ECS 32A - `while` Loops

Aaron Kaloti

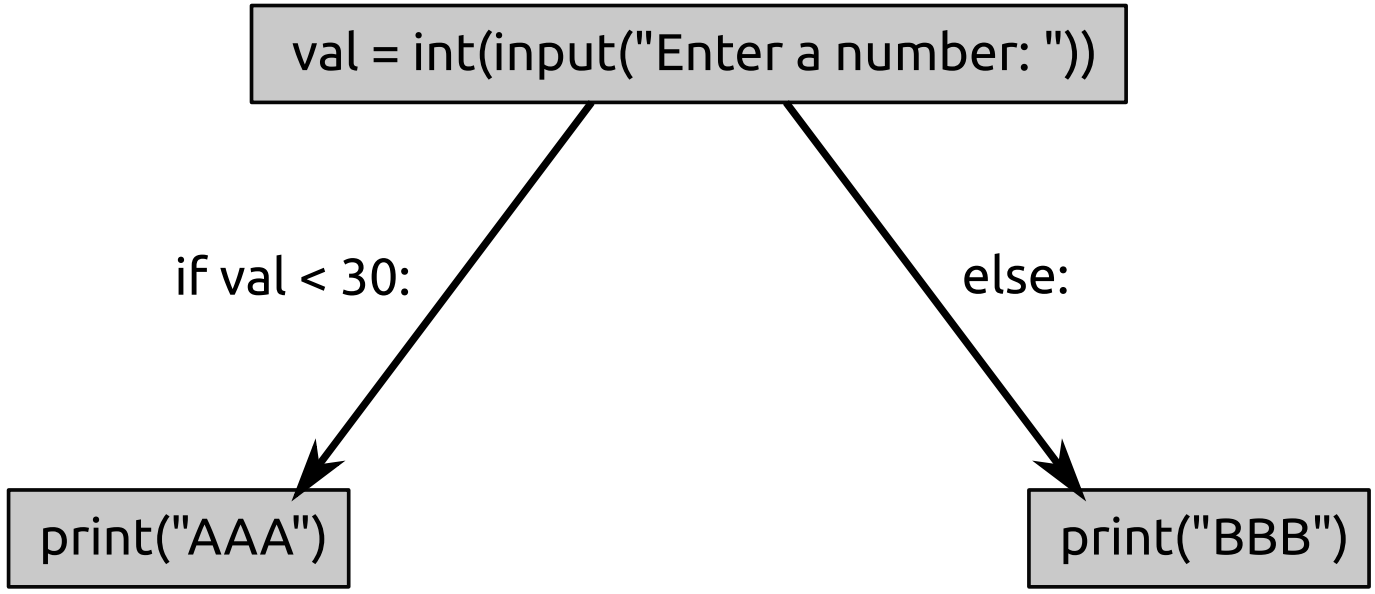
UC Davis - Summer Session #1 2020

UC DAVIS
COMPUTER SCIENCE

Review: Conditional Statements

Example #1

```
val = int(input("Enter a number: "))
if val < 30:
    print("AAA")
else:
    print("BBB")
```



Review: Conditional Statements

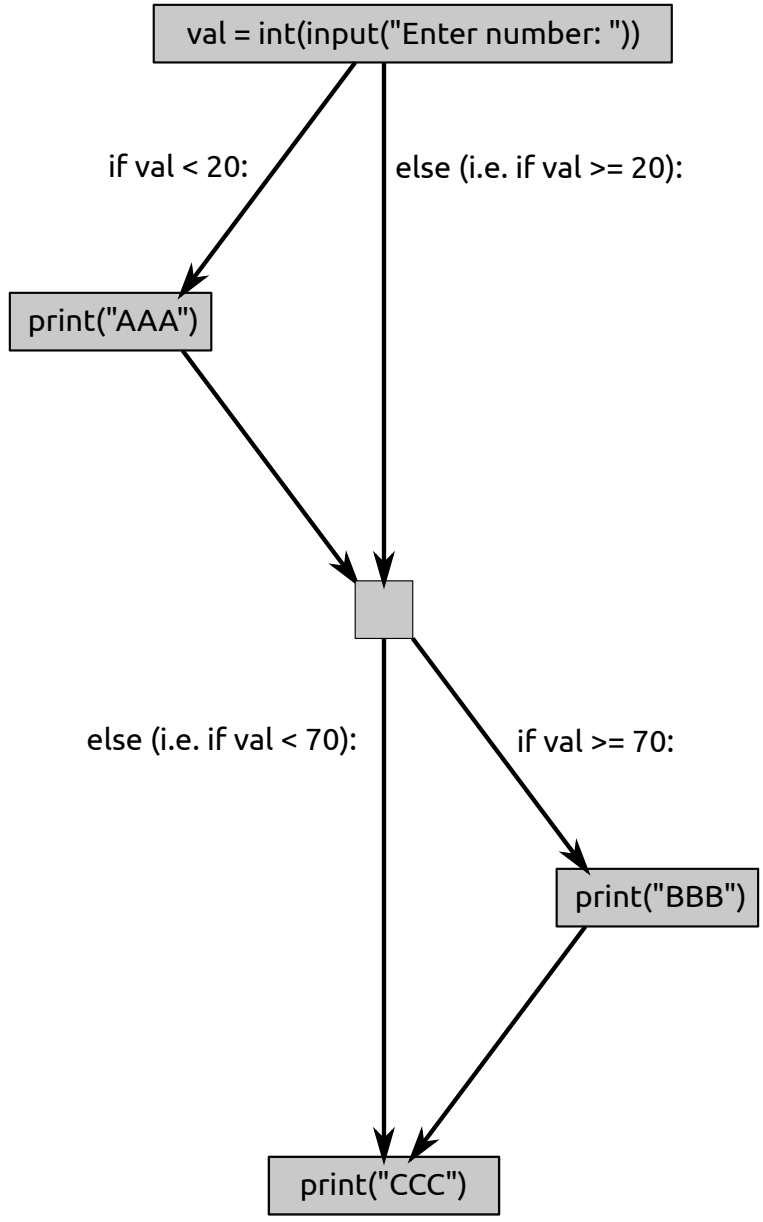
Example #2

```
val = int(input("Enter number: "))
if val < 20:
    print("AAA")
if val >= 70:
    print("BBB")
print("CCC")
```

```
Enter number: 19
AAA
CCC
```

```
Enter number: 75
BBB
CCC
```

```
Enter number: 50
CCC
```



Review: Conditional Statements

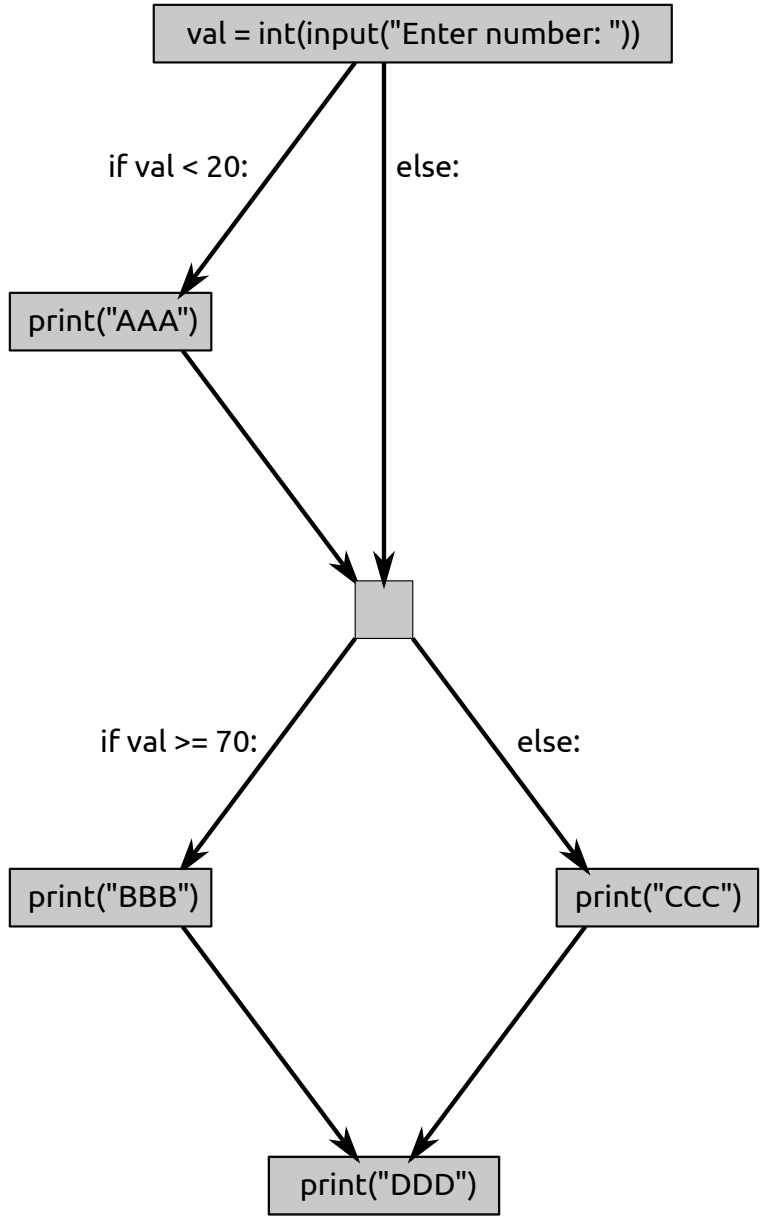
Example #3

```
val = int(input("Enter number: "))
if val < 20:
    print("AAA")
if val >= 70:
    print("BBB")
else:
    print("CCC")
print("DDD")
```

```
Enter number: 15
AAA
CCC
DDD
```

```
Enter number: 100
BBB
DDD
```

```
Enter number: 30
CCC
DDD
```



Motivating Example

```
choice = input("Are you cool? (Enter \"yes\" or \"no\".)\n")
if choice == "yes":
    print("Cool people don't say they're cool.")
elif choice == "no":
    print("Rip.")
```

```
Are you cool? (Enter "yes" or "no".)
yes
Cool people don't say they're cool.
```

```
Are you cool? (Enter "yes" or "no".)
no
Rip.
```

```
Are you cool? (Enter "yes" or "no".)
blah
```

- Instead of doing nothing if they enter invalid response, let's ask them again.

Motivating Example

```
choice = input("Are you cool? (Enter \"yes\" or \"no\".)\n")
if choice == "yes":
    print("Cool people don't say they're cool.")
elif choice == "no":
    print("Rip.")
else:
   choice = input("That response was invalid. Try again. (Enter \"yes\" or \"no\".)\n")
    if choice == "yes":
        print("Cool people don't say they're cool.")
    elif choice == "no":
        print("Rip.")
```

```
Are you cool? (Enter "yes" or "no".)
blah
That response was invalid. Try again. (Enter "yes" or "no".)
yes
Cool people don't say they're cool.
```

```
Are you cool? (Enter "yes" or "no".)
blah
That response was invalid. Try again. (Enter "yes" or "no".)
blah
```

- Should we ask them again?

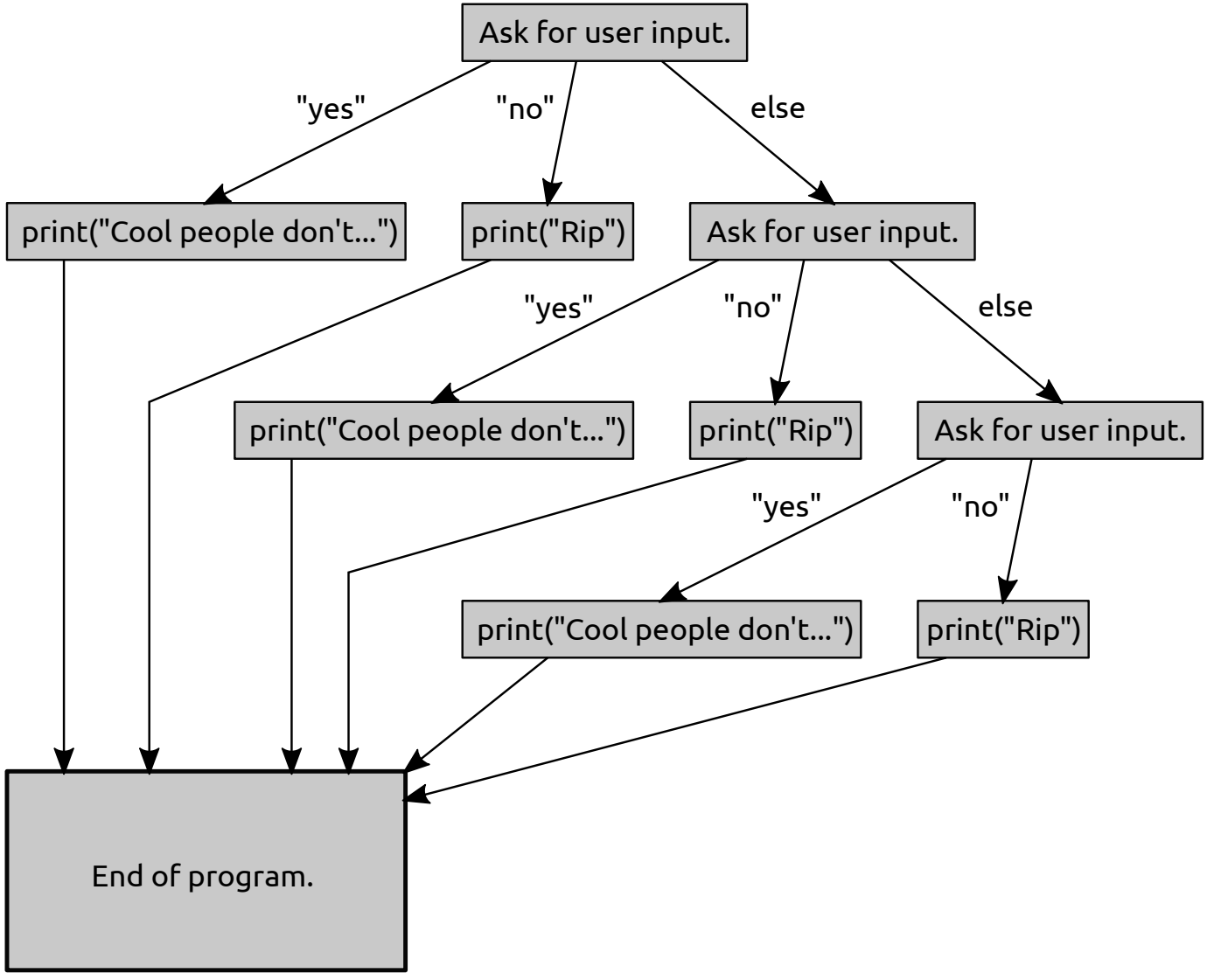
Motivating Example

```
choice = input("Are you cool? (Enter \"yes\" or \"no\".)\n")
if choice == "yes":
    print("Cool people don't say they're cool.")
elif choice == "no":
    print("Rip.")
else:
    choice = input("That response was invalid. Try again. (Enter \"yes\" or \"no\".)
\n")
    if choice == "yes":
        print("Cool people don't say they're cool.")
    elif choice == "no":
        print("Rip.")
    else:
        choice = input("That response was invalid. Try again. (Enter \"yes\" or \"no
\".)\n")
        if choice == "yes":
            print("Cool people don't say they're cool.")
        elif choice == "no":
            print("Rip.")
```

- This won't end...

Motivating Example

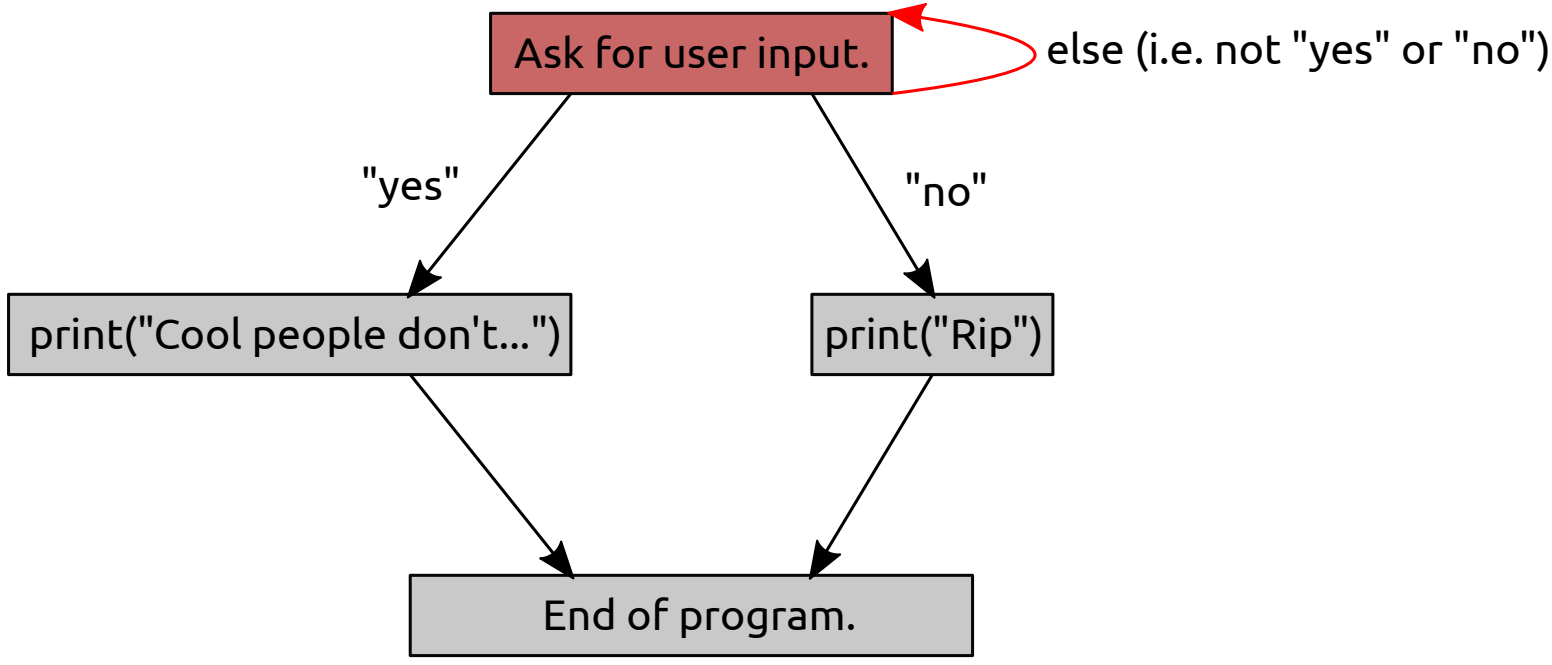
What We Have



Motivating Example

We Need Iteration

- **iterate**: repeat the same code until some condition is fulfilled
- We need some way of achieving this:



`while` Loops

Example: Fixing the Motivating Example

```
choice = ""
while choice != "yes" and choice != "no":
    choice = input("Are you cool? (Enter \"yes\" or \"no\".)\n")
if choice == "yes":
    print("Cool people don't say they're cool.")
elif choice == "no":
    print("Rip.")
```

```
Are you cool? (Enter "yes" or "no".)
blah
Are you cool? (Enter "yes" or "no".)
nah
Are you cool? (Enter "yes" or "no".)
NO
Are you cool? (Enter "yes" or "no".)
no
Rip.
```

`while` Loops

Structure

Syntactical

```
while condition:
    body
```

- `body` will keep being executed until `condition` is `False`.

Practical

- Necessary components.
    1. Something that keeps track of state.
    2. Condition that checks the state.
    3. Something that changes the state.
        - Avoids infinite loop.

Example: Print from 3 to 8

```
x = 3
while x <= 8:
    print(x)
    x += 1
```

```
3
4
5
6
7
8
```

Example: Print from 3 to 8

A Closer Look

```
x = 3
while x <= 8:
    print(x)
    x += 1
```

- (From earlier slide...) Necessary components.
  1. Something that keeps track of state.
     - `x` variable
  2. Condition that checks the state.
     - `x <= 8`
  3. Something that changes the state.
     - `x += 1`

Example: Print from 7 to 24

```
y = 7
while y <= 24:
    print(y)
    y += 1
```

```
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

Example: Print from 6 to User-Chosen Integer

```
end_bound = int(input("Enter end bound: "))
counter = 6
while counter <= end_bound:
    print(counter)
    counter += 1
# Random ending message...
print("After while loop!")
```

```
Enter end bound: 12
6
7
8
9
10
11
12
After while loop!
```

```
Enter end bound: 8
6
7
8
After while loop!
```

```
Enter end bound: -3
After while loop!
```

Example: Complicated Condition

Prompt

- Write a program that asks the user for three integers -- a, b, and c -- and keeps prompting the user for these three integers until the sum of the first two integers equals the product of the last two integers.

Solution

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))
c = int(input("Enter c: "))
while a + b != b * c:
    a = int(input("Enter a: "))
    b = int(input("Enter b: "))
    c = int(input("Enter c: "))
print("{} + {} == {} * {}".format(a, b, b, c))
```

Example Run

```
Enter a: 2
Enter b: 3
Enter c: 5
Enter a: 7
Enter b: 1
Enter c: 7
Enter a: -2
Enter b: 1
Enter c: -1
-2 + 1 == 1 * -1
```

Example: Sum of Integers from 0 to N

Prompt

- Write a program that prompts the user for an integer N and prints the sum of all integers from 1 up to and including N, i.e. prints $\sum_{i=1}^{N} i$. If the user enters a negative integer, then an error message should be printed instead.

Solution

```
n = int(input("Enter N: "))
if n < 0:
    print("Error: n can't be negative.")
else:
    i = 1
    sum = 0
    while i <= n:
        sum += i
        i += 1
    print("The sum is {}.".format(sum))
```

Example Runs

```
Enter N: 3
The sum is 6.
```

```
Enter N: 5
The sum is 15.
```

```
Enter N: -2
Error: n can't be negative.
```

Infinite Loops

- **infinite loop**: loop that never ends
- The simplest one:

```
while True:
    pass
```

- A noisy one:

```
while True:
    print("NOISE")
```

- Doesn't need to use `while True`:

```
num = 2
while num == 2:
    print("MORE NOISE")
```

Timing of Condition Evaluation

- Note that the condition is *only* evaluated *when we come back to it*.
- Therefore, this loop will never end.

```
a = 5
while a == 5:
    a = 2
    a = 5
```

Timing of Condition Evaluation

- Even though `a` no longer equals 5 after this line:

```
a = 5
while a == 5:
    a = 2
    a = 5
```

- `a` becomes 5 again after this line:

```
a = 5
while a == 5:
    a = 2
    a = 5
```

- Thus, when the condition is reevaluated, `a == 5` is `True`, and the looping continues.

```
a = 5
while a == 5:
    a = 2
    a = 5
```

Can Decrement Counter Variable

Example

```
x = 5
while x > 0:
    print("x is", x)
    x -= 1
print("Lift off!")
```

```
x is 5
x is 4
x is 3
x is 2
x is 1
Lift off!
```

Example: Loop Until User Enters 3

Prompt

- As the title states, write a program that keeps prompting the user to enter an integer until they enter 3, after which the program should print some special message and then end.

Example: Loop Until User Enters 3

Solution

```
n = 0
# n = int(input("Enter: "))
while n != 3:
    n = int(input("Enter: "))
print("Good bye!")
```

loop_until_3.py

Example: Print Even Numbers

Prompt

- Write a program that asks the user for an integer N and prints all even numbers x such that $0 \leq x \leq N$, in ascending order.
  - For example, if the user enters $N = 7$, then the program should print 0, 2, 4, and 6.
  - Do this program in two different ways: one way that uses the modulo (`%`) operator and one way that doesn't.

Example: Print Even Numbers

Solution

Without Modulo Operator

```
n = int(input("Enter N: "))
i = 0
while i <= n:
    print(i)
    i += 2
```

print_evens_no_modulo.py

With Modulo Operator

```
n = int(input("Enter N: "))
i = 0
while i <= n:
    if i % 2 == 0: # if i is even
        print(i)
    i += 1
```

print_evens_modulo.py

Example: Print Odd Numbers in Reverse

Prompt

- Rewrite the program from the previous example to print ***odd*** numbers instead of even numbers and to print in ***descending*** order instead of ascending order.
  - For example, if the user enters $N = 13$, then the program should print 13, 11, 9, 7, 5, 3, and 1.

Example: Print Odd Numbers in Reverse

Solution #1

```
n = int(input("Enter N: "))
while n > 0:
    if n % 2 == 1:  # if n is odd
        print(n)
    n -= 1
```

Solution #2

```
n = int(input("Enter N: "))
if n % 2 == 0:  # if n is even
    n -= 1
while n > 0:
    print(n)
    n -= 2
```

Example: Product of Range

Prompt

- Write a program that prompts the user for two integers M and N and prints the product of all integers between M and N (exclusive). In other words, the program should print $\prod_{i=M+1}^{N-1} i$.

Example: Product of Range

Solution #1

```
M = int(input("Enter M: "))
N = int(input("Enter N: "))
i = M + 1
product = 1
while i < N:  # could be <= N - 1
    product *= i  # product = product * i
    i += 1
print("The product is", product)
```

product_of_range.py

Solution #2

- During the 07/01 lecture, I talked about (and recommended against) incrementing/decrementing the loop counter variable at the beginning of the loop's body instead of at the end.

```
M = int(input("Enter M: "))
N = int(input("Enter N: "))
i = M
product = 1
while i < N - 1:  # could be <= N - 1
    i += 1
    product *= i  # product = product * i
print("The product is", product)
```

Example: More Math

Prompt

- Write a program that prompts the user for an integer N and outputs the value $\sum_{i=1}^{N} [i \cdot (i+1)]$.
  - Examples (of the summation):
    - $N = 4$: $\sum_{i=1}^{4} [i \cdot (i+1)] = (1 \cdot 2) + (2 \cdot 3) + (3 \cdot 4) + (4 \cdot 5) = 2 + 6 + 12 + 20 = 40$.
    - $N = 6$: $\sum_{i=1}^{7} [i \cdot (i+1)] = (1 \cdot 2) + (2 \cdot 3) + (3 \cdot 4) + (4 \cdot 5) + (5 \cdot 6) + (6 \cdot 7)$
      $= 2 + 6 + 12 + 20 + 30 + 42 = 112$.

Solution

```
N = int(input("Enter N: "))
i = 1
result = 0
while i <= N:
    result += i * (i + 1)
    i += 1
print("The result is", result)
```

`break`

- Can use `break` to break out of middle of loop.
- Immediately ends enclosing loop when encountered.
- Not often used; sometimes arguably more convenient than not using.

Without `break`:

```
i = 0
while i < 5:
    print(i)
    i += 1
```

With `break`:

```
i = 0
while i < 5:
    if i == 3:
        break
    print(i)
    i += 1
```

break1.py

- Output:

```
0
1
2
3
4
```

```
0
1
2
```

`break`

Nested Conditional Statements

- `break` breaks out of a *loop*; doesn't matter how many nested conditional statements it's in.

```
i = 0
while i < 5:
    if i == 0:
        if i == 0:
            if i == 0:
                break  # escape the loop
    print(i)  # this statement is never triggered
    i += 1    # this statement is never triggered
print("Escaped the loop!")
```

continue

- Jumps back to the condition from the middle of a loop.
- Rarely used.

```
i = 0
while i <= 10:
    print(i)
    if i % 2 == 0:  # if i is even
        i += 3
        continue
    i += 1
```

```
0
3
4
7
8
```

Example: Toggling a Variable

Prompt

- Write a program that keeps prompting the user to enter an integer until they enter 0. The program should then tell the user the sum of every *other* integer that the user has entered. For example, if the user enters 7, -3, 4, 2, 8, and -1, then the program should output the sum $7 + 4 + 8 = 19$.

Solution #1

```
sum = 0
val = 1  # initial value can't be 0
ignoreValue = False  # True if should ignore current value
while val != 0:
    val = int(input("Enter integer: "))
    if ignoreValue == False:  # if not ignoring this value
        sum += val
        # Since we did NOT ignore this value, we should ignore
        # the next value.
        ignoreValue = True
    elif ignoreValue == True:  # could also be else:
        # Since we DID ignore this value, we should NOT ignore
        # the next one.
        ignoreValue = False
print("The sum is", sum)
```

Example: Toggling a Variable

Solution #2

```
sum = 0
val = 1  # initial value can't be 0
ignoreValue = False  # True if should ignore current value
while val != 0:
    val = int(input("Enter integer: "))
    if not ignoreValue:  # if not ignoring this value
        sum += val
        # Since we did NOT ignore this value, we should ignore
        # the next value.
        ignoreValue = True
    elif ignoreValue:  # could also be else:
        # Since we DID ignore this value, we should NOT ignore
        # the next one.
        ignoreValue = False
print("The sum is", sum)
```

Example: Toggling a Variable

Solution #3: No `elif` Case[1]

```
sum = 0
val = 1  # initial value can't be 0
ignoreValue = False  # True if should ignore current value
while val != 0:
    val = int(input("Enter integer: "))
    if not ignoreValue:
        sum += val
    # Change ignoreValue to its opposite boolean value
    # (i.e. literally toggle the value of ignoreValue).
    ignoreValue = not ignoreValue
print("The sum is", sum)
```

1. I wrote this version after the lecture, just to give you an idea of how far the number of lines could be minimized.

Example: Tracking Multiple Things

Prompt

- Write a program that keeps prompting the user to enter an integer until they enter 0. As this occurs, the program should keep track of the sum of all integers entered by the user *and* the product of all integers entered by the user (ignoring the final 0). After the user enters 0, the program should say which of the sum or product is greater and end.

Solution

```
sum = 0
product = 1
val = 1  # can't init to 0
while val != 0:
    val = int(input("Enter integer: "))
    if val != 0:
        sum += val
        product *= val
if sum > product:
    print("sum {} exceeds product {}".format(sum, product))
elif product > sum:
    print("product {} exceeds sum {}".format(product, sum))
else:
    print("sum and product are both {}".format(sum))
```

Example: Highest So Far

- Let's try an example where the user could enter an integer *or* a word.

Prompt

- Write a program that keeps prompting the user to enter an integer until the user enters the string "Done". At this point, the highest integer entered by the user should be printed.
  - *Hint*: In order to check if the user entered "Done", you must not immediately convert the input to an integer (i.e. you must not immediately use `int(...)`).
  - Simplifying assumptions:
    - You may assume the user will always enter at least one integer.
    - You may assume that any integer entered by the user will be nonnegative.
  - **Two versions**: Write this program so that it uses `break` *and* write another version that does not use `break`.

Example: Highest So Far

Solution #1

```
user_input = ""
highest = -1
while user_input != "Done":
    user_input = input("Enter \"Done\" or integer: ")
    if user_input != "Done":
        int_value = int(user_input)
        # If this user input is greater than the highest
        # we've seen so far, update the highest.
        if int_value > highest:
            highest = int_value
if highest == -1:
    print("You entered no values!")
else:  # if highest:
    print("Highest value seen is {}".format(highest))
```

Example: Highest So Far

Solution #2: Version with `break`

```
user_input = ""
highest = 0
while True:
    user_input = input("Enter \"Done\" or integer: ")
    if user_input == "Done":
        break
    int_value = int(user_input)
    # If this user input is greater than the highest
    # we've seen so far, update the highest.
    if int_value > highest:
        highest = int_value
print("Highest value seen is {}".format(highest))
```

Example: Highest So Far

Solution #3: Version that Allows Negative Values

```
user_input = ""
highest = None
while user_input != "Done":
    user_input = input("Enter \"Done\" or integer: ")
    if user_input != "Done":
        int_value = int(user_input)
        # If this user input is greater than the highest
        # we've seen so far OR if it's the first input,
        # update the highest.
        if highest == None or int_value > highest:
            highest = int_value
if not highest:  # or if highest == None
    print("You entered no values!")
else:  # if highest:
    print("Highest value seen is {}".format(highest))
```