Arrows point to **functions/objects**

- **Function definition** vs **Function call**
  - Creating **new arrow** vs creating **new frame**

Q7 (example)

Q7: Assignment Diagram

Use these rules to draw an environment diagram for the assignment statements below:

```
x = 11 % 4
y = x
x **= 2
```

Q8. def diagram

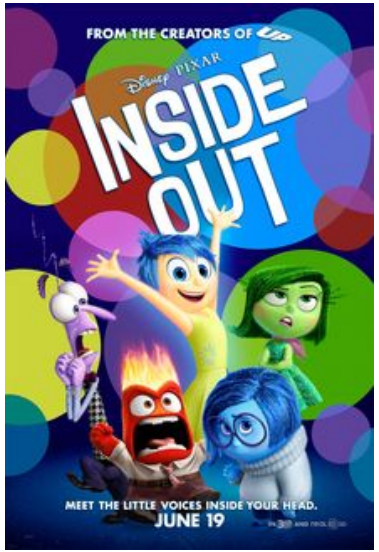
Q8: def Diagram

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

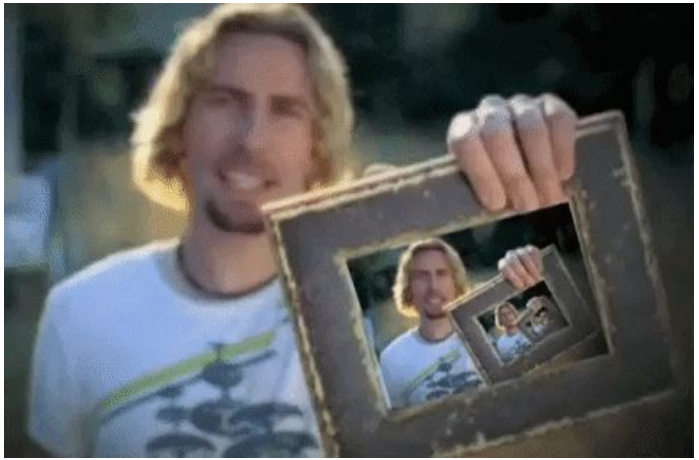
```
def double(x):
    return x * 2

def triple(x):
    return x * 3

hat = double
double = triple
```

The 6 steps of a Function Call

1. **Evaluate the operator**
2. **Evaluate the operands**
    a. Inside out, left to right
3. **Create a new frame**
4. **Copy parameters**
5. **Evaluate body**
6. **Return value**



Symbol lookup



Trying to find variable x?

1. Look in the current frame
2. Look in the parent frame
3. Look in the parent's parent frame
4. Look in the parent's parent's parent frame
5. ...
6. If there are no more frames, then error