3. Conditional Execution

Topics:
- Boolean values
- Relational operators
- if statements
- The Boolean type
Motivation

Problem:

Assign positive float values to variables \(a\) and \(b\) and print the values \(a^{b}\) and \(b^{a}\).

Solution:

\[
a = \text{input}(\text{`Enter a pos float: `})
b = \text{input}(\text{`Enter a pos float: `})
\text{print } a^{b}, b^{a}
\]
Motivation

Problem:

Assign float values to variables $a$ and $b$ and print just the larger of $a^b$ and $b^a$

Solution:

$7^2 < 2^7$  
$2^3 < 3^2$
Solution Using If-Else

```python
a = input('Enter a pos float: ')
b = input('Enter a pos float: ')
aTob = a**b
bToa = b**a
if aTob > bToa:
    print(aTob)
else:
    print(bToa)
```

This is what is called “conditional execution.”
If-Else: How Does it Work?

aTob = a**b
bToa = b**a
if aTob > bToa:
    print aTob
else:
    print bToa

Let's suppose the value of a is 2 and the value of b is 7.
Solution Using If-Else

\[
a_{Tob} = a^{**}b \\
b_{Toa} = b^{**}a
\]

if \(a_{Tob} > b_{Toa}\):
    print \(a_{Tob}\)
else:
    print \(b_{Toa}\)

The comparison \(a_{Tob} > b_{Toa}\) is called a boolean expression. It is either True or False.

Is the value of \(a_{Tob}\) larger than the value of \(b_{Toa}\)?
Solution Using If-Else

\[
\begin{align*}
\text{aTob} &= a**b \\
\text{bToa} &= b**a \\
\text{if aTob} &> \text{bToa:} \\
&\quad \text{print aTob} \\
\text{else:} \\
&\quad \text{print bToa}
\end{align*}
\]

The boolean expression \( \text{aTob} > \text{bToa} \) is True so execute print \( \text{aTob} \)

Is the value of \( \text{aTob} \) larger than the value of \( \text{bToa} \)? Yes!
If-Else: How Does it Work?

\[
\begin{align*}
a_{Tob} &= a^{**} b \\
b_{Toa} &= b^{**} a \\
\text{if } a_{Tob} > b_{Toa}: \\
&\quad \text{print } a_{Tob} \\
\text{else:} \\
&\quad \text{print } b_{Toa}
\end{align*}
\]

Now let's suppose the value of \( a \) is 7 and the value of \( b \) is 2.
If-Else: How Does it Work?

```python
aTob = a**b
bToa = b**a
if aTob > bToa:
    print aTob
else:
    print bToa
```

Is the value of aTob larger than the value of bToa?
If-Else: How Does it Work?

\[
a_{Tob} = a^{**}b \\
b_{Toa} = b^{**}a
\]

if \(a_{Tob} > b_{Toa}\):
    print \(a_{Tob}\)
else:
    print \(b_{Toa}\)

The boolean expression \(a_{Tob} > b_{Toa}\) is False so execute
print \(b_{Toa}\)

Is the value of \(a_{Tob}\) larger than the value of \(b_{Toa}\)? No!
If-Else: How Does it Work?

```python
aTob = a**b
bToa = b**a
if aTob > bToa:
    print aTob
else:
    print bToa
```

Note the punctuation and the indentation.

This is essential syntax.

Forgetting the colons is a major boo boo!
"Synonym"

\[ a_{Tob} = a^{**}b \]
\[ b_{Toa} = b^{**}a \]

if \( a_{Tob} > b_{Toa} \):
    print \( a_{Tob} \)
else:
    print \( b_{Toa} \)

if \( a^{**}b > b^{**}a \):
    print \( a^{**}b \)
else:
    print \( b^{**}a \)

In a comparison, we can have general expressions on either side of the "<".
The if-else Construction

```
if  Boolean expression  :
    Statements to execute if the expression is True
else:
    Statements to execute if the expression is False
```

This is an example of conditional execution. The if-else construction is sometimes called “alternative execution”
The **if-else** Construction

```python
if a**b > b**a:
    z = b**a
else:
    z = a**b
print 'The smaller value is: ', z
```

The blue box decides whether the green box or the pink box is executed.

After that choice is processed, this print is carried out.
Reminder that Indentation Is Important

if x%2==0:
    y = x/2
    print y
else:
    y = (x+1)/2
    print y

If x is even, then the code on the left will print x/2 while the code on the right will print x/2 twice (on separate lines).
Another Example

Problem:

The last character in a string 5-character string is 'y'.
Change the 'y' to 'i' and add 'es'

Solution:

\[ s = s[0:4] + 'ies' \]
A Modified Problem

If the last character in a 5-character string \( s \) is 'y', then
1. change the 'y' to 'i'
2. add 'es'
3. assign the result to a variable \( \text{plural} \).

Otherwise, just add 's' and assign the result to a variable \( \text{plural} \).

This will require the if-else construction.
if s[4]=='y':
    plural = s[0:4] + 'ies'
else:
    plural = s + 's'
print s, plural

Remember: s[0:4] names the substring comprised of the first 4 characters.
Discussion of Solution

```python
if s[4] == 'y':
    plural = s[0:4] + 'ies'
else:
    plural = s + 's'
print s, plural
```

A new comparison is being used.

If you want to check to see if two expressions have the same value, use `==`.

Discussion of Solution

```python
if s[4] == 'y':
    plural = s[0:4] + 'ies'
else:
    plural = s + 's'
print s, plural
```

The print statement is executed after the if-else is processed. E.g.

```
carry  carries
```
Relational Operators

< Less than
> Greater than
<= Less than or equal to
>= Greater than or equal to
== Equal to
!= Not equal to
Relational Operators in Action

x --> 3  y --> 6

x < y      True
2*x > y    False
x <= y     True
x >= y     False
x == y/2   True
x != y/2   False

If the expression on the left is a different numerical type then the expression on the right, everything is converted to float.
Boolean Operations with Strings

Are two strings equal?

```python
>>> s = 'abc'
>>> s == 'abc'
True
>>> s == 'abc '
False
```

Two strings are equal if they have the same length and agree in each position.
Boolean Operations with Strings

>>> s = 'Dog'
>>> s > 'Horse'
False
>>> s < 'Horse'
True
>>> s < 'dog'
True

Alphabetical order? Alphabetical order. If s < t is true then s comes before t in the “extended dictionary” based on this ordering of characters: '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
Relational Operators in Action

\[ x \text{ ---\textgreater} \ 'key' \quad y \text{ ---\textgreater} \ 'hockey' \]

\[
\begin{align*}
    x & < y \quad \text{False} \\
    x & > y \quad \text{True} \\
    'hoc' + x & \leq y \quad \text{True} \\
    x & \geq y \quad \text{True} \\
    x & == y[3:] \quad \text{True} \\
    x & != x+' \quad \text{True}
\end{align*}
\]

Comparisons based on alphabetical order. \( x < y \) is false because 'key' does not come before 'hockey' in the dictionary.
Another Problem

Assume that \( s_1 \) and \( s_2 \) are initialized strings.

Write code that prints them in alphabetical order on separate lines.
Solution

```python
if s1 < s2:
    print s1
    print s2
else:
    print s2
    print s1
```

```
s1 ---+-> 'cat'
s2 ---+-> 'dog'
```

Is this True or False?
Solution

```python
if s1 < s2:
    print s1
    print s2
else:
    print s2
    print s1
```

Output:

s1 ---> 'cat'
s2 ---> 'dog'

It's true!
**Solution**

```python
if s1 < s2:
    print s1
    print s2
else:
    print s2
    print s1
```

Is this True or False?

- `s1` ---> 'dog'
- `s2` ---> 'cat'
- `s1 < s2`
```python
if s1<s2:
    print s1
    print s2
else:
    print s2
    print s1
```

It's false!

Output:

```
cat
dog
```
Indentation Is Important

if s1 < s2:
    print s1
    print s2
else:
    print s2
    print s1

s1  --->  'cat'
s2  --->  'dog'

Output:

    cat
dog
cat
What if You Have More than Two Alternatives?

For example, given a numerical test score between 0 and 100, print out the letter grade equivalent according to these rules:

A  90-100
B  80-89
C  70-79
U  <70
The If-Elif-Else Construction

```python
x = input('Score: ')  
if x>=90:  
    grade = 'A'
elif x>=80:  
    grade = 'B'
elif x>=70:  
    grade = 'C'
else:  
    grade = 'U'
print grade
```

Read “elif” as “else if”
The If-Elif-Else Construction

```python
x = input('Score: ')  
if x>=90:  
    grade = 'A'  
elif x>=80:  
    grade = 'B'  
elif x>=70:  
    grade = 'C'  
else:  
    grade = 'U'  
print grade
```

Note the punctuation and the indentation.

Read “elif” as “else if”
If-Elif-Else: How it Works

```python
x = input('Score: ')  
if x>=90:
    grade = 'A'
elif x>=80:
    grade = 'B'
elif x>=70:
    grade = 'C'
else:
    grade = 'U'
print grade
```

1. Is this true?
2. No.
3. Proceed to the next comparison.
x = input('Score: ')  
if x>=90:  
    grade = 'A'  
elif x>=80:  
    grade = 'B'  
elif x>=70:  
    grade = 'C'  
else:  
    grade = 'U'  
print grade
If-Elif-Else: How it Works

```python
x = input('Score: ')  
if x>=90:
    grade = 'A'
elif x>=80:
    grade = 'B'
elif x>=70:
    grade = 'C'
else:
    grade = 'U'
print grade
```

1. Is this true?
2. Yes.
3. Execute the statement(s) it guards and proceed to whatever follows the if-elif-else

The indentation scheme “tells” Python what comes after the if-elif-else
If-Elif-Else: How it Works

```python
x = input('Score: ')  
if x>=90:              
    grade = 'A'        
elif x>=80:            
    grade = 'B'        
elif x>=70:            
    grade = 'C'        
else:                  
    grade = 'U'        
print grade
```

x ---> 95

1. Is this true?
2. Yes.
3. Execute the statement(s) it guards and proceed to whatever follows the If-elif-else
x = input('Score: ')  
if x>=90:  
    grade = 'A'  
elif x>=80:  
    grade = 'B'  
elif x>=70:  
    grade = 'C'  
else:  
    grade = 'U'  
print grade  

1. Is this true?  
2. No.  
3. Proceed to the next comparison.
If-Elif-Else: How it Works

```python
x = input('Score: ')  
if x>=90: 
    grade = 'A'  
elif x>=80: 
    grade = 'B'  
elif x>=70: 
    grade = 'C'  
else:  
    grade = 'U'  
print grade
```

- x ---+> 65

1. Is this true?
2. No.
3. Proceed to the next comparison.
If-Elif-Else: How it Works

```python
x = input('Score: ')  
if x>=90:  
    grade = 'A'  
elif x>=80:  
    grade = 'B'  
elif x>=70:  
    grade = 'C'  
else:  
    grade = 'U'  
print grade
```

1. Is this true?
2. No.
3. Execute “the else”
4. Proceed to what follows the if-elif-else.

x --->
65
I prefer the one on the left. The letter grade is an essential feature of the computation and having a variable that houses it reminds me of that fact,
grade = 'B'
nApples = input('How many apples sent to Prof:')
if nApples<10:
    grade = grade + '-'
print grade

Let's review all the "if" variations...
**Standard if-else**

```
if A boolean expression:

else:

Code that is executed after the whole “if” is processed.
```

Exactly one of the green boxes is executed.
**if-elif**

```python
if A boolean expression:

elif Another boolean expression:
```

If both boolean expressions are false, no green box is executed. Otherwise, the “first” green box that is “guarded” by a true boolean expression is executed.
Multiple if-elif With Else

if : 
  : 
elif : 
  : 
elif : 
  : 
else:
  :

The first green box guarded by a true boolean expression is executed. If they are all false, then the else's green box is executed.
Note that if all the boolean expressions are False, then no green code is executed. Otherwise the first green box guarded by a true boolean expression is executed.
More Complicated Boolean Expressions

\[ x \rightarrow 3 \quad y \rightarrow 6 \quad z \rightarrow 9 \]

\[
(x < y) \ \text{and} \ (x < z) \quad \text{True}
\]

\[
(x > y) \ \text{and} \ (x < z) \quad \text{False}
\]

\[
(x < y) \ \text{and} \ (x > z) \quad \text{False}
\]

\[
(x > y) \ \text{and} \ (x > z) \quad \text{False}
\]

This showcases the and operator.
The and Operator

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>and</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td></td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Here and are Boolean-valued expressions
More Complicated Boolean Expressions

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>((x &lt; y) \text{ or } (x &lt; z))</td>
<td>True</td>
</tr>
<tr>
<td>((x &gt; y) \text{ or } (x &lt; z))</td>
<td>True</td>
</tr>
<tr>
<td>((x &lt; y) \text{ or } (x &gt; z))</td>
<td>True</td>
</tr>
<tr>
<td>((x &gt; y) \text{ or } (x &gt; z))</td>
<td>False</td>
</tr>
</tbody>
</table>

This showcases the or operator.
Example

Fact: A length-4 string is a palindrome if
The first and last characters are the same and
The middle two characters are the same

```python
s = input('s: ') 

if (s[0] == s[3]) and (s[1] == s[2]):
    print 'palindrome'
else:
    print 'not a palindrome'
```
Example

Fact: $x$ is inside the interval $[L,R]$ if it is no smaller than $L$ and no bigger than $R$.

```python
x = input('x: ')
L = input('L: ')
R = input('R: ')

if (L<=x) and (x<=R):
    print 'Inside'
else:
    print 'Outside'
```
Equivalent Solution

```python
x = input('x: ')
L = input('L: ')
R = input('R: ')

if (L<=x) and (x<=R):
    print 'Inside'
else:
    print 'Outside'
```
# The or Operator

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>or</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td></td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Here **green** and **yellow** are boolean-valued expressions.
Example

Fact: A length-4 string is a partial palindrome if the first and last characters are the same or if the middle two characters are the same.

```python
s = input('s: ')  
if (s[0]==s[3]) or (s[1]==s[2]):  
    print 'partial palindrome'  
else:  
    print 'not a partial palindrome'
```
Example

Fact: $x$ is inside the interval $[L,R]$ if it is no smaller than $L$ and no bigger than $R$.

```python
x = input('x: ')  
L = input('L: ')  
R = input('R: ')  

if (x<L) or (R<x):
    print 'Outside'
else:
    print 'Inside'
```
Example

Fact: $x$ is inside the interval $[L,R]$ if it is no smaller than $L$ and no bigger than $R$.

If $(x<L)$ or $(R<x)$:
   print ‘Outside’
Else:
   print ‘Inside’

If $(L<=x)$ and $(x<=R)$:
   print ‘Inside’
Else:
   print ‘Outside’

Often you can arrange a conditional execution in several ways.
More Complicated Boolean Expressions

\[ \text{not } (x < y) \quad \text{False} \]
\[ \text{not } (x > y) \quad \text{True} \]

This showcases the not operator.
The not Operator

<table>
<thead>
<tr>
<th></th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

Here is a boolean-valued expression
A Summarizing Example

Input a string. If it has even length, then hyphenate in the middle:

baseball          base-ball

If it has odd length, then hyphenate around the middle character:

frisbee          fri-s-bee
The `len` Function

If ever you need to compute the length of a string then use the built-in function `len`.

\[
\begin{align*}
\text{s} & = \text{‘abcdef’} \\
\text{n} & = \text{len(s)} \\
\text{m} & = \text{n}/2 \\
\text{First} & = \text{s}[:\text{m}] \\
\text{Second} & = \text{s}[:\text{m}] \\
\end{align*}
\]

\[
\begin{align*}
\text{x} & \longrightarrow \text{‘abcdef’} \\
\text{n} & \longrightarrow 6 \\
\text{m} & \longrightarrow 3 \\
\text{First} & \longrightarrow \text{‘abc’} \\
\text{Second} & \longrightarrow \text{‘def’}
\end{align*}
\]
The `len` Function

If ever you need to compute the length of a string then use the built-in function `len`.

```python
s = 'abcdefg'
n = len(s)
m = n/2
First = s[:m]
Second = s[m:]
```

- `x` ----> `‘abcdefg’`
- `n` ----> 7
- `m` ----> 3
- `First` ----> `‘abc’`
- `Second` ----> `‘defg’`
So Let’s Solve this Problem

Input a string. If it has even length, then hyphenate in the middle:

baseball          base-ball

If it has odd length, then hyphenate around the middle character:

frisbee           fri-s-bee
Developing a Solution

Instead of just showing the solution, let’s “derive” the solution using a methodology that is called stepwise refinement.

The course is really about problem solving with the computer. So developing problem-solving strategies is VERY IMPORTANT.
“Reformat” the task.

Read in the string
Compute its length
if the length is even
   Hyphenate in the middle
else
   Hyphenate around around the middle character.

Still in English, but it looks a little more like python.
"Reformat" the task.

Read in the string
Compute its length
if the length is even
    Hyphenate in the middle
else
    Hyphenate around around the middle character.
Refine

s = input('Enter a string: ')  
n = len(s)  
if the length is even     
    Hyphenate in the middle  
else  
    Hyphenate around around the middle character.

We have turned the first two lines into python.
Refine Some More

```python
s = input('Enter a string: ')
n = len(s)
if the length is even
    Hyphenate in the middle
else
    Hyphenate around around the middle character.
```

How do we check if the value in `n` is even?
Refine Some More

```python
h = input('Enter a string: ')  
n = len(s)  
if n%2==0:  
    # s has even length  
    Hyphenate in the middle  
else:  
    # s has odd length  
    Hyphenate around around the middle  
character.
```

We add comments to summarize what we may assume about the value of n.
h = input('Enter a string: ') 
n = len(s) 
if n%2==0:
    # s has even length
    Hyphenate in the middle
else:
    # s has odd length
    Hyphenate around around the middle character.

Figure out the even-length hyphenation
Even-Length Hyphenation

We look at a small example.
These statements

\[
s = 'abcdef' \\
h = s[0:3] + '-' + s[3:] \\
\]

assign ‘abc-def’ to \( h \).

In general:

\[
m = n/2 \\
h = s[0:m] + '-' + s[m:] \\
\]
Refine Some More

```python
h = input('Enter a string: ')
n = len(s)
if n%2==0:
    # s has even length
    m = n/2
    h = s[0:m] + '-' + s[m:]
else:
    # s has odd length
    Hyphenate around the middle character.
```
Refine Some More

define some more

```python
h = input('Enter a string: ')  
n = len(s)  
if n%2==0:
    # s has even length
    m = n/2
    h = s[0:m] + ' ' + s[m:]
else:
    # s has odd length
    Hyphenate around around the middle character.
```

Figure out the odd-length hyphenation
Odd-Length Hyphenation

We look at a small example.
This
   \[ s = \text{‘abcdefg’} \]
   \[ h = s[0:3] + \text{‘-’} + s[3] + \text{‘-’} + s[3:] \]
assigns ‘abc-d-efg’ to \( h \).

In general:

\[
\begin{align*}
m & = n/2 \\
h & = s[0:m] + \text{‘-’} + s[m] + \text{‘-’} + s[m+1:] 
\end{align*}
\]
```python
h = input('Enter a string: ')
n = len(s)
if n%2==0:
    # s has even length
    m = n/2
    h = s[0:m] + '-' + s[m:]
else:
    # s has odd length
    m = n/2
    h = s[0:m] + '-' + s[m] + '-' + s[m+1:]
```
Summary

1. A Boolean expression evaluates to either True or False

2. A Boolean expression is made up of comparisons that are either True or False

3. The and, or, not operations combine boolean values

4. Various if constructions can be used to organize conditional execution.
boolean

A primitive type whose values are True and False.