Implementing the Caesar Cipher

Counting Loops

Duke
UNIVERSITY

Concept Review: Loops, Indexes

- You've used different loops in solving problems:
  - Finding Codons or Tags in a String
    - `while(true){… break …}`
  - Reading lines from a FileResource
    - `for(String s : fr.lines()) {..}`
- You've used indexes to access strings
  - `"cgatga".indexOf("atg");`
  - `"cgatga".substring(1,4);`

Accessing Characters in a String

- The reverse of "CGATTA" is "ATTAGC"
  - Useful in genomics to process strand this way
- Palindromes can be source of fun
  - Нажал кабан на баклажан
  - Alli trota la tortilla
  - Eh, ça va, la vache?
  - Draw, O Caesar, erase a coward.
- How do we create the reverse of a string?

Indexing a String

- Must understand for loop with three parts
  - Separated by semi-colons
  - Initialization (happens once, before guard)

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Indexing a String

- Must understand for loop with three parts
  - Separated by semi-colons
  - Initialization (happens once, before guard)
  - Guard, evaluated before loop body

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Indexing a String

- Must understand for loop with three parts
  - Separated by semi-colons
  - Initialization (happens once, before guard)
  - Guard, evaluated before loop body
  - Increment, executed after loop body

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

For and While Compared

- For loop: "syntactic sugar" for while loop

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

```
public String reverse(String s){
    String ret = "";
    int k=0;
    while (k < s.length()){
      ret = s.charAt(k) + ret;
      k += 1;
    }
    return ret;
  }
```

For and While Compared

- For loop: "syntactic sugar" for while loop

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

- initialize

```
public String reverse(String s){
    String ret = "";
    int k=0;
    while (k < s.length()){
      ret = s.charAt(k) + ret;
      k += 1;
    }
    return ret;
  }
```

For and While Compared

- For loop: "syntactic sugar" for while loop

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

- initialize
- loop guard

```
public String reverse(String s){
    String ret = "";
    int k=0;
    while (k < s.length()){
      ret = s.charAt(k) + ret;
      k += 1;
    }
    return ret;
  }
```

For and While Compared

- For loop: "syntactic sugar" for while loop

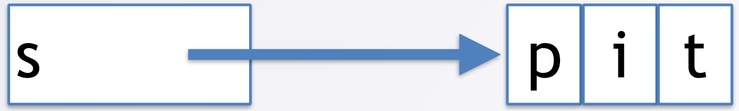
```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

```
public String reverse(String s){
    String ret = "";
    int k=0;
    while (k < s.length()){
      ret = s.charAt(k) + ret;
      k += 1;
    }
    return ret;
  }
```

- initialize
- loop guard
- increment

Anatomy of a For Loop

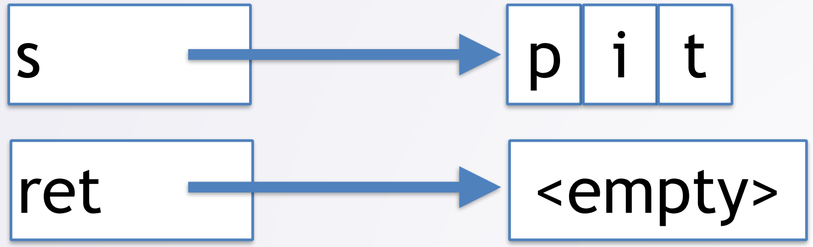
- Consider the call reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Anatomy of a For Loop

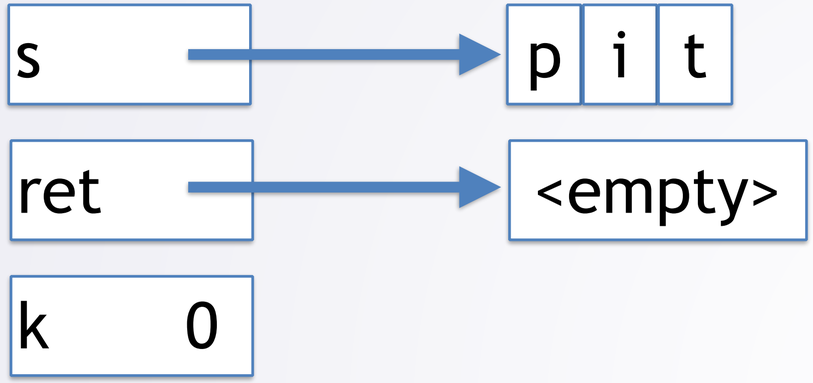
- Consider the call reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Anatomy of a For Loop

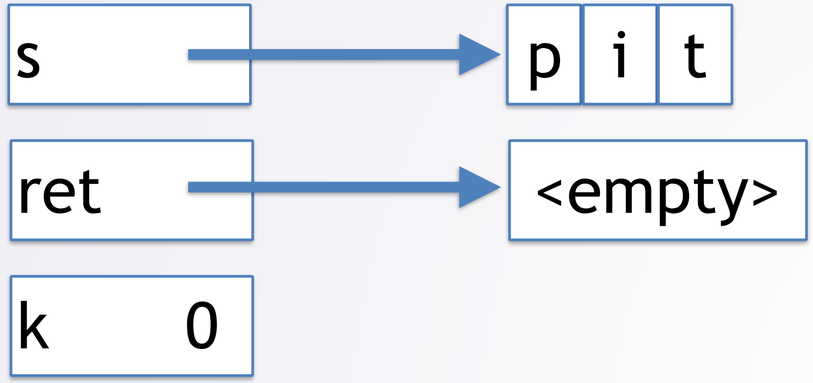
- Consider the call reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Anatomy of a For Loop

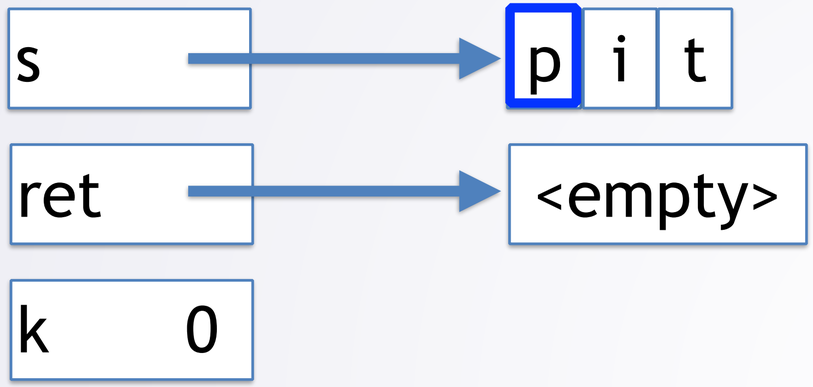
- Consider the call reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Anatomy of a For Loop

- Consider the call reverse("pit")

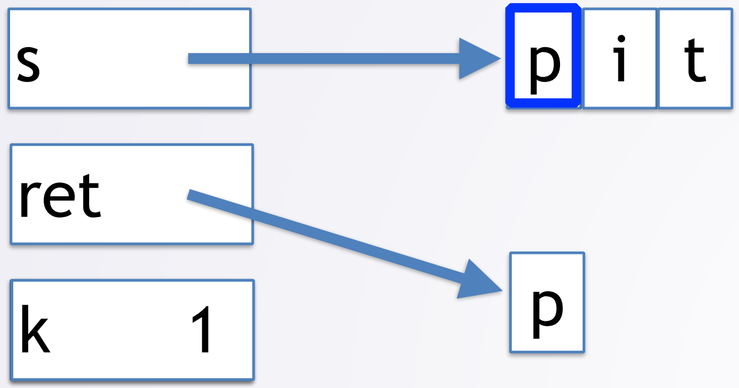


‘p’ + “” = “p”

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Anatomy of a For Loop

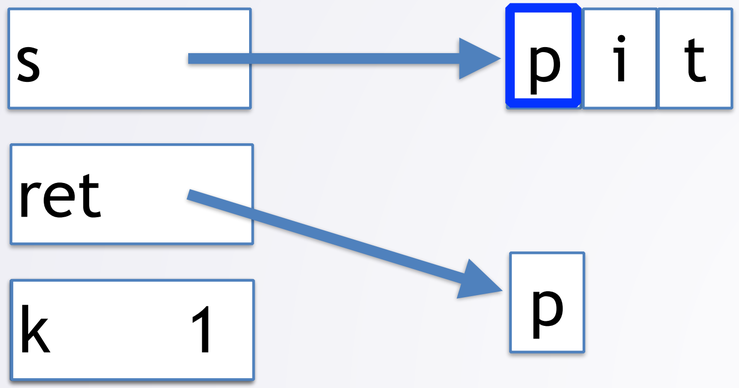
- Consider the call reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Anatomy of a For Loop

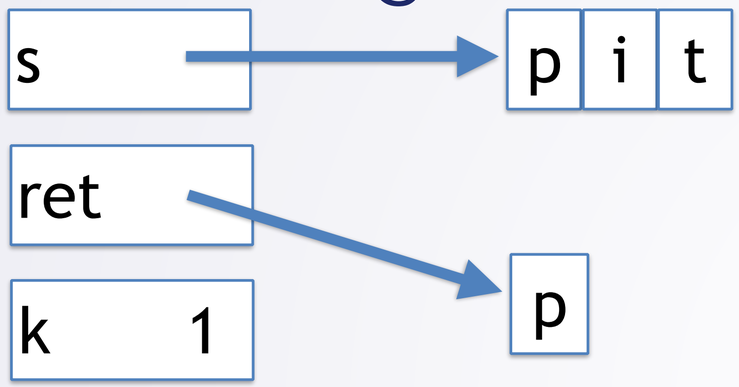
- Consider the call reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

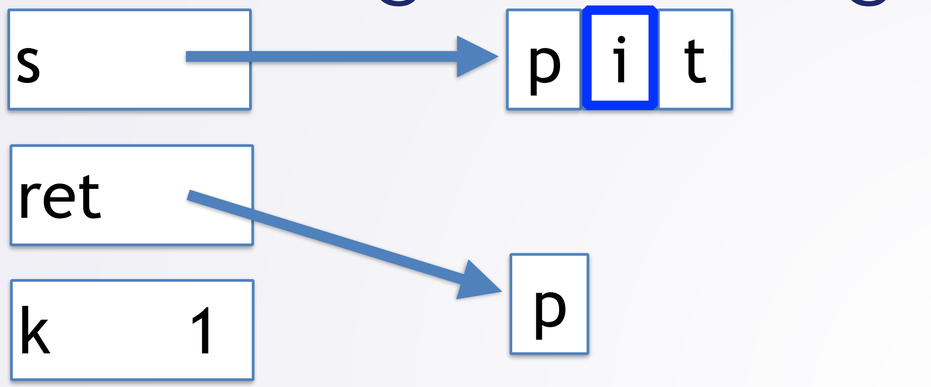
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

- Continuing with tracing reverse("pit")

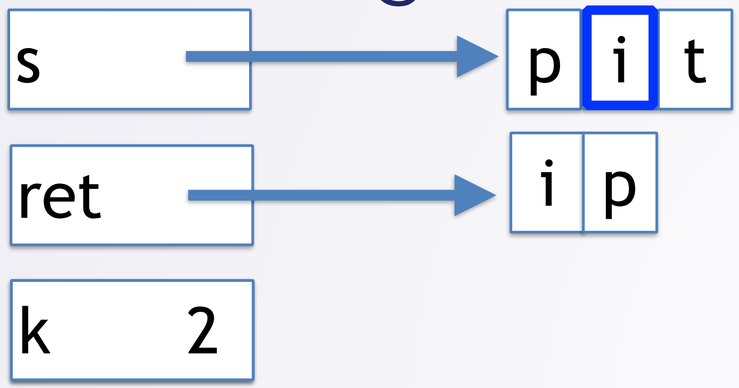


"i" + "p" = "ip"

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
      ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

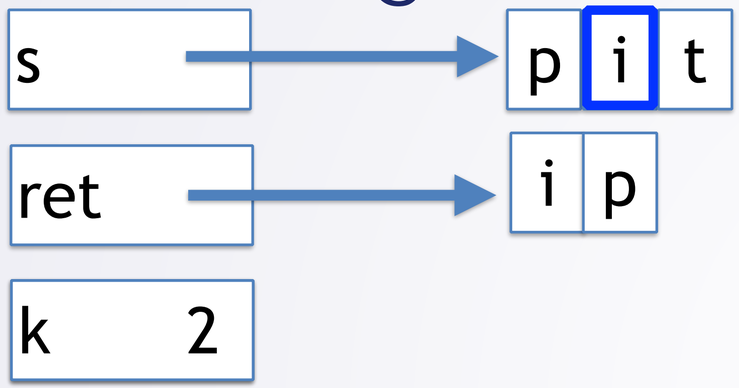
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

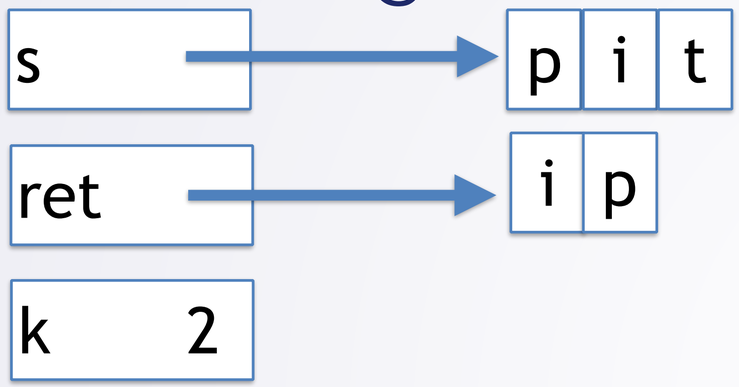
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

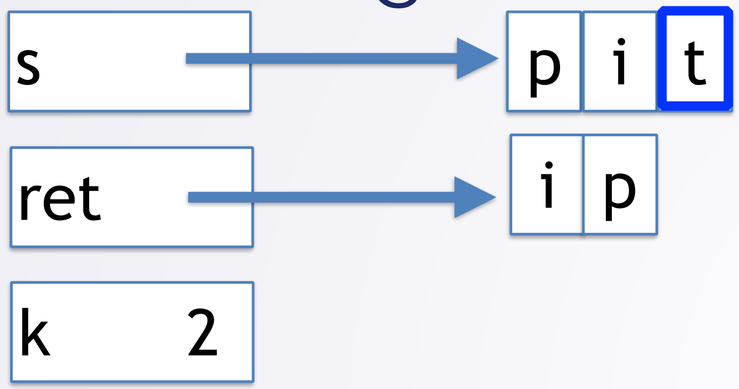
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

- Continuing with tracing reverse("pit")

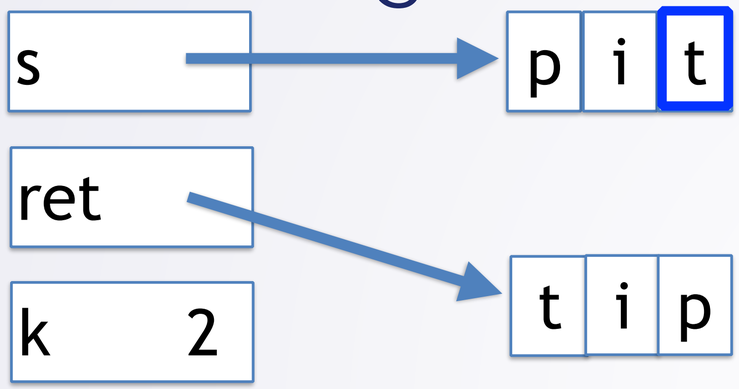


't' + "ip" ="tip"

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

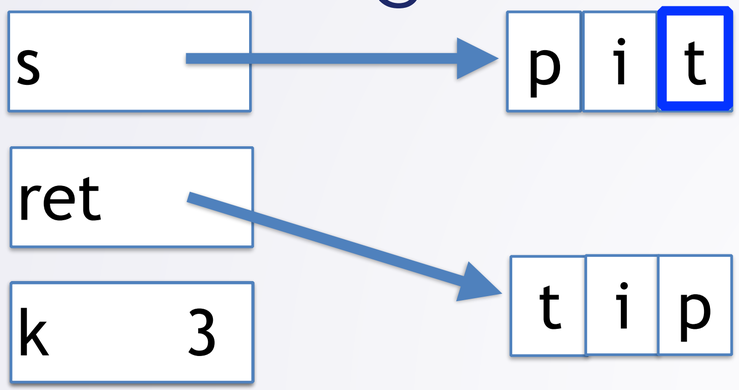
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

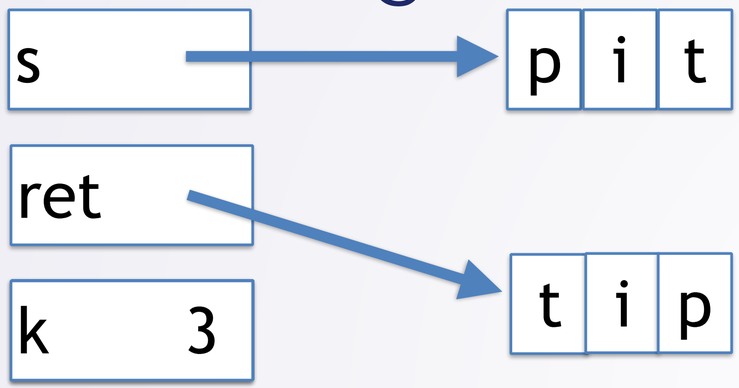
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

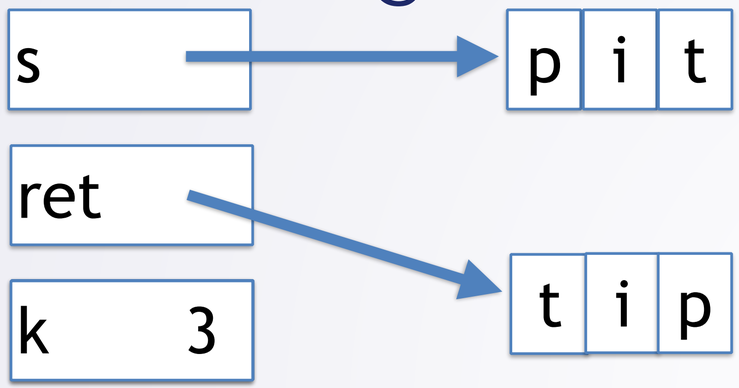
- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

- Continuing with tracing reverse("pit")

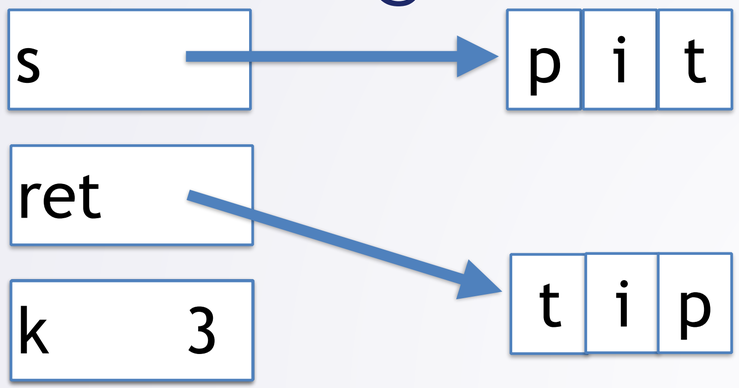


3 < 3 is false

```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
        ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Continuing with Loop Tracing

- Continuing with tracing reverse("pit")




```
public String reverse(String s){
    String ret = "";
    for(int k=0; k < s.length(); k += 1){
       ret = s.charAt(k) + ret;
    }
    return ret;
}
```

Idiomatic For Loop

- Many programmers use `i` as index variable

```
for(int i=0; i < s.length(); i += 1)
```

- Many programmers use **`i++`**, not **`i += 1`**

```
for(int i=0; i < s.length(); i++)
```

- Sometimes declare variable before loop, so scope extends to after loop

```
int i;
for(i=0; i < s.length(); i += 1){
}
// can reference value of i here
```