MORE ITERATION, NESTED LOOPS

Staying on the Same Line When Printing

- By default, print puts an invisible newline character at the end of whatever it prints.
  - Causes separate prints to print on different lines
- Example: What does this output?

```
for i in range(7):
    print(i * 5)
```

```
0
5
10
15
20
25
30
```

Staying on the Same Line When Printing

- To get separate prints to print on the same line, we can replace the newline with something else
- Example

```
for i in range(7):
    print(i * 5, end=‘ ’)
```

```
0 5 10 15 20 25 30
```

```
for i in range(7):
    print(i * 5, end=‘,’)
```

```
0,5,10,15,20,25,30,
```

Indefinite Loops

- Use an indefinite loop when the # of repetitions you need is
  - Not obvious or known
  - Impossible to determine before the loop begins, e.g.
    - Finding an element
    - Computing an estimate up to some error bound
    - Playing a game of rock, paper, scissors (as opposed to one round)

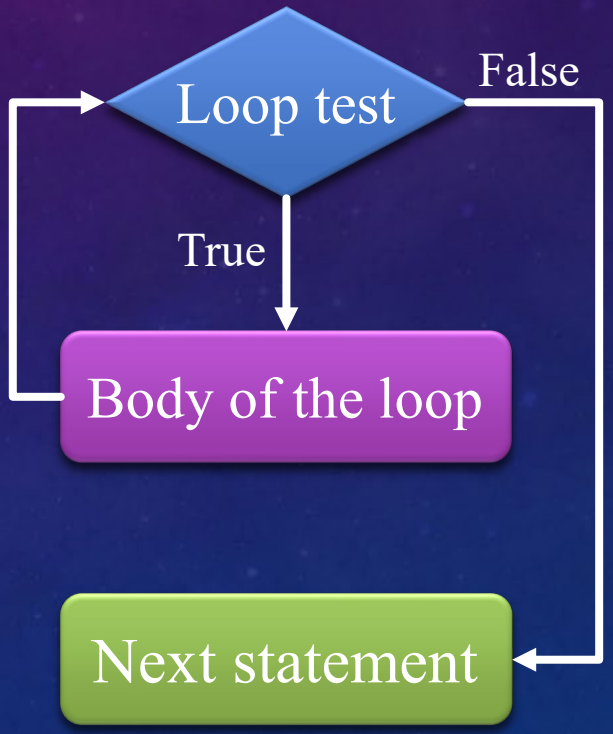
Indefinite Loops for Printing Multiples

- *while* loops are how you code indefinite loops in Python

```
n = 15
mult = 15
bound = 70
while mult < bound:
      print(mult, end=‘ ’)
      mult = mult + n
print(mult)
```

while Loops

```
while <loop test>:
    <body of the loop>
```



Steps

1. Evaluate the loop test (a boolean expression)
2. If it's True, execute the statements in the body, and go back to step 1
3. If it's False, skip the statements in the body and go to the statement after the loop

Tracing a *while* Loop

```
n = 15
mult = 15
bound = 70
while mult < bound:
      print(mult, end=‘ ’)
      mult = mult + n
print(mult)
```

← Prints everything on the same line with spaces in between

mult < bound	Output thus far	mult
		15
15 < 70 (True)	15	30
30 < 70 (True)	15 30	45
45 < 70 (True)	15 30 45	60
60 < 70 (True)	15 30 45 60	75
75 < 70 (False)		

Important

- In general, a while loop's test includes a key “loop variable”
- We need to update that loop variable in the body of the loop
- Failing to update it can produce an infinite loop!

```
n = 15
mult = 15
bound = 70
while mult < bound:
      print(mult, end=‘ ’)
      mult = mult + n
print(mult)
```

- What is the loop variable? mult
- Where is it updated? In the body of the loop

Factorial Using a *while* Loop

- We don't need an indefinite loop, but we can still use while

```
n = 4
result = 1
while n > 0:
    result *= n
    ___________
print(result)
```

← What do we need here?

Factorial Using a *while* Loop

- We don't need an indefinite loop, but we can still use while
- Let's trace when n = 4

```
n = 4
result = 1
while n > 0:
    result *= n
    n = n - 1
print(result)
```

n	n > 0	result
4		1
4	4 > 0 (True)	1 * 4 = 4
3	3 > 0 (True)	4 * 3 = 12
2	2 > 0 (True)	12 * 2 = 24
1	1 > 0 (True)	24 * 1 = 24
0	0 > 0 (False)	

Extreme Looping

- What does this code do?

```
print('It keeps')
while True:
    print('going and')
print('Phew! Done!')
```
← Never gets here!

- An infinite loop!

```
Use I, I or press ■ to stop a
program in Jupyter Notebook
```