Programming With Google
Who here is a developer?
Coverage

Some key fundamentals

Write a script to hack an app
Who Am I?

Robin Wood

https://digi.ninja

@digininja
From the UK
Qualifications for this talk

• Started work as desktop app developer in 1996
• Moved to web app in 2003
• Now a tester but still do bits on the site
• Published over 50 security tools
The Theory

The potentially patronising* bit

*patronizing
Variables
Main Types

Strings – “hello”, ‘world’, “1234”, “this is it”
Integers – 42, -1, 9223372036854775807*
Floats – 3.1415, 42.42, -57.4, 10.0
Objects – car.no_of_wheels, fruit.colour**

* Technically a Long but you get the idea ** color
Assigning

$number = 42
my_string = “hello”

var aFloat = 3.14

int i
i = 27
Conditionals
Equality

== usually means “is this equal to?”

if my_number == 27
    print “x equals 27”

if message == “hello”
    print “He said hello”
Equality Gotcha

Remember from variables...

= usually means “becomes equal to”

if x = 42

print “x has now been assigned the value 42”
Inequality

Is not equal to can sometimes be:

!=  Or  <>

if x != 10
    print "It isn’t 10"
Otherwise

if x = 10
    print “x is 10”
else
    print “x is not 10”
if x == 10
    print "it is 10"
else
    if x != 20
        print "it isn’t 20"
    else
        print "it must be 20"

Not the best way to do this!
Maths*

Standard maths things

if x < 10    if x is less than 10
if x >= 10   if x is 10 or greater
Boolean Logic

if a==“hello” and b==“robin”

if username==“root” or uid==0

if count == 10 && time < 40 || hits > 7
Brackets

To force order, use brackets.

These two statements mean different things:

if (x==10 and y==3) or (z==7)

if (x==10) and (y==3 or z==7)
### Proof

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

### if \((x==10 \text{ and } y==3) \text{ or } (z==7)\)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

### if \((x==10) \text{ and } (y==3 \text{ or } z==7)\)
Loops
For Loops

Do this so many times

```plaintext
for i=1; i < 10; i = i + 1
  print i
```

Result:

1
2
3
4
5
6
7
8
9

Shortcut

```
i = i + 1
```
can often be written
```
i++ or i += 1
```
While Loops

Do this until criteria met

key = “n”
while key != “y”
    print “not yet”
    key = get_keypress()
print “done”
Blocks
Keyword/Symbol Delimited

if result == true then
    print "it worked"
x = 32
    <more stuff here>
end
print "outside the block"

if result == true {
    print "it worked"
x = 32
    <more stuff here>
}
print "outside the block"
if result == true:
    print "it worked"
if x == 32:
    print "and the answer was 32"
print "outside the block"
Step Back – Delimited

if result == true then
    print “it worked”
    x = 32
    <more stuff here>
end
print “outside the block”
Functions
Group Code Together

- Like a block but can be reused
- Avoids repetition
- Saves effort
- Saves mistakes
Parameters In

```javascript
function saySomething (message, num) {
    print “the message is: ” + message
    print “double the number is: ”
    print num * 2
}
saySomething ("hello", 7)
```
Parameters Out

```javascript
function invert_evens (number) {
    if number % 2 == 0
        return (number * -1)
    return number
}

neg = invert_evens (1471)
```
Practice Time
Disclaimer

• This is not for production code
• Stay on the first page of Google
• If you don’t understand it at a common sense level, don’t use it
Let’s Do It!
Disclaimer

• This is not for production code
• Stay on the first page of Google
• If you don’t understand it at a common sense level, don’t use it
What Language?

• It doesn’t matter
• One you can get support on
• Ruby/Python probably have most online support
Practice
Practice
Practice
Keep all your work
Any Questions?

Robin Wood

https://digi.ninja

@digininja