2019

Python if statements

Natalia Novak
Bronx Community College, City University of New York

How does access to this work benefit you? Let us know!

Follow this and additional works at: https://academicworks.cuny.edu/bx_oers

Part of the Other Computer Sciences Commons

Recommended Citation
https://academicworks.cuny.edu/bx_oers/34

This Lecture or Presentation is brought to you for free and open access by the Bronx Community College at CUNY Academic Works. It has been accepted for inclusion in Open Educational Resources by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@cuny.edu.
If-else

Topics to be covered:

• If-else branches (general)
• If-else statement
• Equality and relational operators
• More if-else

Additional topics:

• Boolean operators and expressions
• Membership and identity operators
If-else branches (general)

In many circumstances when we write a program we need the ability to check conditions and change the behavior of the program accordingly.

*Selection statements* or *conditional statements*, give us this ability.
If-else branches (general)

In many circumstances when we write a program we need the ability to check conditions and change the behavior of the program accordingly.

*Selection statements* or *conditional statements*, give us this ability.

**Example:** Let’s look through the following code

```python
if my_class_average > 1:
    print("I passed the class! Hooray!"")
else:
    print("Bummer! I will have to re-take this class!")
```
If-else branches (general)

Consider another code fragment:

```python
x = int(input("Enter an integer value:"))
y = int(input("Enter another integer value:"))

if x > y:
    a = x

if x < y:
    a = y

else:
    print("They are equal!")
```
If-else branches (general)

Consider another code fragment:

```python
x = int(input("Enter an integer value:"))
y = int(input("Enter another integer value:"))

if x > y:
    a = x

if x < y:
    a = y

else:
    print("They are equal!")
```

**conditions** (evaluated to a **Boolean value**: True or False)
If-else branches (general)

If we type the following commands in the Python shell, we will get the responses highlighted with blue

```python
>>> 2==2
True
>>> 2<3
True
>>> 3>7
False
>>> 5>9 or 2<3
True
```
If-else statement

Let’s write a program that will report the grade for the test, given a numeric score.

test_score = float(input("Enter test score:"))

if test_score >= 90:
    print("This is an A grade!")

if 80 <= test_score < 90:
    print("This is a B grade!")

if 70 <= test_score < 80:
    print("This is a C grade!")

if 60 <= test_score < 70:
    print("This is a D grade!")

else:
    print("Unfortunately this is an F grade")
If-else statement

Multi-branch if-else statements

Let’s write a program that will report the grade for the test, given a numeric score.

test_score = float(input("Enter test score:

if test_score >= 90:
    print("This is an A grade!"

if 80 <= test_score < 90:
    print("This is a B grade!")

if 70 <= test_score < 80:
    print("This is a C grade!")

if 60 <= test_score < 70:
    print("This is a D grade!")

else:
    print("Unfortunately this is an F grade")
Let’s write a program that will report the grade for the test, given a numeric score.

test_score = float(input(“Enter test score:”))

if test_score >= 90:
    print(“This is an A grade!”)
elif 80 <= test_score < 90:
    print(“This is a B grade!”)
elif 70 <= test_score < 80:
    print(“This is a C grade!”)
elif 60 <= test_score < 70:
    print(“This is a D grade!”)
else:
    print(“Unfortunately this is an F grade”)
Equality and relational operators

Equality operators

An equality operator checks whether two operands' values are the same (==) or different (!=x).

Note that equality is ==, not just =.

<table>
<thead>
<tr>
<th>Equality operators</th>
<th>Description</th>
<th>Example (assume x is 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>a == b means a is equal to b</td>
<td>x == 3 is true x == 4 is false</td>
</tr>
<tr>
<td>!=</td>
<td>a != b means a is not equal to b</td>
<td>x != 3 is false x != 4 is true</td>
</tr>
</tbody>
</table>

An expression evaluates to a Boolean value.
A Boolean is a type that has just two values: True or False.
A relational operator checks how one operand's value relates to another, like being greater than.

<table>
<thead>
<tr>
<th>Relational operators</th>
<th>Description</th>
<th>Example (assume x is 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>a &lt; b means a is less than b</td>
<td>x &lt; 4 is true, x &lt; 3 is false</td>
</tr>
<tr>
<td>&gt;</td>
<td>a &gt; b means a is greater than b</td>
<td>x &gt; 2 is true, x &gt; 3 is false</td>
</tr>
<tr>
<td>&lt;=</td>
<td>a &lt;= b means a is less than or equal to b</td>
<td>x &lt;= 4 is true, x &lt;= 3 is true, x &lt;= 2 is false</td>
</tr>
<tr>
<td>&gt;=</td>
<td>a &gt;= b means a is greater than or equal to b</td>
<td>x &gt;= 2 is true, x &gt;= 3 is true, x &gt;= 4 is false</td>
</tr>
</tbody>
</table>
Python supports *operator chaining*.  

**Example:** \(a < b < c\) determines whether \(b\) is greater-than \(a\) but less-than \(c\).  

Chaining performs *comparisons left to right*, evaluating \(a < b\) first.  
- If the result is true, then \(b < c\) is evaluated next.  
- If the result of the first comparison \(a < b\) is false, then there is no need to continue evaluating the rest of the expression.
Equality and relational operators

In-class work: see the handout, problems 1-5
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 78
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as \textit{nested if-else statements}.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 78
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 78
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 78

not an A grade
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 95
A branch's statements can include any valid statements, including another if-else statement, which are known as *nested if-else statements*.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 95
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```
Nested if-else statements

A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```
A branch's statements can include any valid statements, including another if-else statement, which are known as nested if-else statements.

```python
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

if grade = 95

that's an A
Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = 12`?
Multiple if statements

Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = 12`?
Consider the following code fragment:

```python
if num >= 10:
    print(“A”)
if num >= 0:
    print(“B”)
if num < 0:
    print(“C”)
if num < -10:
    print(“D”)
```

What would the program output if `num = 1`?
Multiple if statements

Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = 1`? B
Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = -1`?
Multiple if statements

Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = -1`? C
Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = -12`?
Consider the following code fragment:

```python
if num >= 10:
    print("A")
if num >= 0:
    print("B")
if num < 0:
    print("C")
if num < -10:
    print("D")
```

What would the program output if `num = -12`?
More if-else

In-class Activity

See exercises 6-7
A **Boolean** refers to a value that is either **True** or **False**. These two are constants in Python.

- we can assign a **Boolean value** by specifying **True** or **False**, 
  \[
  x = True
  \]
- an expression can evaluate to a **Boolean value** 
  \[
  y > 10
  \]
Boolean operators and expressions

**and** operator

The Boolean expression $a$ and $b$ is True if and only if both $a$ and $b$ are True.
Boolean operators and expressions

**and** operator

The Boolean expression \( a \text{ and } b \) is **True if and only if** both \( a \) and \( b \) are **True**.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a and b</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
The Boolean expression \(a \text{ and } b\) is **True** if and only if both \(a\) and \(b\) are **True**.

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(a \text{ and } b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

**Examples:** assume that \(a = 8\) and \(b = 3\), then the Boolean value of
1) \((a > 10) \text{ and } (b < 5)\) is **False**
2) \((a \neq 10) \text{ and } (b > 1)\) is **True**
or operator

The Boolean expression \( a \ or \ b \) is \textit{False if and only if} both \( a \) and \( b \) are \textit{False}. 
The Boolean expression \( a \text{ or } b \) is \textbf{False} if and only if both \( a \) and \( b \) are \textbf{False}.

\[
\begin{array}{|c|c|c|}
\hline
a & b & a \text{ or } b \\
\hline
\text{True} & \text{True} & \text{True} \\
\text{True} & \text{False} & \text{True} \\
\text{False} & \text{True} & \text{True} \\
\text{False} & \text{False} & \text{False} \\
\hline
\end{array}
\]
The Boolean expression \( a \ or \ b \) is **False if and only if** both \( a \) and \( b \) are **False**.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a</strong></td>
<td><strong>b</strong></td>
<td><strong>a or b</strong></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

**Examples:** assume \( a = 8 \) and \( b = 3 \), then the Boolean value of
1) \(( a > 10 ) \ or \ ( b < 5 )\) is **True**
2) \(( a == 10 ) \ or \ ( b < 1 )\) is **False**
The Boolean expression **not** a is **False** when a is **True**, and is **True** when a is **False**.
Boolean operators and expressions

**not** operator

The Boolean expression **not a** is **False** when **a** is **True**, and is **True** when **a** is **False**.

<table>
<thead>
<tr>
<th>a</th>
<th>not a</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
The Boolean expression \texttt{not a} is \texttt{False} when \texttt{a} is \texttt{True}, and is \texttt{True} when \texttt{a} is \texttt{False}.

\begin{center}
\begin{tabular}{|c|c|}
\hline
\texttt{a} & \texttt{not a} \\
\hline
\texttt{True} & \texttt{False} \\
\texttt{False} & \texttt{True} \\
\hline
\end{tabular}
\end{center}

\textbf{Examples:} assume \texttt{a = 8} and \texttt{b = 3}, then the Boolean value of
1) \texttt{not ( a > 10 )} is \texttt{True}
2) \texttt{not ( a * b > 20 )} is \texttt{False}
Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```python
if letter == 'a' or letter == 'b':
    print("Help!")
elif letter == 'c' or letter == 'd':
    print("We are in trouble!")
else:
    print("We are good!")
```
Consider the following code fragment:

```python
if letter == 'a' or letter == 'b':
    print("Help!")

elif letter == 'c' or letter == 'd':
    print("We are in trouble!")

else:
    print("We are good!")
```

If `letter = 'a'`, then we will get:
Consider the following code fragment:

```python
if letter == 'a' or letter == 'b':
    print("Help!")
elif letter == 'c' or letter == 'd':
    print("We are in trouble!")
else:
    print("We are good!")
```

If `letter = 'a'`, then we will get: `Help!`
Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```python
if letter == 'a' or letter == 'b':
    print("Help!")

elif letter == 'c' or letter == 'd':
    print("We are in trouble!")

else:
    print("We are good!")
```

If `letter = 'c'`, then we will get:
Consider the following code fragment:

```python
if letter == 'a' or letter == 'b':
    print("Help!")
elif letter == 'c' or letter == 'd':
    print("We are in trouble!")
else:
    print("We are good!")
```

If `letter = 'c'`, then we will get: **We are in trouble!**
The order in which operators are evaluated in an expression is known as **precedence** of operators.

<table>
<thead>
<tr>
<th>operator</th>
<th>description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>parentheses are evaluated first</td>
<td>$(2+5<em>3) – (5/6+2</em>4)$</td>
</tr>
<tr>
<td>+ - * / % // **</td>
<td>arithmetic operations next (in their order)</td>
<td>$10-2**5 &gt;= 10%7$</td>
</tr>
<tr>
<td>&lt; &lt;= &gt; &gt;= == !=</td>
<td>then comparisons and membership operators</td>
<td>$a &gt; 9$ and $b$ in $[1,2,3]$</td>
</tr>
<tr>
<td>not</td>
<td>negation operator next</td>
<td>not $(a &gt; 9)$ or $b == 2$</td>
</tr>
<tr>
<td>and</td>
<td>conjunction (and) next</td>
<td>$a &gt;9$ or $a &lt; 0$ and $b &gt; 1$</td>
</tr>
<tr>
<td>or</td>
<td>disjunction (or) last</td>
<td>$a &gt;9$ or $a &lt; 0$ and $b &gt; 1$</td>
</tr>
</tbody>
</table>
Example: Let’s evaluate the Boolean expression below for
\( g = 12, \ b = \text{True}, \ \text{and} \ a = 17 \)

\[ g \geq 90 \ \text{or} \ b \ \text{and} \ a > 100 \]
Example: Let’s evaluate the Boolean expression below for $g = 12$, $b = True$, and $a = 17$

$$g \geq 90 \text{ or } b \text{ and } a > 100$$

$$(g \geq 90) \text{ or } (b \text{ and } a > 100)$$
**Example**: Let’s evaluate the Boolean expression below for 

\( g = 12, \ b = \text{True}, \ \text{and} \ a = 17 \)

\( g \geq 90 \ \text{or} \ b \ \text{and} \ a > 100 \)

\[(g \geq 90) \ \text{or} \ (b \ \text{and} \ a > 100)\]

\( F \ \text{or} \ (T \ \text{and} \ F)\)
Order of evaluation

Precedence rules

**Example:** Let’s evaluate the Boolean expression below for $g = 12$, $b = True$, and $a = 17$

$$g \geq 90 \text{ or } b \text{ and } a > 100$$

\[
\begin{align*}
(g \geq 90) \text{ or } (b \text{ and } a > 100) \\
\text{F} \text{ or } (T \text{ and } F) \\
\text{F} \text{ or } \text{F} \\
\text{F}
\end{align*}
\]
Boolean operators and expressions

In-class work

Exercises 8-10
Membership and identity operators

Membership operators: in/not in

Quite often we need to check if a value can be or cannot be found within a container, such as a list or dictionary.

`in` and `not in` operators, known as *membership operators*, can help us!

**Example:**
```python
num = int(input("Enter an integer:"))
myContainer = [1,2,3,4,5,6,7]
if num in myContainer:
    print("Found it! It is in myContainer!")
else: print("Nope. It is not in myContainer.")
```
Membership and identity operators

Membership operators: in/not in

Example:
name = int(input("Enter a name:"))
MyNamesContainer = {
    "Maria": 23,
    "Anna": 19,
    "Jack": 5,
    "Alex": 12,
    "John": 18
}
if name in MyNamesContainer:
    print("Found it! It is corresponds to ", MyNamesContainer[name])
else: print("No such name in the container.")
Membership and identity operators

Membership operators: in/not in

Example:
name = int(input("Enter a name:"))
MyNamesContainer = {
    "Maria": 23,
    "Anna": 19,
    "Jack": 5,
    "Alex": 12,
    "John": 18
}

if name in MyNamesContainer:
    print("Found it! It is corresponds to", MyNamesContainer[name])
else: print("No such name in the container.")

Note that the keys are matched, not the values!
Identity operators: `is`/`is not`

Sometimes we want to determine whether two variables are the same object.

`is` and `is not` operators, known as *identity operators*, can help us out!

Identity operators return `True` only if the operands reference the same object (they do not compare object’s values).
Membership and identity operators

Identity operators: is/is not

Example:
myContainer = [1,2,3,4,5,6,7]
otherContainer = [9,8,7,6,5,4,3,2,1]

a = myContainer
b = otherContainer
a = b

if a is myContainer:
    print("a is myContainer!")

elif a is otherContainer:
    print("a is otherContainer!")
Identity operators: is/is not

Example:
myContainer = [1,2,3,4,5,6,7]
otherContainer = [9,8,7,6,5,4,3,2,1]

a = myContainer
b = otherContainer
a = b

if a is myContainer:
    print("a is myContainer!")
elif a is otherContainer:
    print("a is otherContainer!")
else:
    print("I have no idea that is a!")
Membership and identity operators

In-class work

Exercise 11
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)
else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)
print("That’s it!")
```
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)
else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)
print("That's it!")
```
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)
else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)
print("That's it!")
```
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)

else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)
print("That's it!")
```
Consider the following code fragment:

```python
if a > 5:
    myString = input(“Enter a word:”)
    print(myString*a)

else:
    myNum = int(input(“Enter an integer:”))
    print(myNum-a)

print(“That’s it!”)
```

Caution: be consistent!
Either use 4 spaces or a Tab (3 spaces)
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)

else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

```python
a = 3
```
Consider the following code fragment:

```python
a = 3
Enter an integer: 10
7
That's it!
```

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)

else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```
Consider the following code fragment:

```python
a = 6
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)
else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)
print("That's it!")
```
Consider the following code fragment:

```python
if a > 5:
    myString = input(“Enter a word:”)
    print(myString*a)
else:
    myNum = int(input(“Enter an integer:”))
    print(myNum-a)
print(“That’s it!”)
```

```
a = 6
Enter a word: my
mymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymymym}
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)

else:
    myNum = int(input("Enter an integer:")
    print(myNum-a)
    print("That's it!")
```

\[ a = 4 \]
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)
else:
    myNum = int(input("Enter an integer:")
    print(myNum-a)
print("That's it!")
```

```
a = 4
Enter a word: ten
tentententen
Enter an integer: 20
16
That's it!
```
Consider the following code fragment:

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)
else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)
print("That's it!")
```

Enter a word: ten
tentententen
Enter an integer: 20
16
That's it!
Code blocks and indentation

A conditional expression has the following form:

<expr_t> if <condition> else <expr_when_f>

Example:

print("A") if a < 10 else print("B")
A conditional expression has the following form:

<expr_t> if <condition> else <expr_when_f>

Example:

print(“A”) if a < 10 else print(“B”)

A conditional expression has three operands and thus is sometimes referred to as a ternary operation.
9.9 Conditional expressions

A conditional expression has the following form:

<expr_t> if <condition> else <expr_when_f>

Example:

x = 5 if a < 10 else x = 6
Conditional expressions

In-class Activity
This OER material was produced as a result of the CS04ALL CUNY OER project.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 4.0 License.