

Special Topics in Computer Science- CSC 4992

Making Choices with **if** Statements

Adding Intelligence

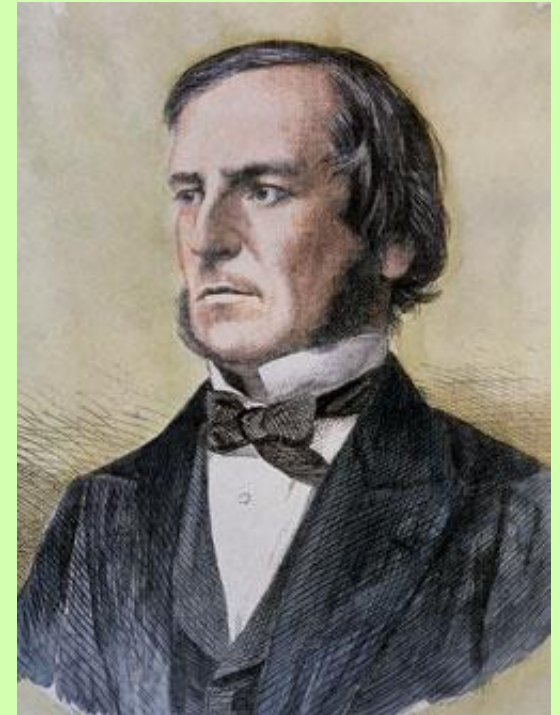
- Check for the presence or absence of a condition in the environment
- Take the appropriate actions
- Involves making a choice (to do or not to do, that is the question!)

Asking Questions

- Is the number greater than 0?
- Is the password correct?
- Are we at the end of the list?
- Does the file exist?
- Is the number evenly divisible by 2?

Answers: Boolean Expressions

- Literals: **True**, **False**
- Variables
- Results of comparisons
- Results of logical operations



George Boole
(1815-1865)

Comparisons

```
>>> x = 2
>>> x == 2
True
>>> x < 3
True
>>> x > 4
False
```

Operators are ==, !=, <, >, <=, >=

== means equals, = means assignment

!= means not equal to

Example: Absolute Value

- $|x| = -x$ if $x < 0$, or x otherwise
- Write code that converts a number x to its absolute value

Use an **if** Statement

```
If x is less than 0  
    Set x to -x
```

Use an **if** Statement

```
if x < 0:  
    x = -x
```

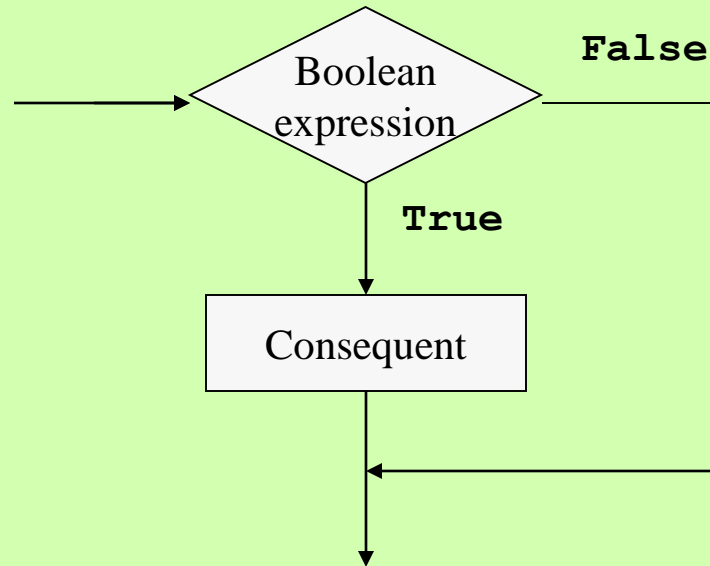
Also called a *one-way if* statement

If the comparison returns **True**, run the nested statement

Otherwise, do nothing

```
if <Boolean expression>:                # The condition  
    <sequence of statements>            # The consequent
```


Behavior of One-Way **if** Statement



| | |
|---------------------------------|------------------|
| if <Boolean expression>: | # The condition |
| <sequence of statements> | # The consequent |

Example: Checking User Inputs

- A program will work only for inputs > 0
- All other numbers should be rejected with an error message
- Only the positive inputs can be processed

The Area of a Circle

```
import math

radius = float(input('Enter the radius: '))
area = math.pi * radius ** 2
print(area)
```

This version allows negative inputs to be used - very bad!

Use an **if-else** Statement

```
import math

radius = float(input('Enter the radius: '))
if radius <= 0:
    print('ERROR: Input number must be positive! ')
else:
    area = math.pi * radius ** 2
    print(area)
```

This version checks the input and traps errors before they can cause damage

The program responds gracefully with an error message

Syntax of **if-else** Statement

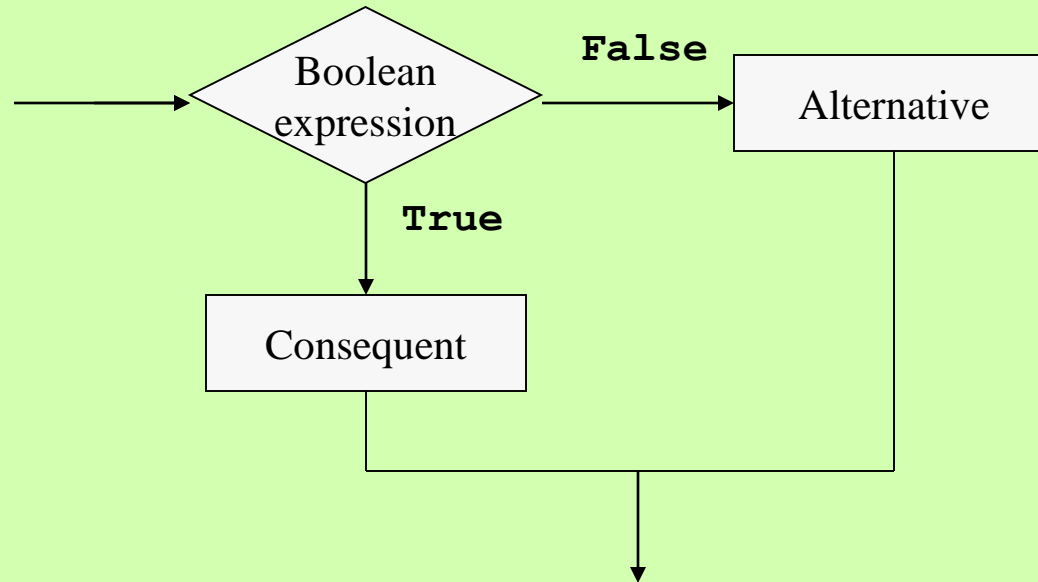
```
import math

radius = float(input('Enter the radius: '))
if radius <= 0:
    print('ERROR: Input number must be positive! ')
else:
    area = math.pi * radius ** 2
    print(area)
```

| | |
|---------------------------------|-------------------|
| if <Boolean expression>: | # The condition |
| <sequence of statements> | # The consequent |
| else : | |
| <sequence of statements> | # The alternative |

Also called a *two-way if* statement

Behavior of Two-Way **if** Statement



| | |
|---------------------------------|-------------------|
| if <Boolean expression>: | # The condition |
| <sequence of statements> | # The consequent |
| else : | |
| <sequence of statements> | # The alternative |

More Input Checking

```
rate = int(input('Enter the interest rate[0-100]: '))  
interest = principal * rate / 100  
print('Your interest is', interest)
```

This version allows rates < 0 or rates > 100

Very bad!

More Input Checking

```
rate = int(input('Enter the interest rate[0-100]: '))
if rate < 0 or rate > 100:
    print('ERROR: Rate must be between 0 and 100!')
else:
    interest = principal * rate / 100
    print('Your interest is', interest)
```

Use comparisons and the *logical operator* **or** to restrict the rate to the legitimate values

The Logical Operators

```
<Boolean expression> or <Boolean Expression>
```

```
<Boolean expression> and <Boolean expression>
```

```
not <Boolean expression>
```

The Boolean expressions can be

- literals (**True** or **False**)

- variables

- comparisons

- Boolean function calls

- other expressions connected by logical operators

Truth Table for **or**

```
rate = -1  
  
print(rate < 0 or rate > 100)
```

| A | B | A or B |
|-------|-------|--------|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

Python stops evaluating operands when enough info is available to determine the result (short-circuit evaluation)

Truth Table for **or**

```
rate = 160  
  
print(rate < 0 or rate > 100)
```

| A | B | A or B |
|-------|-------|--------|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

In this case, both operands must be evaluated

Truth Table for **or**

```
rate = 50
```

```
print(rate < 0 or rate > 100)
```

| A | B | A or B |
|-------|-------|--------|
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

In this case, likewise

Truth Table for **and**

```
rate = -1  
  
print(rate >= 0 and rate <= 100)
```

| A | B | A and B |
|-------|-------|---------|
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

The **and** operator uses short-circuit evaluation, too

Truth Table for **not**

```
import os.path

filename = input('Enter the file name: ')
if not os.path.isfile(filename):
    print('ERROR: File does not exist!')
else:
    # Process the file
```

| A | not A |
|-------|-------|
| True | False |
| False | True |

Precedence of Logical Operators

```
print(False or True and not True)  
  
# Same as  
  
print(False or (True and (not True)))
```

Ranking:

| | |
|------------|--------------------|
| not | (logical negation) |
| and | (logical product) |
| or | (logical sum) |

Multiple Conditions

```
rate = int(input('Enter the interest rate[0-100]: '))
if rate < 0 or rate > 100:
    print('ERROR: Rate must be between 0 and 100!')
else:
    interest = principal * rate / 100
    print('Your interest is', interest)
```

An alternative way to write this is to use a series of **if** statements with different conditions

Multiway **if** Statement

```
rate = int(input('Enter the interest rate[0-100]: '))
if rate < 0:
    print('ERROR: Rate must be greater than 0!')
elif rate > 100:
    print('ERROR: Rate must be less than 101!')
else:
    interest = principal * rate / 100
    print('Your interest is', interest)
```

| | |
|-----------------------------------|---------------------------|
| if <Boolean expression>: | # The first condition |
| <sequence of statements> | # The first consequent |
| elif <Boolean expression>: | # The second condition |
| <sequence of statements> | # The second consequent |
| ... | |
| else : | |
| <sequence of statements> | # The default alternative |

Multiway **if** Statement

```
grade = int(input('Enter the numeric grade: '))

if grade >= 90:
    print('A')
elif grade >= 80:
    print('B')
elif grade >= 70:
    print('C')
else:
    print('F')
```