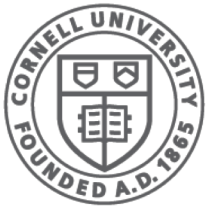
http://www.cs.cornell.edu/courses/cs1110/2019sp

Lecture 8: Conditionals & Control Flow

(Sections 5.1-5.7)

CS 1110

Introduction to Computing Using Python



Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Big Picture

Statements either affect **data** or **control**

- **DATA:** change the value in a box, create a box, *etc.*

  Examples:

  ```
  x = x + 1
  name = "Alex"
  ```

- **CONTROL:** tell python what line to execute next

  Examples:

  ```
  greet(name)
  if name == "Alex":
  ```
  ← today's Lecture

Conditionals: If-Statements

Format

if <*boolean-expression*>:
    <*statement*>
    …
    <*statement*>

Example

```
# is there a new high score?
if curr_score > high_score:
    high_score  = curr_score
    print("New high score!")
```

Execution:

if ⟨*Boolean-expression*⟩ is true, then execute all of the statements indented directly underneath (until first non-indented statement)

What are Boolean expressions?

Expressions that evaluate to a Boolean value.

```
is_student = True
is_senior = False
num_credits  = 25
```

Boolean variables:

```
if is_student:
   print("Hi student!")
```

Boolean operations:

```
if is_student and is_senior:
   print("Hi senior student!")
```

Comparison operations:

```
if num_credits > 24:
   print("Are you serious?")
```

What gets printed, Round 1

```
a = 0
print(a)
```

```
a = 0
a = a + 1
print(a)
```

```
a = 0
if a == 0:
    a = a + 1
print(a)
```

```
a = 0
if a == 1:
    a = a + 1
print(a)
```

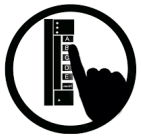
```
a = 0
if a == 0:
    a = a + 1
a = a + 1
print(a)
```

0	1	1	0	2

What gets printed? (Question)

```
a = 0
if a == 0:
    a = a + 1
if a == 0:
    a = a + 2
a = a + 1

print(a)
```

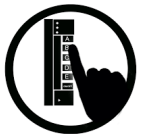
A: 0
B: 1
C: 2
D: 3
E: I do not know



What gets printed? (Solution)

```
a = 0                    Executed
if a == 0:               Executed
    a = a + 1            Executed
if a == 0:               Executed
    a = a + 2            Skipped
a = a + 1                Executed

print(a)
```

A: 0
B: 1
C: 2 **CORRECT**
D: 3
E: I do not know



Conditionals: If-Else-Statements

Format

```
if <boolean-expression>:
    <statement>
    ...
else:
    <statement>
    ...
```

Example

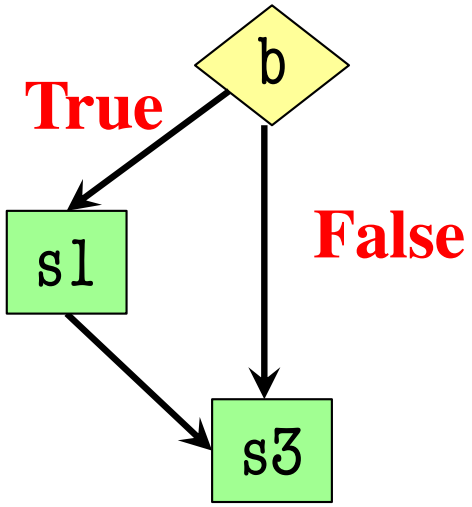
```
# who is the winner?
if score1 > score2:
    winner  = "Player 1"
else:
    winner = "Player 2"
```

Execution:

if ⟨*Boolean-expression*⟩ is true, then execute statements indented under **if**; otherwise execute the statements indented under **else**

Conditionals: "Control Flow" Statements

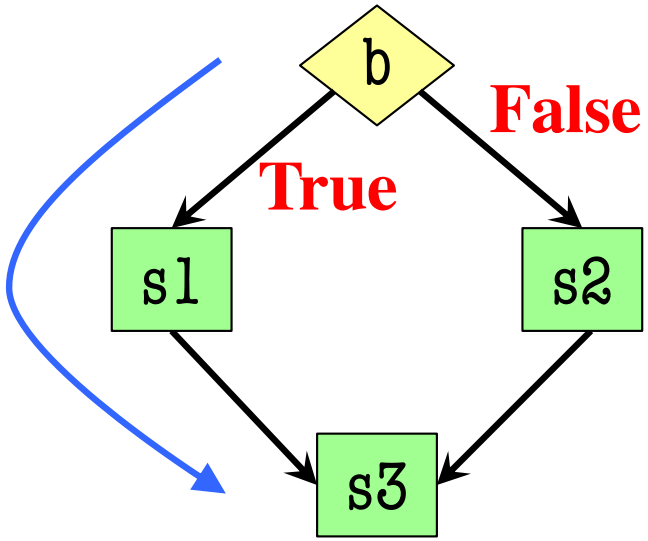
```
if b :
    s1   # statement
s3       # statement
```



Branch Point:
Evaluate & Choose

Statements:
Execute

```
if b :
    s1
else:
    s2
s3
```



Flow

Program only takes one path each execution (something will **not** be executed!)

What gets printed, Round 2

```
a = 0
if a == 0:
    a = a + 1
else:
    a = a + 2

print(a)
```

1

```
a = 0
if a == 1:
    a = a + 1
else:
    a = a + 2

print(a)
```

2

```
a = 0
if a == 1:
    a = a + 1
else:
    a = a + 2
a = a + 1

print(a)
```

3

```
a = 0
if a == 1:
    a = a + 1
else:
    a = a + 1
    a = a + 1
a = a + 1

print(a)
```

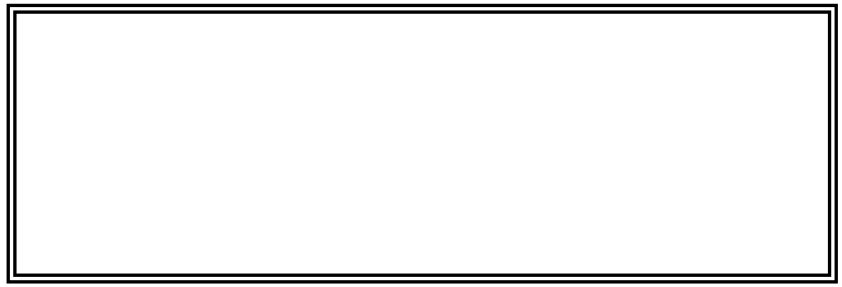
3

not in_love (0)

if determines which statement is executed next

Global Space

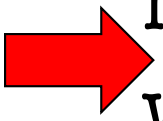
```
   def write_valentine(in_love):
1      if not in_love:
2          print("Let's be friends!")
3      print("Happy Valentine's Day.")

   name = input("Recipient Name: ")
   write_valentine(name=="Kilian")
```



not in_love (1)

if determines which statement is executed next

```
   def write_valentine(in_love):
1      if not in_love:
2          print("Let's be friends!")
3      print("Happy Valentine's Day.")

   name = ("Recipient Name: ")
   write_valentine(name=="Kilian")
```



Global Space

name	"Bob"

```
Recipient Name: Bob
```

not in_love (2)

if determines which statement is executed next

```
def write_valentine(in_love):
    if not in_love:
        print("Let's be friends!")
    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Global Space

name	"Bob"

Call Frame

write_valentine	1
in_love	False

Recipient Name: Bob

not in_love (3)

if determines which statement is executed next

```
def write_valentine(in_love):
1   if not in_love:
2       print("Let's be friends!")
3   print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Global Space

name	"Bob"

Call Frame

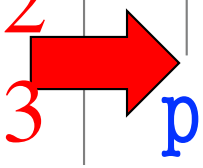
write_valentine	~~1~~ 2
in_love	False

```
Recipient Name: Bob
```

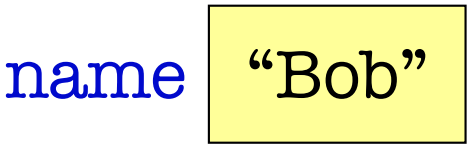
not in_love (4)

if determines which statement is executed next

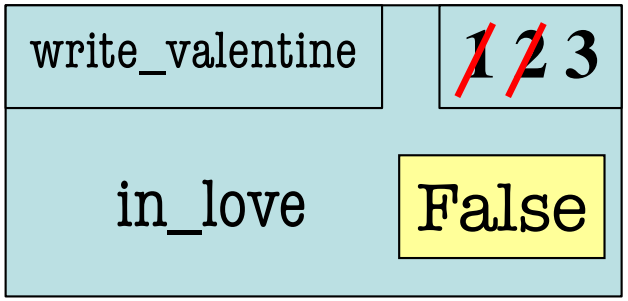
```
def write_valentine(in_love):
1    if not in_love:
2        print("Let's be friends!")
3    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```



Global Space



Call Frame




```
Recipient Name: Bob
Let's be friends!
```

not in_love (5)

if determines which statement is executed next

```
def write_valentine(in_love):
1    if not in_love:
2        print("Let's be friends!")
3    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Global Space

name	"Bob"

Call Frame

write_valentine	~~1 2 3~~
in_love	False
RETURN	None

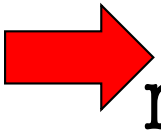
```
Recipient Name: Bob
Let's be friends!
Happy Valentine's Day.
```

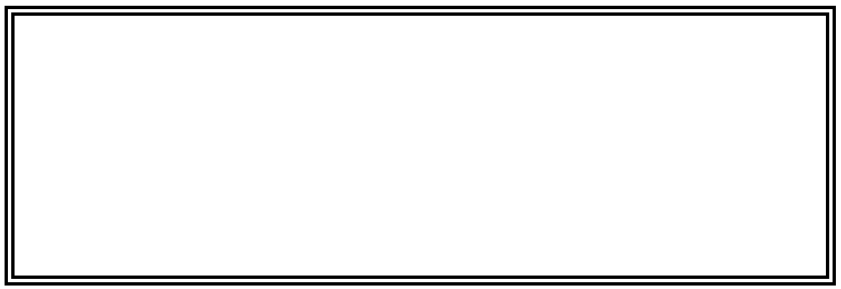
in_love (0)

if determines which statement is executed next

Global Space

```
def write_valentine(in_love):
1    if not in_love:
2        print("Let's be friends!")
3    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

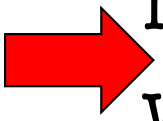




in_love (1)

if determines which statement is executed next

```
def write_valentine(in_love):
1   if not in_love:
2       print("Let's be friends!")
3   print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```



Global Space

name	"Kilian"

Recipient Name: Kilian

in_love (2)

if determines which statement is executed next

```
def write_valentine(in_love):
    if not in_love:
        print("Let's be friends!")
    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Global Space

name	"Kilian"

Call Frame

write_valentine	1
in_love	True

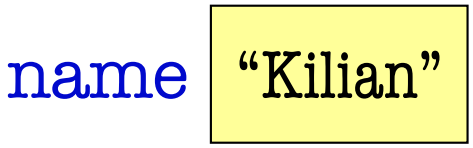
Recipient Name: Kilian

in_love (3)

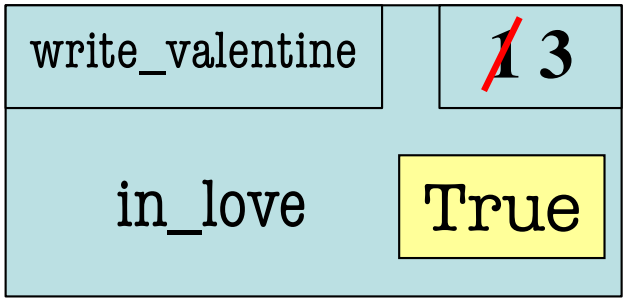
if determines which statement is executed next

```
def write_valentine(in_love):
1    if not in_love:
2        print("Let's be friends!")
3    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Global Space



Call Frame



Recipient Name: Kilian

in_love (4)

if determines which statement is executed next

```
def write_valentine(in_love):
1    if not in_love:
2        print("Let's be friends!")
3    print("Happy Valentine's Day.")

name = input("Recipient Name: ")
write_valentine(name=="Kilian")
```

Global Space

name	"Kilian"

Call Frame

write_valentine	~~1~~ ~~3~~
in_love	True
RETURN	None

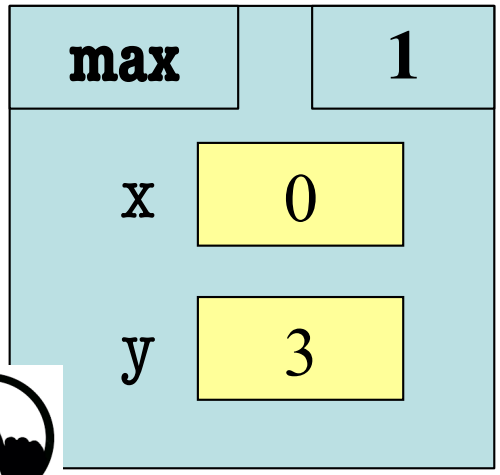
Recipient Name: Kilian

What does the call frame look like next? (Q)

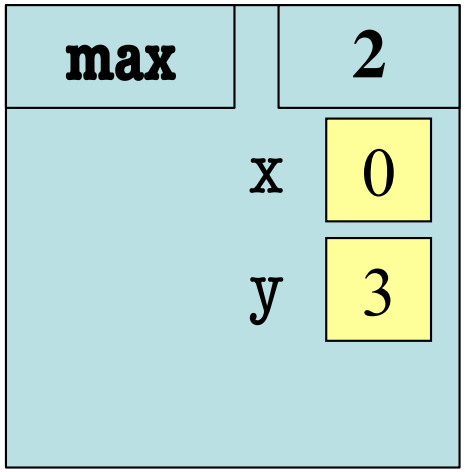
```
def max(x,y):
    if x > y:
        return x
    return y
```

max(0,3)

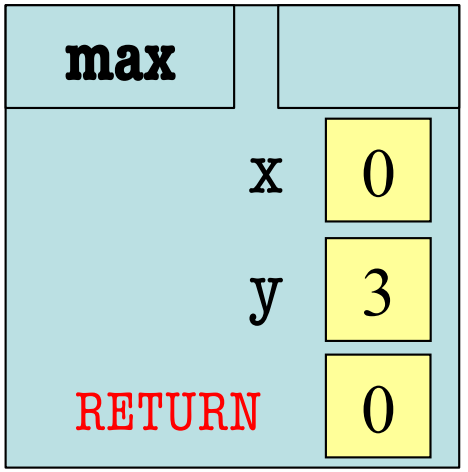
Current call frame:



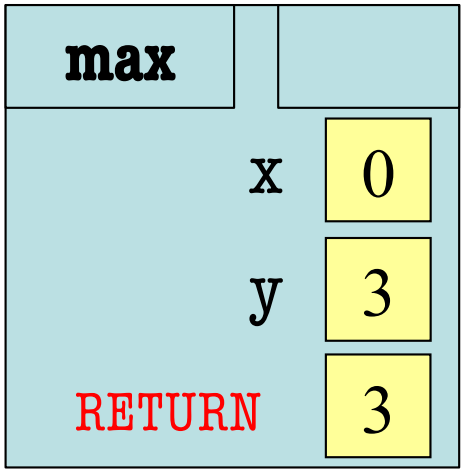
A:



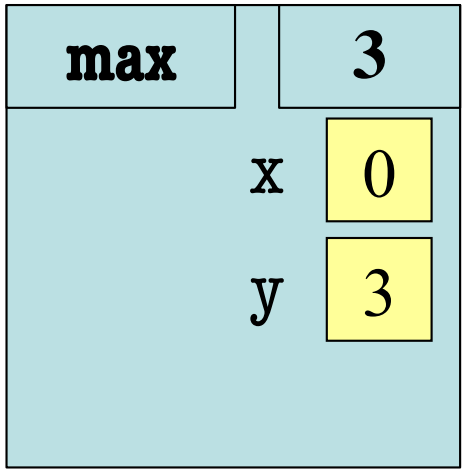
B:



C:



D:

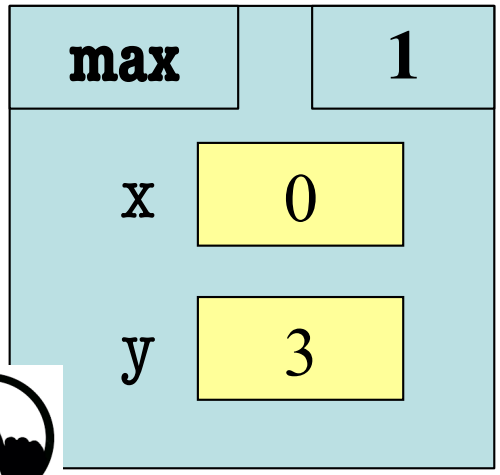


What does the call frame look like next? (A)

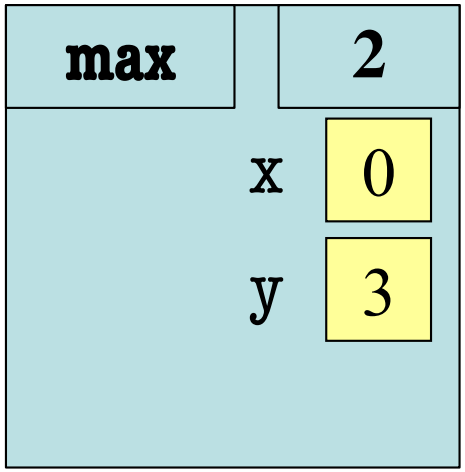
```
def max(x,y):
1   if x > y:
2       return x
3   return y
```

max(0,3)

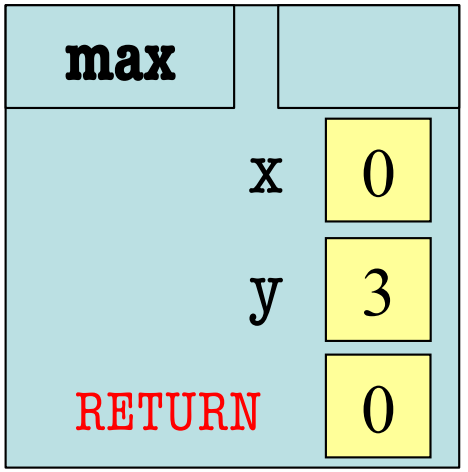
Current call frame:



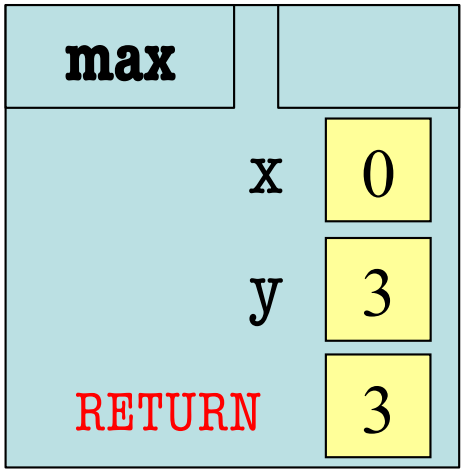
A:



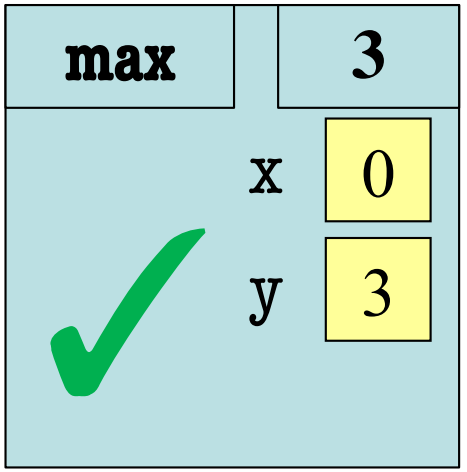
B:



C:



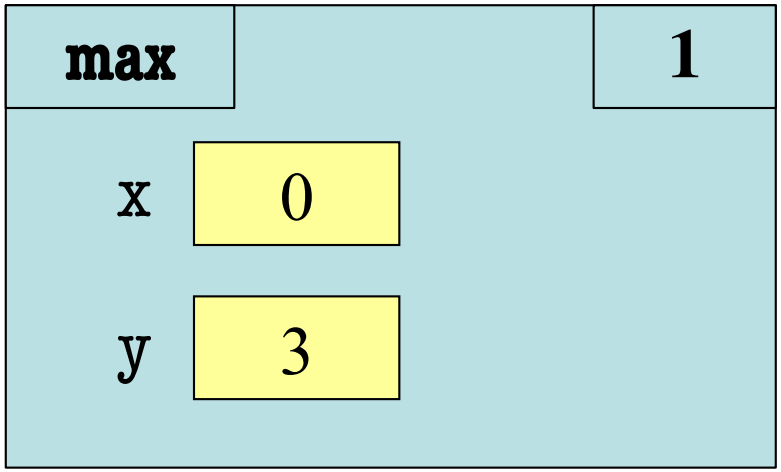
D:



Call Frame Explanation (1)

```
  def max(x,y):
1   if x > y:
2       return x
3   return y
```

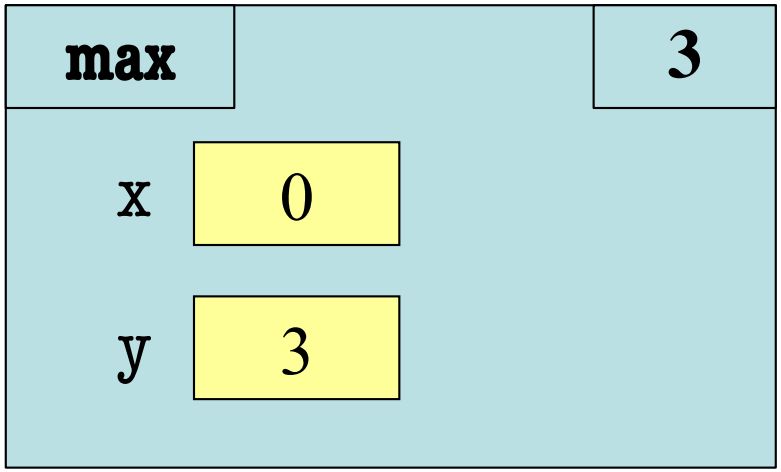
max(0,3):



Call Frame Explanation (2)

```
def max(x,y):
1   if x > y:
2       return x
3   return y
```

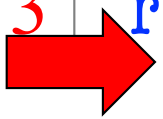
max(0,3):



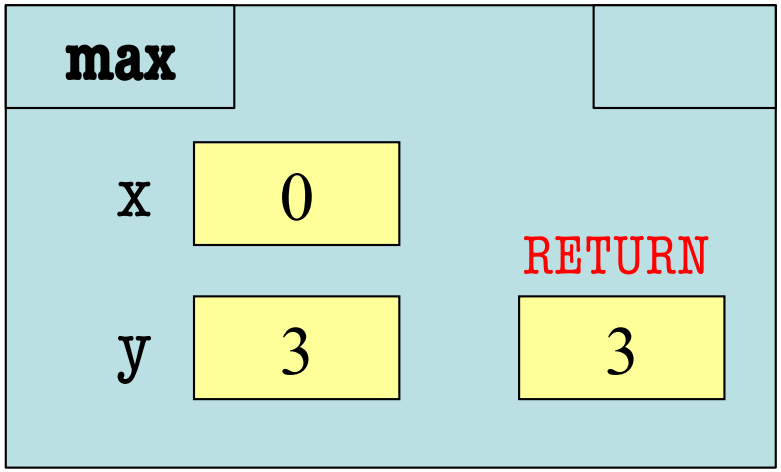
Skips line 2

Call Frame Explanation (3)

```
  def max(x,y):
1     if x > y:
2         return x
3     return y
```



max(0,3):



Program Flow and Variables

Variables created inside **if** continue to exist past **if**:

```
a = 0
if a == 0:
    b = a + 1
print(b)
```

…but are only created if the program actually executes that line of code

What gets printed, Round 3

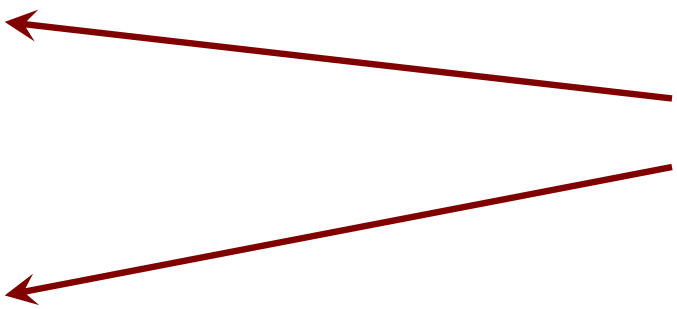
```
a = 0
if a == 0:
    b = 0
print(b)
```

```
a = 0
if a == 1:
    b = 0
print(b)
```

0	Error!

Program Flow and Variables

```
def zero_or_one(a):
    if a == 1:
        b = 1
    else:
        b = 0
    print(b)
```



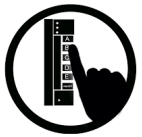
make sure that ALL **if** branches create the variable

Control Flow and Variables (Q1)

```
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger


maximum = max(3,0)
```

Value of maximum?

A: 3
B: 0
C: **Error!**
D: I do not know



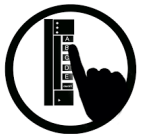
Control Flow and Variables (A1)

```
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger


maximum = max(3,0)
```

Value of maximum?

A: 3 **CORRECT**
B: 0
C: **Error!**
D: I do not know

- Local variables last until
  - They are deleted or
  - End of the function
- Even if defined inside **if**

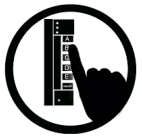


Control Flow and Variables (Q2)

```
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger


maximum = max(0,3)
```

Value of maximum?

A: 3
B: 0
C: **Error!**
D: I do not know



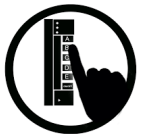
Control Flow and Variables (A2)

```
def max(x,y):
    """Returns: max of x, y"""
    # note: code has a bug!
    # check if x is larger
    if x > y:
        bigger = x
    return bigger

maximum = max(0,3)
```

Value of maximum?

A: 3
B: 0
C: **Error!** **CORRECT**
D: I do not know

- Variable existence depends on flow
- Generally terrible idea to refer to variables defined inside an **if** clause



Program Flow and Testing

Can use print statements to examine program flow

```
'before if'
'inside if x>y'
'after if'
```

x must have been greater than y

```
# Put max of x, y in z
print('before if')
if x > y:
    print('inside if x>y')
    z = x
else:
    print('inside else (x<=y)')
    z = y
print('after if')
```

"traces" or "breadcrumbs"

Traces (control) and Watches (data)

```
# Put max of x, y in z
print('before if')
if x > y:
    print('inside if x>y')
    z = x
    print('z = '+str(z))
else:
    print('inside else (x<=y)')
    z = y
    print('z = '+str(z))
print('after if')
```

TRACES

Trace **program flow**

What code is being executed?

Place them at the beginning of a block of code that might be skipped.

WATCHES

Watch **data** values

What is the value of a variable?

Place them after assignment statements.

Conditionals: If-Elif-Else-Statements

Format

```
if <Boolean expression>:
    <statement>
    ...
elif <Boolean expression>:
    <statement>
    ...
...
else:
    <statement>
    ...
```

Example

```
# Find the winner
if score1 > score2:
    winner  = "Player 1"
elif score2 > score1:
    winner  = "Player 2"
else:
    winner = "Players 1 and 2"
```

Conditionals: If-Elif-Else-Statements

Format

if *<Boolean expression>*:
    <statement>
    ...
elif *<Boolean expression>*:
    <statement>
    ...
...
else:
    <statement>
    ...

Notes on Use

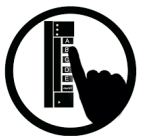
- No limit on number of elif
  - Must be between if, else
- else is optional
  - if-elif by itself is fine
- Booleans checked in order
  - Once Python finds a true *<Boolean-expression>*, skips over all the others
  - else means **all** are false

If-Elif-Else (Question)

```
a = 2

if a == 2:
    a = 3
elif a == 3:
    a = 4
print(a)
```

What gets printed?

A: 2
B: 3
C: 4
D: I do not know

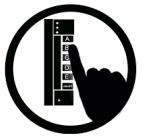


If-Elif-Else (Answer)

```
a = 2

if a == 2:
    a = 3
elif a == 3:
    a = 4
print(a)
```

What gets printed?

A: 2
B: 3 **CORRECT**
C: 4
D: I do not know



What gets printed, Round 4

```
a = 2
if a == 2:
    a = 3
elif a == 3:
    a = 4
print(a)
```

```
3
```

```
a = 2
if a == 2:
    a = 3
if a == 3:
    a = 4
print(a)
```

```
4
```

Nested Conditionals

```
def what_to_wear(raining, freezing):
    if raining:
        if freezing:
            print(”Wear a waterproof coat.”)
        else:
            print(”Bring an umbrella.")
    else:
        if freezing:
            print(”Wear a warm coat!")
        else:
            print(”A jacket will suffice.")
```