Repetition

- For repetition to occur, there needs to be a loop body containing the steps to be repeated
- Question:

  Were any steps repeated when the problem was solved?

  Yes (a repetitive loop is needed)
  - Which steps were repeated? (statements to include in the loop body)
  - Did we know in advance how many times to repeat the steps?
  - No
    - How do we know how long to keep repeating the steps?

Repetition

- We know how many times to repeat the steps
  - Use counting loop since we know exactly the number of loop repetitions needed to solve the problem
- We do not know in advance how many times to repeat the steps
  - Use conditional loop
    - General conditional
    - *Sentinel-controlled
    - Input validation
    - Endfile-controlled

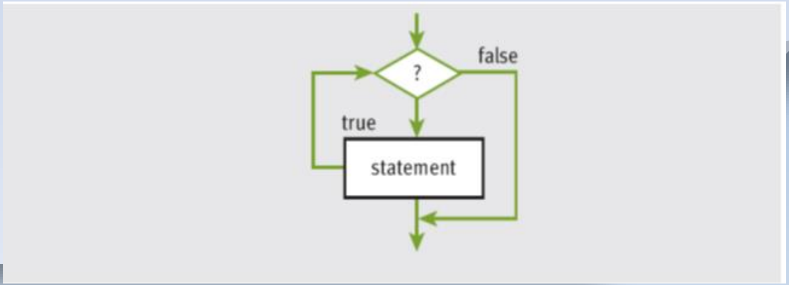
A sentinel value (also referred to as a flag value, trip value, rogue value, signal value, or dummy data) is a special value in the context of an algorithm which uses its presence as a condition of termination, typically in a loop

The *while* Loop

- Executes a block of code repeatedly
- *while* loop repeatedly executes an indented block of statements
  - As long as a certain condition is met
- Form

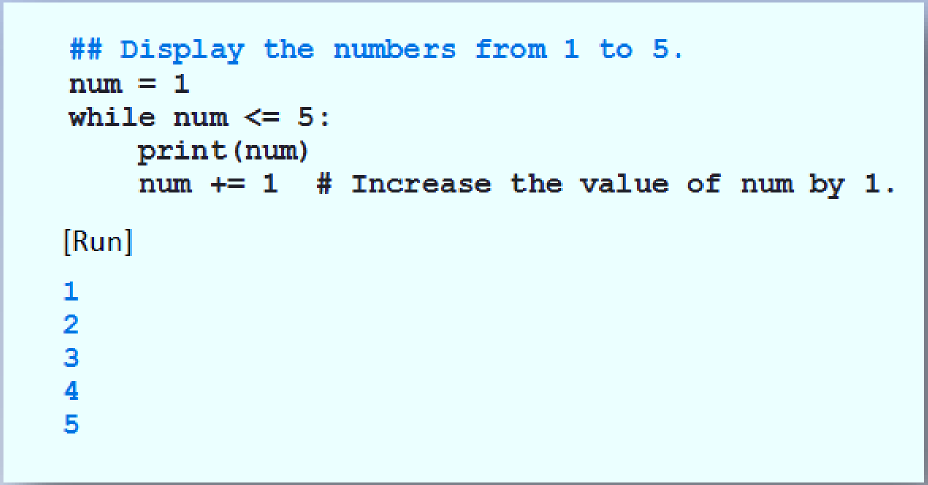
```
while condition:
     indented block of statements
```

- Continuation condition is a boolean expression



The *while* Loop

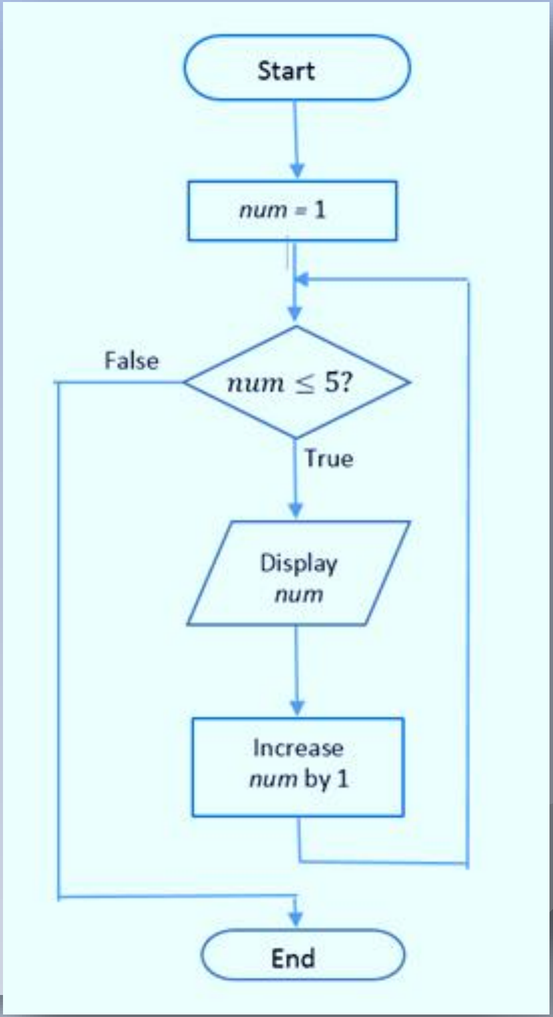
- Example: Program displays 1 – 5, after loop terminates, *num* will be 6



```
## Display the numbers from 1 to 5.
num = 1
while num <= 5:
    print(num)
    num += 1  # Increase the value of num by 1.
```

[Run]

```
1
2
3
4
5
```



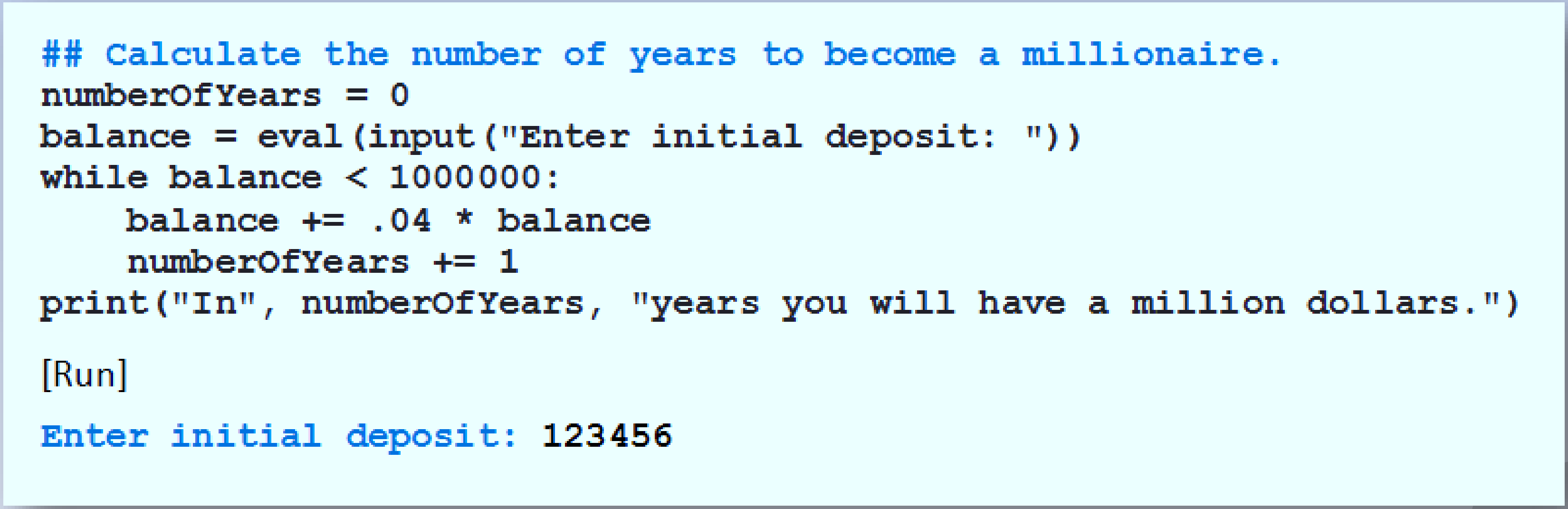
Counting Loops with ***while***

- Write a program fragment that produces this output:

```
0       1
1       2
2       4
3       8
4       16
5       32
6       64
```

The *while* Loop

- Example: Determines when bank deposit reaches one million dollars (interest per year is 4% on balance)



```
## Calculate the number of years to become a millionaire.
numberOfYears = 0
balance = eval(input("Enter initial deposit: "))
while balance < 1000000:
    balance += .04 * balance
    numberOfYears += 1
print("In", numberOfYears, "years you will have a million dollars.")
```

[Run]

```
Enter initial deposit: 123456
```

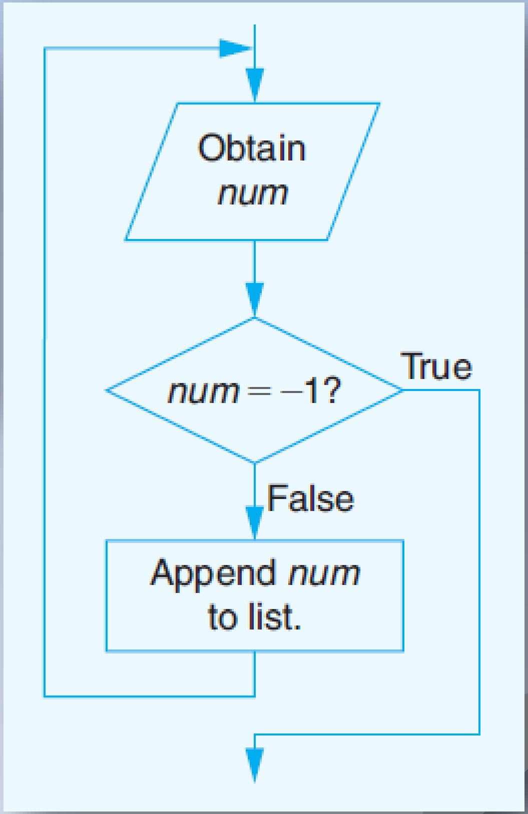
The *break* Statement
(not usually recommended!)

- Causes an exit from anywhere in the body of a loop
- When **`break`** is executed
  - Loop immediately terminates
- Break statements usually occur in *if* statements

The *break* Statement

- Example: Program uses *break* to avoid two input statements.

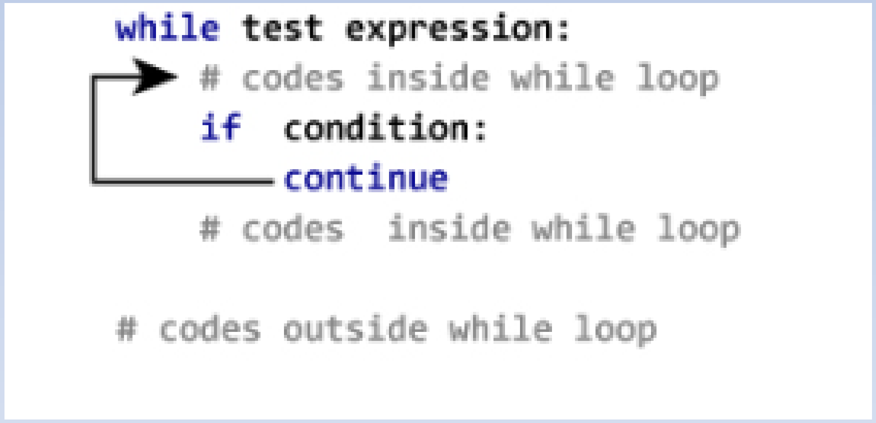
```
## Obtain list of numbers.
list1 = []
while True:
    num = eval(input("Enter a nonnegative number: "))
    if num == -1:
        break    # Immediately terminate the loop.
    list1.append(num)
```



The *continue* Statement

(definitely not recommended!)

- When **`continue`** executed in a while loop
  - Current iteration of the loop terminates
  - Execution returns to the loop's header
- Usually appear inside *if* statements.

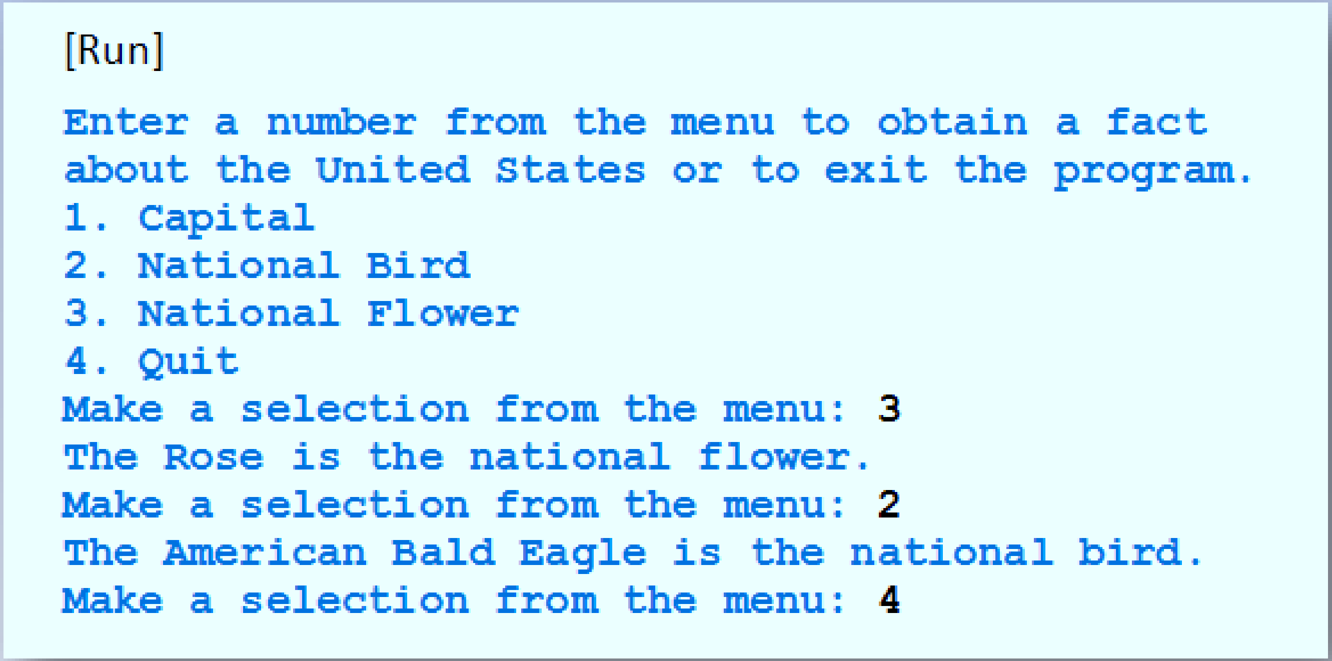


```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

Creating a Menu

- Example: Uses a menu to obtain facts about the United States.

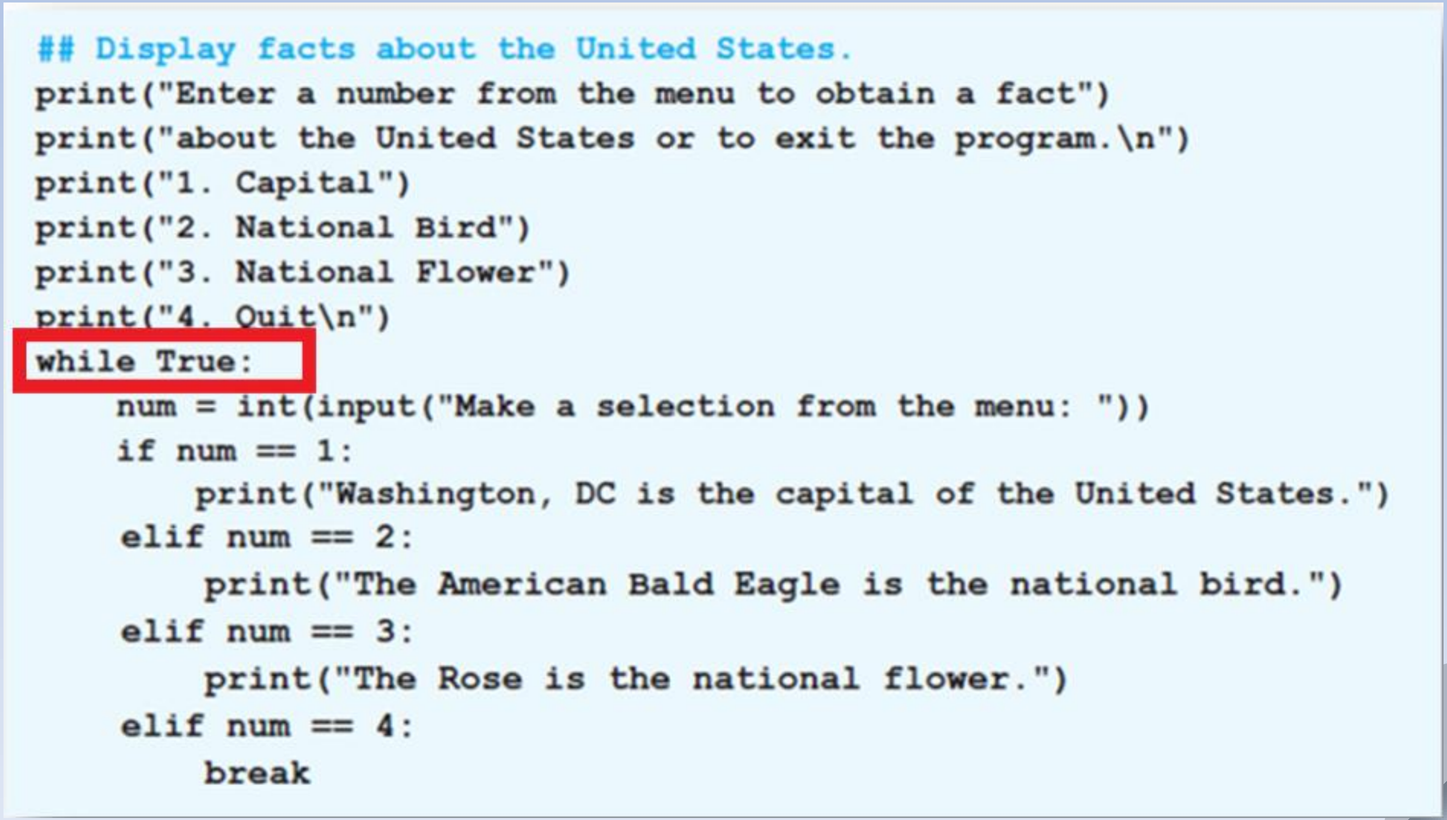


[Run]

```
Enter a number from the menu to obtain a fact
about the United States or to exit the program.
1. Capital
2. National Bird
3. National Flower
4. Quit
Make a selection from the menu: 3
The Rose is the national flower.
Make a selection from the menu: 2
The American Bald Eagle is the national bird.
Make a selection from the menu: 4
```

Creating a Menu

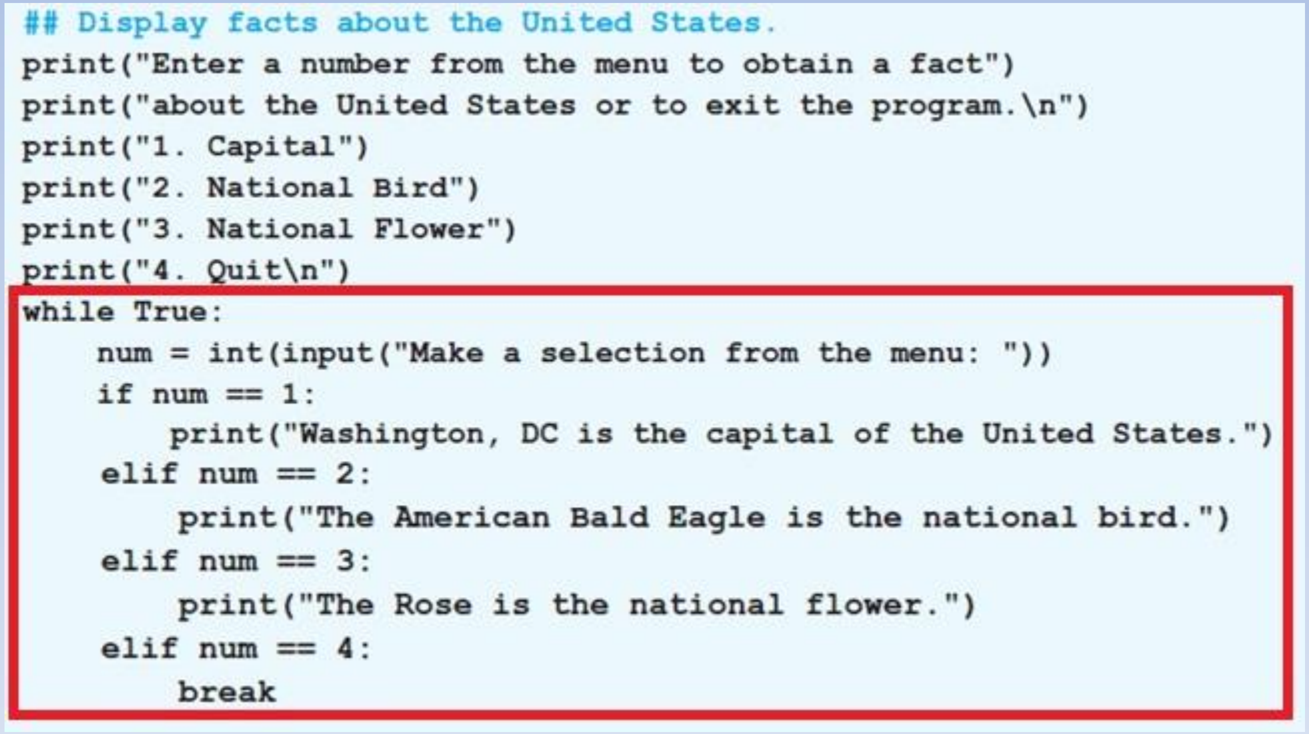
- What is typically ***not*** recommended!
- Example: Uses a menu to obtain facts about the United States using a break statement



```
## Display facts about the United States.
print("Enter a number from the menu to obtain a fact")
print("about the United States or to exit the program.\n")
print("1. Capital")
print("2. National Bird")
print("3. National Flower")
print("4. Quit\n")
while True:
    num = int(input("Make a selection from the menu: "))
    if num == 1:
        print("Washington, DC is the capital of the United States.")
    elif num == 2:
        print("The American Bald Eagle is the national bird.")
    elif num == 3:
        print("The Rose is the national flower.")
    elif num == 4:
        break
```

Creating a Menu

- Can you do this without using a break statement?



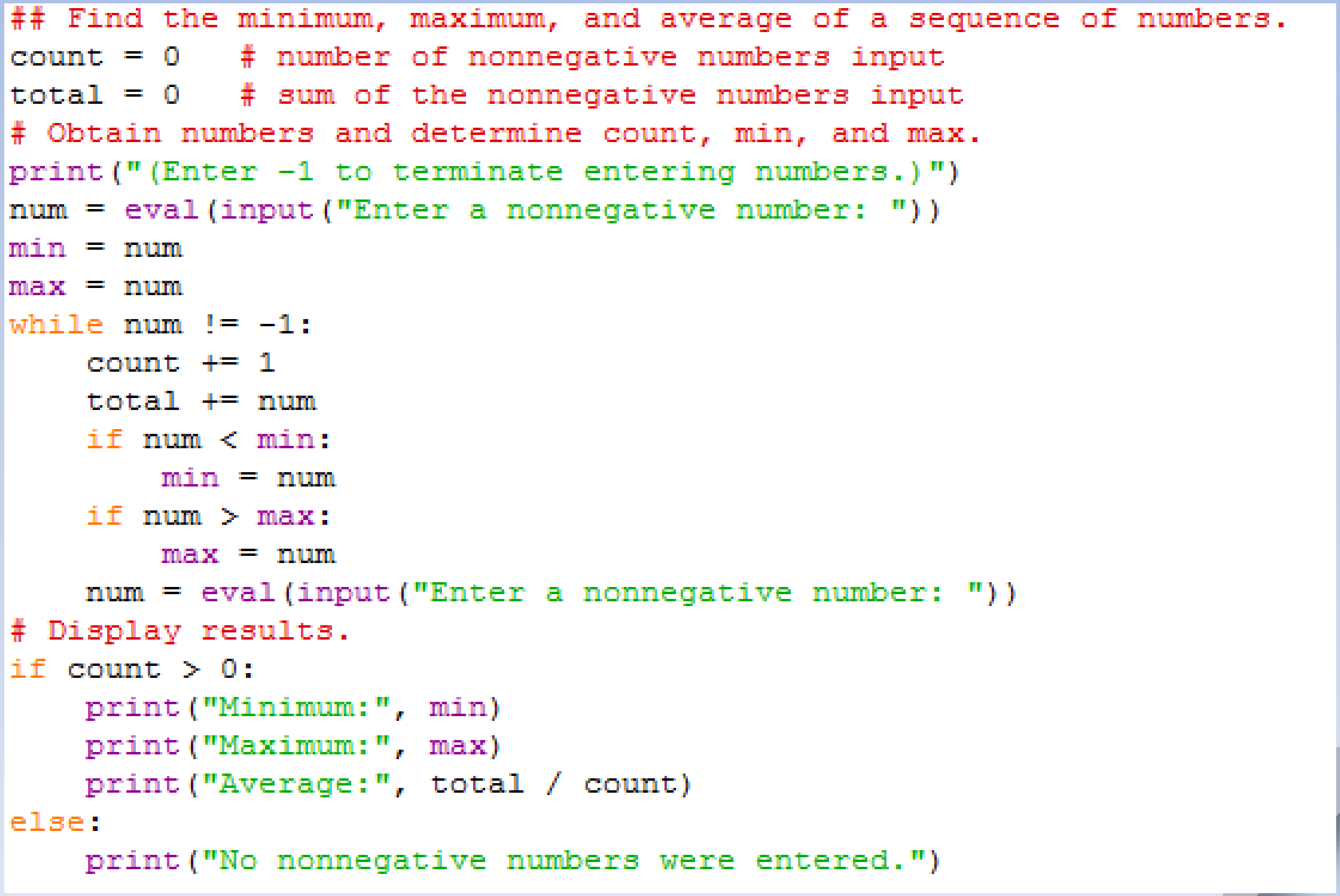
```
## Display facts about the United States.
print("Enter a number from the menu to obtain a fact")
print("about the United States or to exit the program.\n")
print("1. Capital")
print("2. National Bird")
print("3. National Flower")
print("4. Quit\n")
while True:
    num = int(input("Make a selection from the menu: "))
    if num == 1:
        print("Washington, DC is the capital of the United States.")
    elif num == 2:
        print("The American Bald Eagle is the national bird.")
    elif num == 3:
        print("The Rose is the national flower.")
    elif num == 4:
        break
```

The *while* Loop

- Example: Find min, max, average from a series of numbers entered by the user until a value of -1 is input
- Program accepts numbers continuously until a negative number is entered

The *while* Loop

- Example: Find min, max, average from a series of numbers entered by the user until a value of -1 is input



```python
## Find the minimum, maximum, and average of a sequence of numbers.
count = 0   # number of nonnegative numbers input
total = 0   # sum of the nonnegative numbers input
# Obtain numbers and determine count, min, and max.
print("(Enter -1 to terminate entering numbers.)")
num = eval(input("Enter a nonnegative number: "))
min = num
max = num
while num != -1:
    count += 1
    total += num
    if num < min:
        min = num
    if num > max:
        max = num
    num = eval(input("Enter a nonnegative number: "))
# Display results.
if count > 0:
    print("Minimum:", min)
    print("Maximum:", max)
    print("Average:", total / count)
else:
    print("No nonnegative numbers were entered.")
```

How To Seek Input in Loops

Previous example shows how to:

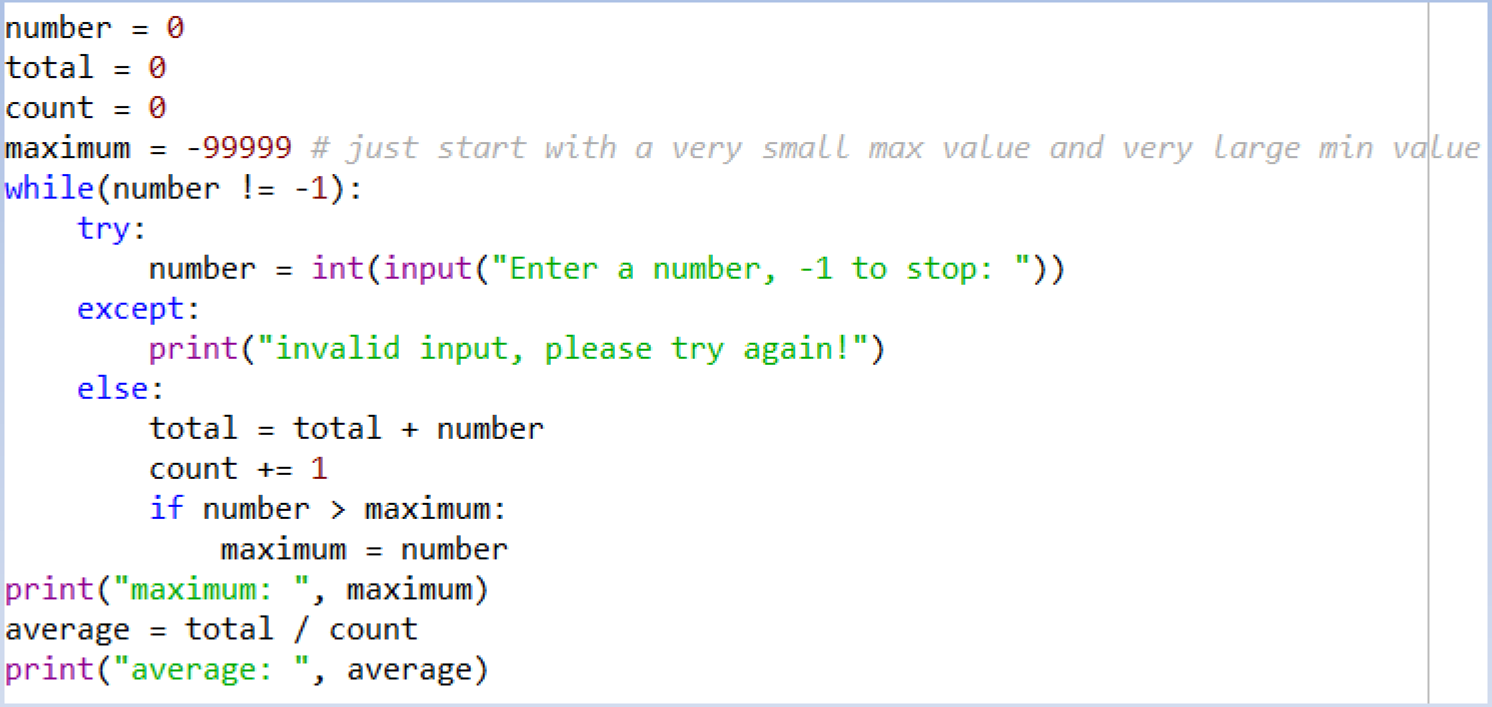
1. Prime the pump (Scan 1st set of inputs)
2. Start loop, checking for End Of Input
3. Perform operations on the input
4. Re-scan (Seek more input)
5. End the loop

The *while* Loop

- You can also use the input as validation to control repetition OR use a ‘sentinel controlled loop’
- How can you extend it to accept only numeric values?

The *while* loop

One way to use an exception handler to catch invalid input and keep accepting input from user until -1 is entered

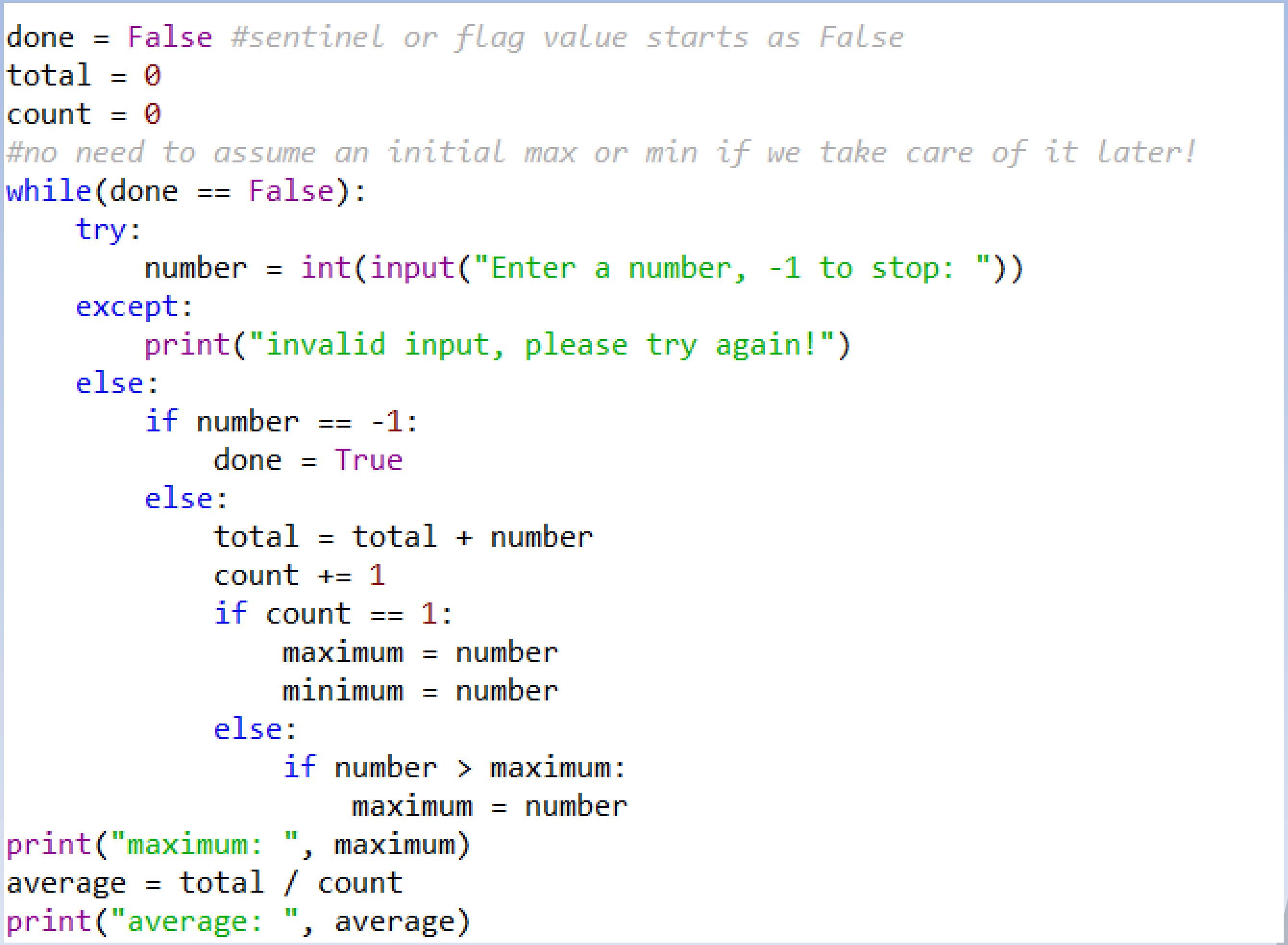


```
number = 0
total = 0
count = 0
maximum = -99999 # just start with a very small max value and very large min value
while(number != -1):
    try:
        number = int(input("Enter a number, -1 to stop: "))
    except:
        print("invalid input, please try again!")
    else:
        total = total + number
        count += 1
        if number > maximum:
            maximum = number
print("maximum: ", maximum)
average = total / count
print("average: ", average)
```

Seek Input in Loops using sentinel

1. Use a “flag” or “sentinel” or “loop indicator”
2. Loop begin/continue (set it up for “true condition” for the flag for the first time)
3. Seek input within the loop
   - If input is done, change flag value
   - Else perform operations on the input
4. Go to Step 2

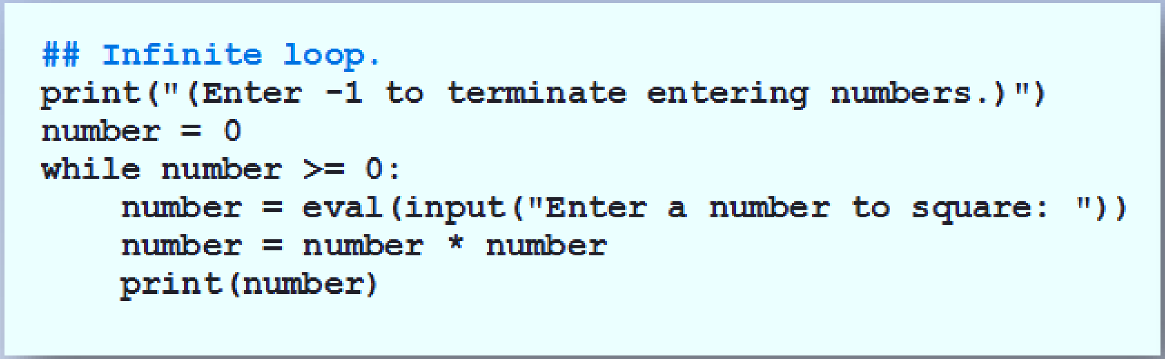
The *while* loop using a sentinel with exception handler



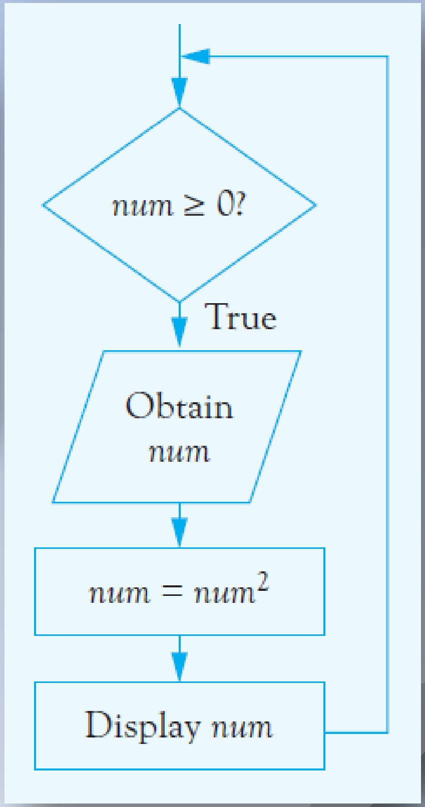
```python
done = False #sentinel or flag value starts as False
total = 0
count = 0
#no need to assume an initial max or min if we take care of it later!
while(done == False):
    try:
        number = int(input("Enter a number, -1 to stop: "))
    except:
        print("invalid input, please try again!")
    else:
        if number == -1:
            done = True
        else:
            total = total + number
            count += 1
            if count == 1:
                maximum = number
                minimum = number
            else:
                if number > maximum:
                    maximum = number
print("maximum: ", maximum)
average = total / count
print("average: ", average)
```

Infinite Loops

- Example: Condition ***`number >= 0`*** always true.



```
## Infinite loop.
print("(Enter -1 to terminate entering numbers.)")
number = 0
while number >= 0:
    number = eval(input("Enter a number to square: "))
    number = number * number
    print(number)
```


$num \geq 0$?
True
Obtain num
$num = num^2$
Display num