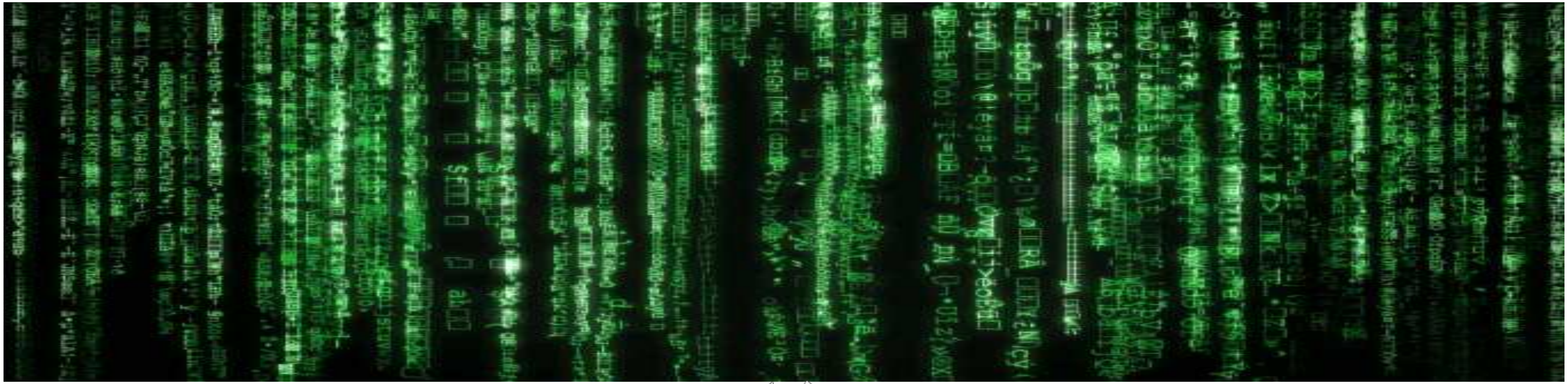


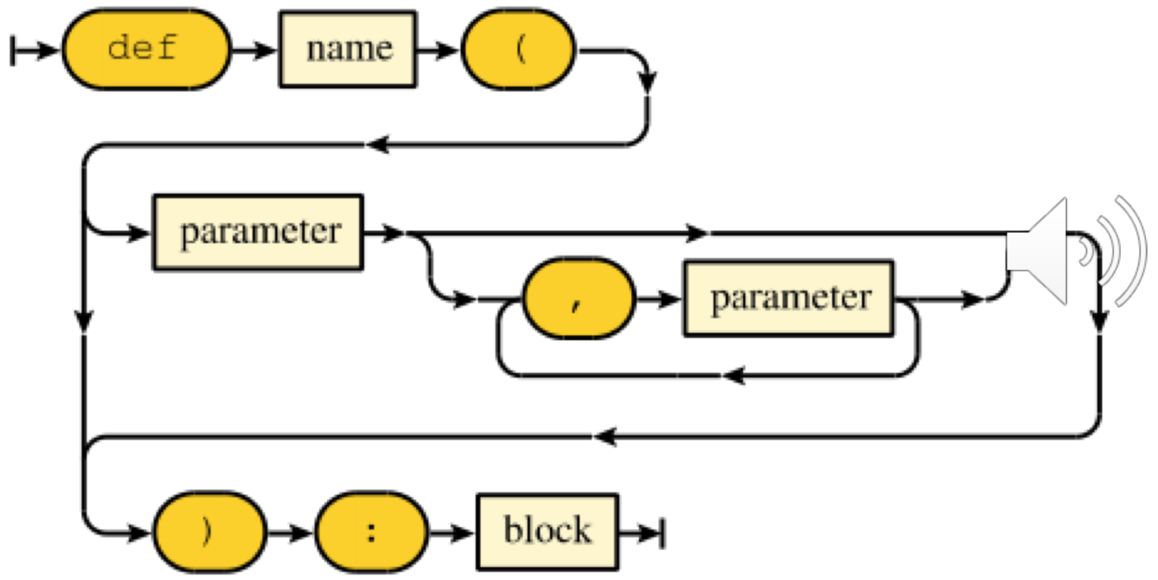
INF 110 Discovering Informatics

Functions

NAU
NORTHERN
ARIZONA
UNIVERSITY

Function Declaration

Syntax:



Example 1:

```
def is_chocolate(flavor):
    return flavor == "chocolate"
```

Live Code is_chocolate?

Tasks: Write the is_chocolate() function on the previous slide. Then apply it to different values and variables.

Learning Outcomes

- Writing simple functions
- Creating and using function parameters

Example 2: Spread

```
def spread(values):
    return max(values) - min(values)
```

Example 3: Pig Latin

```
def pig_latin(word):
    return word[1:] + word[0] + "ay"
```

Note: This is a good start but the pig latin rules are slightly more complicated. What words does this not work for?

Example 4: Quadratic Polynomial

```
def quadratic(x, a, b, c):
    return a*x**2 + b*x + c
```

Positional Arguments

Means you call the arguments in the order listed in the parameter list:

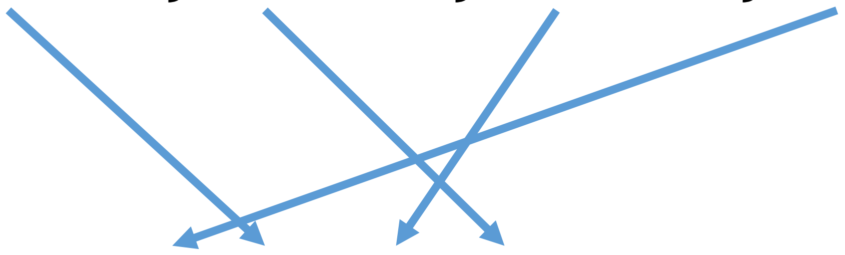
```
quadratic(1.0, 2.0, 3.0, [illegible].0)
```

```
def quadratic(x, a, b, c):
    return a*x**2 + b*x + c
```

Keyword Arguments

Means you call the arguments by specifying the parameter name:

```
quadratic(a=1.0, c=2.0, b=3.0, x=4.0)
```



```
def quadratic(x, a, b, c):
    return a*x**2 + b*x + c
```

Default Arguments

Means the function supplies one or more default values that can be absent from the call:

```
quadratic(c=2.0)
```

```
def quadratic(x=1.0, a=2.0, b=3.0, c=4.0):
    return a*x**2 + b*x + c
```

Apply

The **apply** method creates an array by calling a function on every element in one or more input columns

- First argument: Function to apply
- Other arguments: The input column(s)

```
table_name.apply(my_function, 'column_label')
```

end