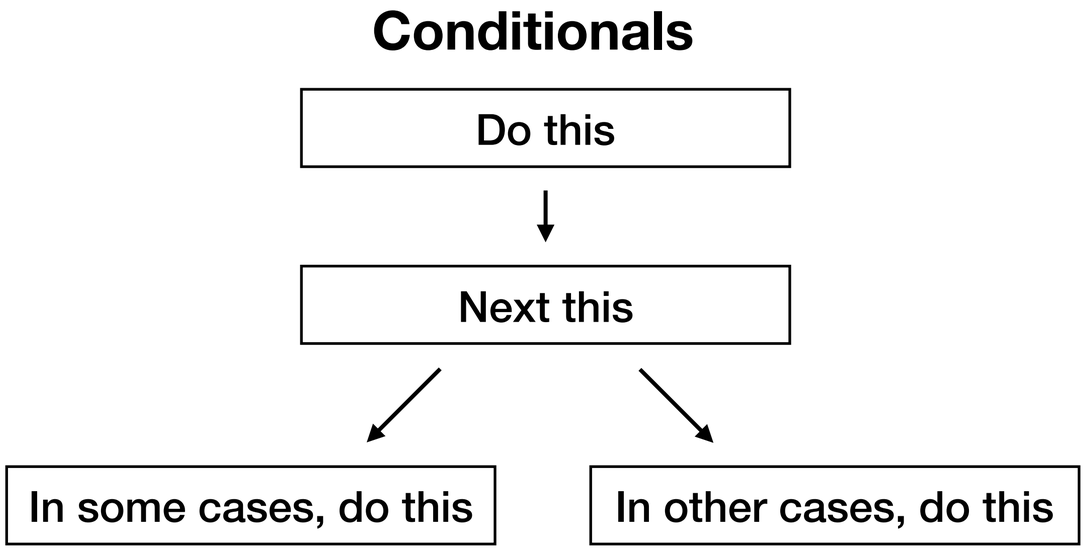
Control Flow II: Loops

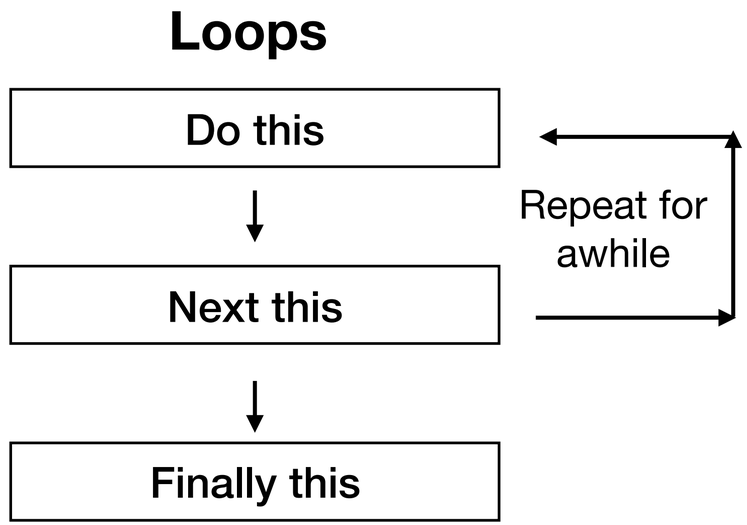
PNI Summer Internship 2020

Mai Nguyen

Control flow

- Control flow allows us to control which code is run (conditionals) and how many times (loops)



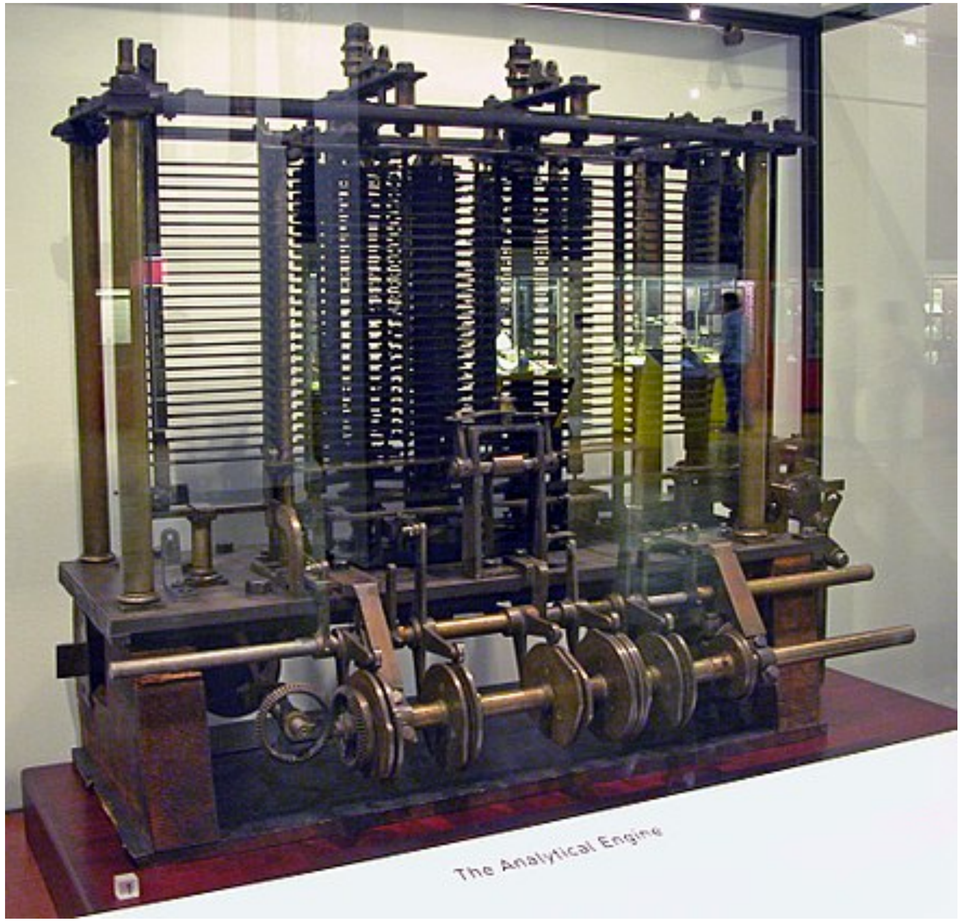


if/else, switch

for loops, while loops

Ada Lovelace and the first software loop

Ada Lovelace, 1815-1852



Model of Charles Babbage’s Analytical Engine

For loops

- Allows for repeating a chunk of code for a specific number of times
- Print the numbers 1-5

Print each number individually

```
disp(1)
disp(2)
disp(3)
disp(4)
disp(5)
```

```
     1

     2

     3
    ....
```

Use for loop

```
for i = 1:5
    disp(i)
end
```

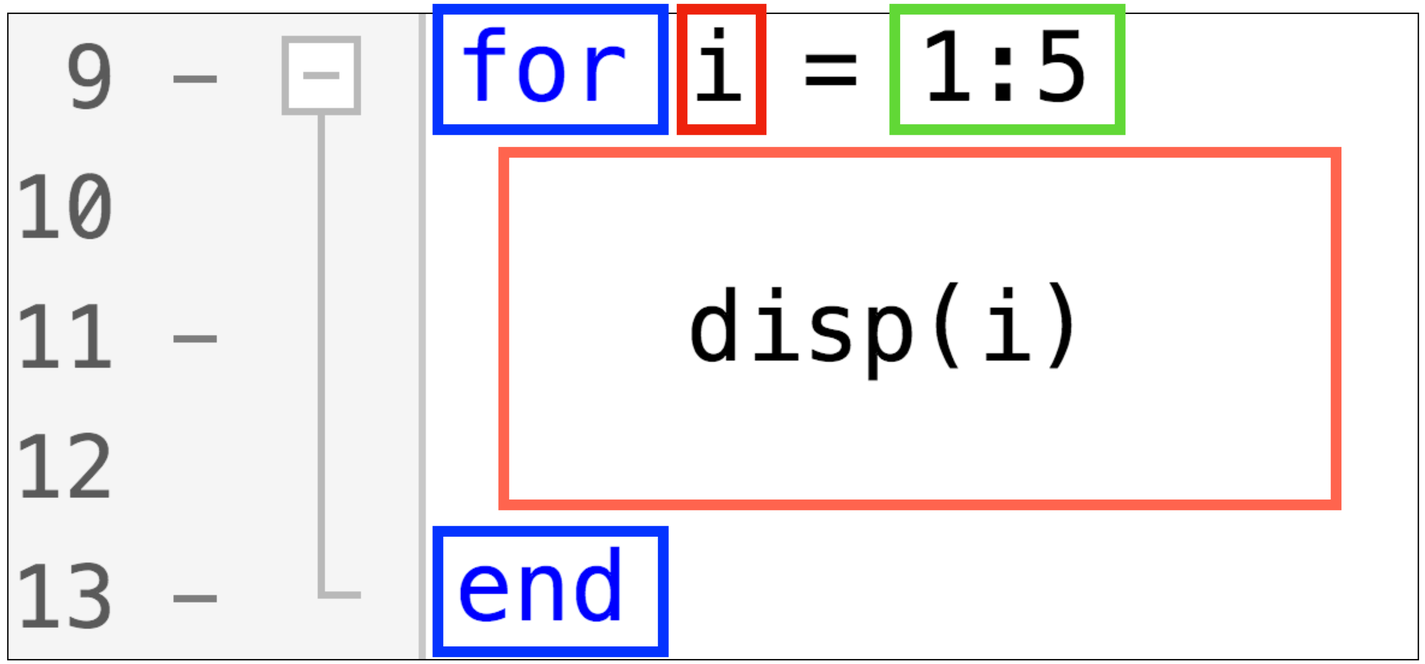
```
     1

     2

     3
    ....
```

Now print numbers 1-1000

For loops: syntax



- Start with keyword **for**
- **Loop index**, in this case i but can be anything
- **Vector counter**, on each iteration of the loop, the loop index is assigned the next value in the vector
- Close with keyword **end**
- **Loop body** is executed each iteration

Python aside:

MATLAB

```
for i = 1:5

    disp(i)

end
```

Python

```
for i in range(5):
    print(i+1)
```

For loops: loop index and vectors

Count 1-5

```
>> % an example
for i = 1:5
    fprintf([num2str(i) ' '])
end



1 2 3 4 5
```

Count vector of random nums

```
>> % an example
for n = [1 3 4 99 0]
    fprintf([num2str(n) ' '])
end



1 3 4 99 0
```

Count through vec variable

```
>> % an example
vec = [-1 1 3 5];
for val = vec
    fprintf([num2str(val) ' '])
end



-1 1 3 5
```

Exercises 1

- **Ex 1.1:** Write a for loop that counts backwards from 10 to 1.
- **Ex 1.2:** Write a for loop that squares each values in a 1x5 vector of random numbers and displays it
- **Ex 1.3:** On your own: Write a for loop takes the square root (use sqrt() function) for the even numbers from 1 to 10 and displays it

For loops: assign values to an array

- Often times use loops to fill values in an array

```
>>
for i = 1:10

    my_vec(i) = i * 2;

end
```

For loops: assign values to an array

- Better practice: pre-define an vector of nans or zeros and fill

```
my_vec = NaN(1,10);
for i = 1:length(my_vec)

    my_vec(i) = i*2;

end
```

For loops: assign values to an array

- Fill matrix rows

```
>> my_matrix = NaN(5,10);

for i = 1:size(my_matrix,1)
    my_matrix(i,:) = i;

end
```

Python aside: assign values to an array

- Typically you don’t do this kind of “pre-allocate and fill” arrays or numpy arrays in Python

- Instead, use `append( )` to add values to end of an array

Exercise 2

- **Exercise 2.1**
  - Write a for loop that increments `i` from 1 to 10. On each iteration, get a random number from 1 to `i` and store to a vector.
  - Modify the for loop to pre-allocate memory in a vector. Display the final vector
- **Exercise 2.2 (on your own)**
  - Write a loop that increments `n` from 7 to 2. On each iteration, calculate n^n and store to a pre-allocated vector.
  - Display the final vector.

For loops: nested loops

week3_lec5.m | scratch.m

```
1  for i = 1:5
2      % print i
3      fprintf([num2str(i) '\n'])
4
5      for j = 1:3
6          % print i,j indented
7          fprintf(['\t' num2str(i) ',' num2str(j) '\n'])
8      end
9
10 end
```

- Outer loop
- Inner loop

For loops: nested loops

week3_lec5.m | scratch.m

```
1  for i = 1:5
2      % print i
3      fprintf([num2str(i) '\n'])
4
5      for j = 1:3
6          % print i,j indented
7          fprintf(['\t' num2str(i) ',' num2str(j) '\n'])
8      end
9
10 end
```

COMMAND WINDOW

For loops: nested loops

week3_lec5.m | scratch.m

```
1  for i = 1:5
2      % print i
3      fprintf([num2str(i) '\n'])
4
5      for j = 1:3
6          % print i,j indented
7          fprintf(['\t' num2str(i) ',' num2str(j) '\n'])
8      end
9
10 end
```

COMMAND WINDOW

```
>> scratch
1
        1,1
        1,2
        1,3
```

On first iteration of outer loop, i = 1

Then iterates through the inner loop, j = 1, j = 2, j =3

For loops: nested loops

week3_lec5.m | scratch.m

```
1  for i = 1:5
2      % print i
3      fprintf([num2str(i) '\n'])
4
5      for j = 1:3
6          % print i,j indented
7          fprintf(['\t' num2str(i) ',' num2str(j) '\n'])
8      end
9
10 end
```

COMMAND WINDOW

```
>> scratch
1
        1,1
        1,2
        1,3
2
        2,1
        2,2
        2,3
```

On second iteration of outer loop, i = 2

Then iterates through the inner loop, j = 1, j = 2, j =3

Exercises 3

- **Ex 3.1**: Create a 3x5 matrix of NaNs. Using nested for loops, fill each element of the matrix with the product of the row and column number.

- **Ex 3.2**: [on your own] Create a 3x4 matrix of 0s. Using nested for loops, fill each element of the matrix with the square of the sum of the row and column $(row+col)^2$

While loops

- **For loops** iterate for a specific number of times specified in the loop counter

- **While loops** continue until a condition is met

While loops

Increment counter until reach 5

```
>> counter = 0;

while counter < 5

    disp(counter)
    counter = counter + 1;

end
     0

     1

     2

     3

     4
```

Calculate 5! (5x4x3x2x1)

```
>> n = 5;
factorial = 1;

while n > 1
    factorial = factorial * n;
    n = n - 1;
end
disp(factorial);
   120
```

While loops: syntax

```
3     counter = 0;
4
5     while counter < 5
6
7         disp(counter)
8         counter = counter + 1;
9
10    end
```

- Start with keyword **while**
- Some **condition** that evaluates to true or false; continue repeating until this is false
- Close with keyword **end**
- **Loop body**, repeat this every iteration
- **Modify value** of loop condition
- **Initialize value** of loop condition

While loops: syntax

MATLAB

```
counter = 0;

while counter < 5

    disp(counter)
    counter = counter + 1;

end
```

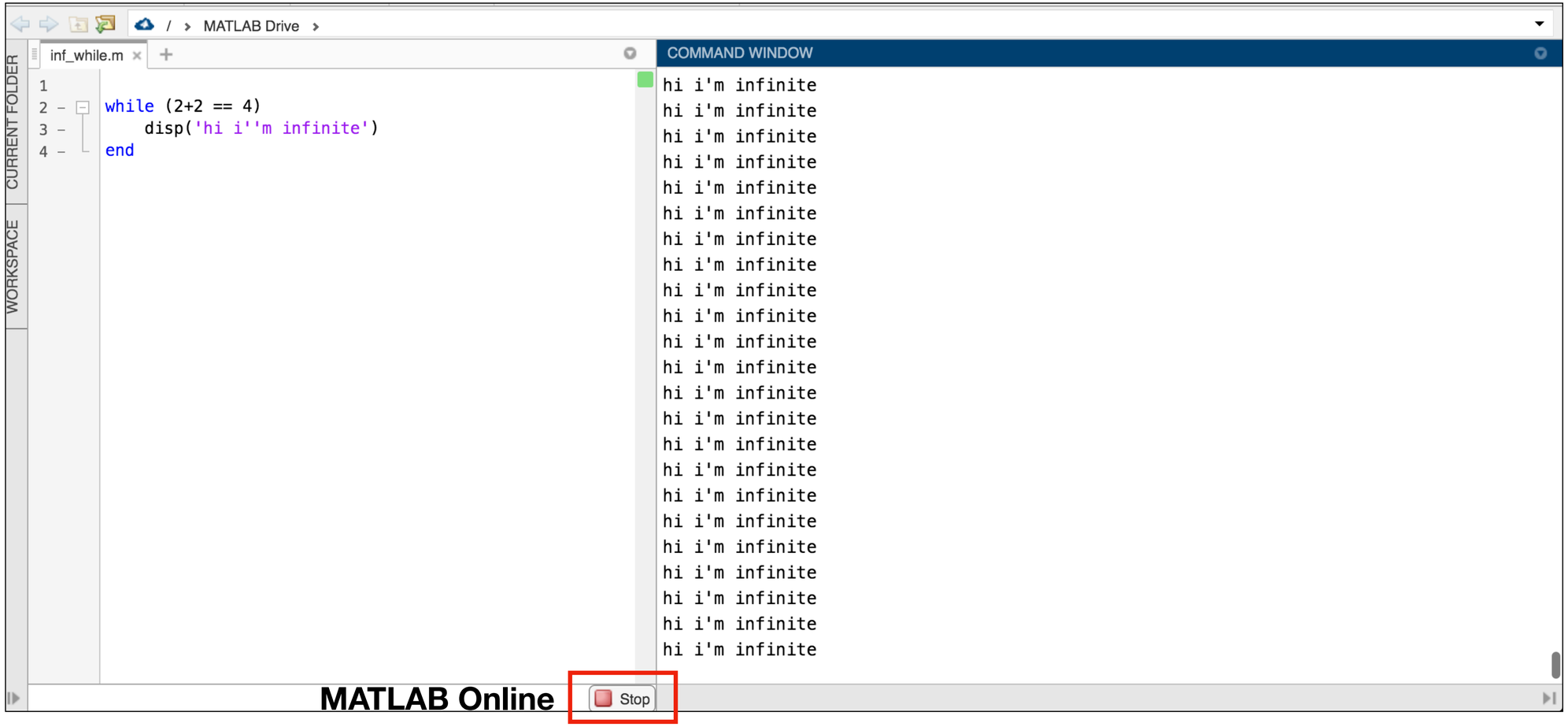
Python

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

Exercises 4

- **Ex 4.0**: Use a while loop to display the integers up to 10
- **Ex 4.1:** Use a while loop to count up the number of consecutive integers starting from 1 that you need to sum together to get at least 100.
- **Ex 4.2:** [on your own] Use a while loop to find the next prime number greater than a 204. Use the isprime() built-in function.

The infinite loop



MATLAB Online

MATLAB desktop: ctrl + c

Breaking out of a loop: break

```
 6 -     count = 1;
 7 -     while count < 500
 8
 9 -         if fraction > 100
10 -             break;
11 -         end
12
13 -     end
14 -     disp(count)
```

- **break** allows you to exit the loop early

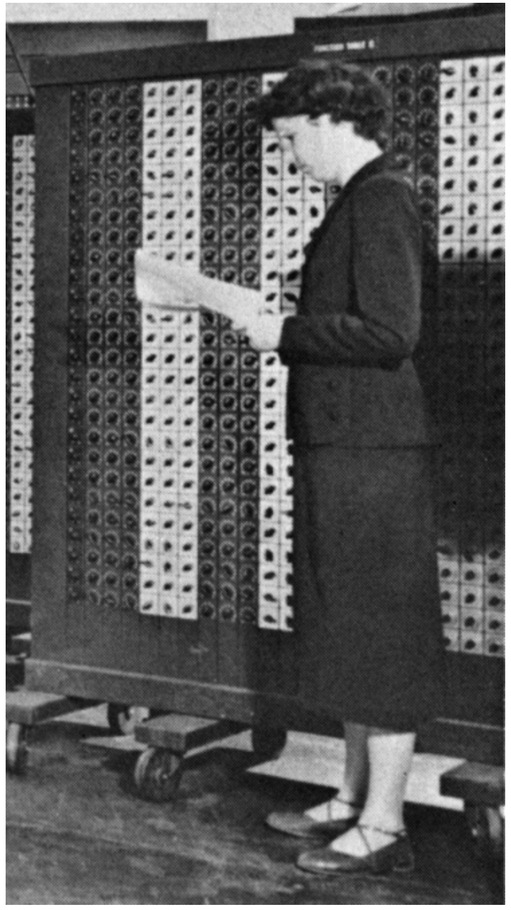
Breaking out of a loop: break

inf_while.m | input_check.m

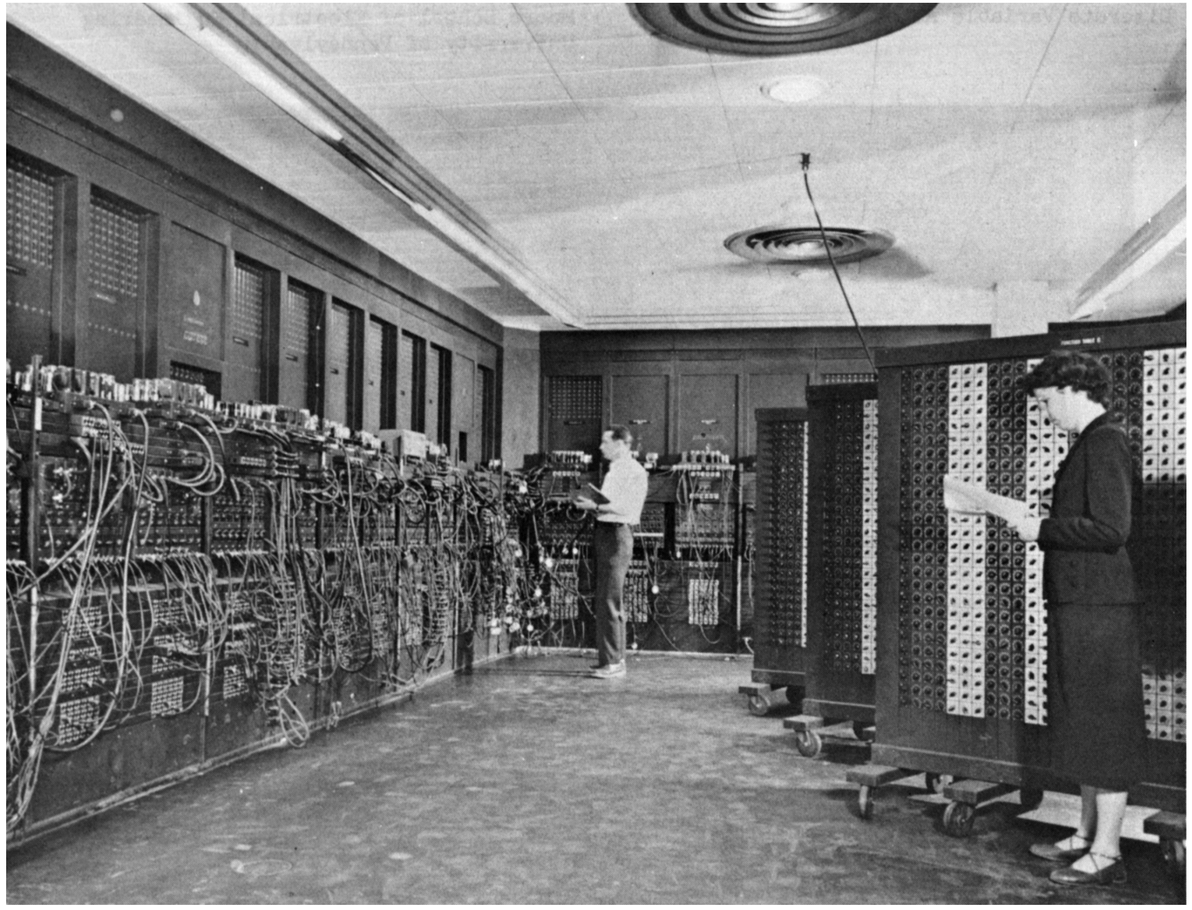
```
1  while true
2      val = input('Enter a number between 1-100: ');
3
4      if (val >= 1) && (val <= 100)
5          disp(' ')
6          disp('That''s a valid number!')
7          break;
8      end
9
10     disp(' ')
11     disp('Invalid. Try again.')
12
13 end
14
15
```

COMMAND WINDOW

Betty Holberton and the first breakpoint



Betty Holberton, 1917-2001



ENIAC ca. 1947-1955

Skip an iteration: continue

inf_while.m | input_check.m | loop_continue.m

```
1
2 -   vec = [1 2 3 NaN 4 5 NaN];
3
4 -   for i = vec
5
6 -       if isnan(i)
7 -           continue;
8 -       end
9
10 -      disp(i)
11
12 -  end
```

- **continue** skips the rest of the loop and goes back to the top

Skip an iteration: continue

inf_while.m | input_check.m | loop_continue.m

```
1
2 -   vec = [1 2 3 NaN 4 5 NaN];
3
4 -   for i = vec
5
6 -       if isnan(i)
7 -           continue;
8 -       end
9
10 -      disp(i)
11
12 -  end
```

COMMAND WINDOW

```
     1

     2

     3

     4

     5

>>
```

For loops versus while loops

Use for loops

- Already know number of iterations
- Iterating through specific values

While loops

- Don’t know number of iterations i.e. searching for something

Review

- For loop: `for i = vector`
- While loop: `while cond_is_true`
- `break`
- `continue`