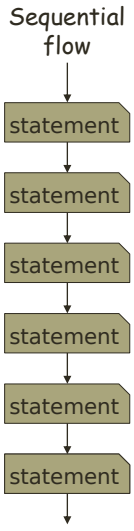
Branching and Decision Making

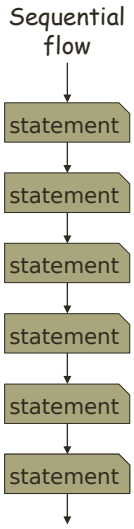
zyBooks
"Programming in Python 3"
chapter 4

if-else Branches



- Programs execute step-by-step by default
  - First statement, then second statement, then third, then fourth, ... to the end
  - "Sequential flow of execution"

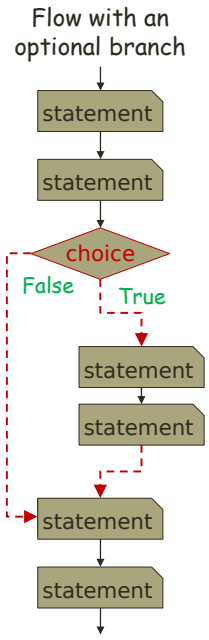
if-else Branches



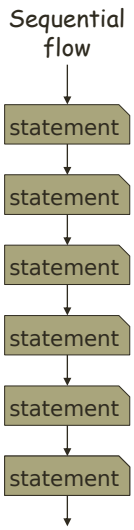
- Programs execute step-by-step by default
  - First statement, then second st… …hen third, then fourth, … d
  - "Sequential flow of execution"

input?

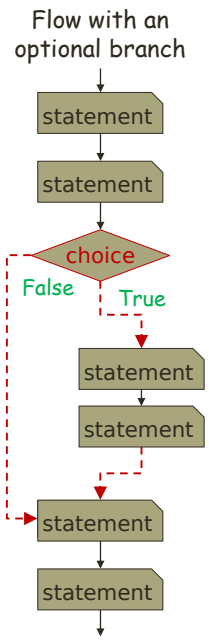
- ranching changes the …g processing?
  - Optionally skips some statements
  - Optionally chooses between statement groups
  - K… output?
    - also "else", "elif"



if-else Branches



- Programs execute step-by-step by default
  - First statement, then second statement, then third, then fourth, ... to the end
  - "Sequential flow of execution"
- Branching changes the sequential flow
  - Optionally skips some statements
  - Optionally chooses between statement groups
  - Keyword: "if"
    - also "else", "elif"



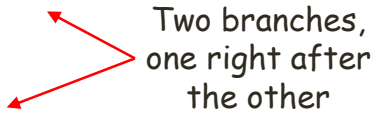
Simplest "if" Branches

```
prompt = 'Enter number between 1 and 10: '
number = int( input(prompt) )

if number < 1:
    print('Too small!')

if number > 10:
    print('Too big!')

print('You entered', number)
```



Details

- The "**`if`**" statement makes a choice based on an expression that is either True or False
- Following statement or statements are *indented*
  - *This is important!*
- All indented statements are *skipped* when the "**`if`**" expression is "False"
  - *i.e.* execute indented statements only "if" the expression is True
- The indented statements form a block
  - Sometimes called an "if clause" of statements

Examples:

- Enter a number
- Decide if it's positive or negative

- Enter two numbers
- Decide which is bigger

- Enter a string
- Decide if it's longer than 10 characters

Multi-Statement Clauses

- All indented statements after an "**if**" are part of the clause
- Clause ends at an **un**-indented statement
- Example:

```
x = float( input('number? ') )
if x == 0:
    print("Zero replaced by small number")
    x = 1e-10
print('Inverse is %f' % ( 1/x ))
print('Done.')
```

Slightly Bigger Example

```
x = float( input('1st number? ') )
y = float( input('2nd number? ') )
if x > y:
    # Swap the numbers:
    t = x
    x = y
    y = t
z = x / y
print('Ratio %f : %f equals %f' % (x, y, z))
print('Done.')
```

Choosing Between Groups of Statements

```
prompt = 'Enter number between 1 and 10: '
number = int( input(prompt) )

msg = 'You entered a '

if number <= 5:
    msg = msg + 'small '
else:
    msg = msg + 'big '

msg = msg + 'number.'
print(msg)
```

"if-else" skips one indented clause OR the other depending on the expression value

Multiple Choices

- Some situations need multiple decisions:
  - "if *this-is-true*, then do ...
    else *if that-is-true*, then do ...
    else *if the-other-is-true*, then do ...
    else do *some-default-statements...*
    ..."
- Handled by "**`if`** - **`elif`** - **`else`**" block
  - The most general form of "**`if`**" statement
  - "**`elif`**" can be repeated as often as needed
  - Single "**`else`**" clause at the end
    - provides a default choice
  - "**`elif`**" and "**`else`**" clauses are *optional*

Example

```
color = input('Enter a color: ')
if color == 'red' \
   or color == 'green' \
   or color == 'blue':
    palette = 'a primary color.'
elif color == 'cyan' \
     or color == 'magenta' \
     or color == 'yellow':
    palette = 'a primary pigment.'
elif color == 'black' or color == 'white':
    palette = 'monochrome.'
else:
    palette = 'not a primary color.'
print('Your color is', palette)
```

try – three-way temperature converter

- Input temperature, space, and units
  - Examples: 100 F
    30 C
    400 K
- Split into temp. part and units part
  - convert temp. part into float()
- if units equals 'F':
  - calculate Celsius, then Kelvin
- elif units equals 'C':
  - calculate Fahrenheit, and Kelvin
- elif units equals 'K':
  - calculate Celsius, then Fahrenheit
- else:
  - print error message

$F = 9/5\ C + 32$
$C = 5/9\ (F-32)$

$K = C + 273.15$
$C = K - 273.15$

Nested If Statements – If Within If

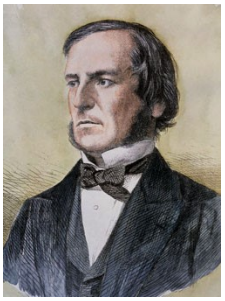
- Make a decision, based on previous decision
- "Pseudo-code" example:

```
If it's a weekday:
    if it's Monday, Wednesday, or Friday:
        go to MWF classes
    else:
        go to TR classes
else:       # weekend
    if before 10am:
        sleep some more
    else if before 4pm:
        have fun
    else if Sunday:
        back to studying
```

It's always 5 o'clock somewhere!
- Jimmy Buffet

Making Choices

- *Boolean expressions* – expressions that are either True or False
- Based on *Boolean operators* that make comparisons
- Boolean data type – only two values
  - **`False`**, corresponding to 0
  - **`True`**, corresponding to 1
- Named for George Boole
  - English mathematician and logician
  - 1815 - 1864
  - Introduced algebra of binary-valued systems, known as Boolean algebra



Boolean Operators

- Relational, *numeric* or *string* comparisons
  - Compare integers or floats to each other

==	is equal to	**!=**	is unequal to
<	is less than	**>=**	is greater or equal to
>	is greater than	**<=**	is less or equal to

- Boolean combinations
  - Operate on Boolean values, produce a Boolean

AND	OR	NOT

  - Operators can combine Boolean values into bigger, more complex Boolean-valued expressions

exercise

- work out truth tables for AND, OR, NOT, XOR

p	*q*	*p AND q*	*p OR q*	*p XOR q*	*NOT p*
		F	F	F	T
False	True	F	T	T	T
		F	T	T	F
True	True	T	T	F	F

Boolean Operators on Sequences

- Membership – is an object part of a sequence?
  - Applies to strings, lists, dictionaries
- Use "**in**" as a membership operator
- Example: get an answer from a user

```
answer = input('Do you want to do this? ')
if 'y' in answer or 'Y' in answer:
    print('Are you sure?')
elif 'n' in answer or 'N' in answer:
    print('Why not?')
else:
    print('Yes or No, please.')
```

Membership in a Dictionary

- "**in**" operator tests against the dictionary keys, not the values
- Test against the "**.values()**" method if desired

```
mydict = {'a': 'apple', 'b': 'banana', 'c': 'cherry'}

print( ('a' in mydict) )     # prints True
print( ('a' in mydict.keys()) )# prints True
print( ('apple' in mydict) )# prints False
print( ('apple' in mydict.values()) )# prints True

if 'banana' in mydict.values():    # yep
    print('I like banana splits!'):
    if 'ice_cream' not in mydict.values(): # nope
        print('Awww...')
    else:
        print('Oh boy!')
```

Identity Operator – Are Two Objects the Same?

- Operator "**is**" tests *identity* of objects
- When assigning a list to a variable, the variable only *refers* to the list - two variables can refer to the same list

```
listA = [ 'x', 'y', 'z' ]
listB = listA    # makes an additional reference to the list
print( (listA is listB) )# prints True
```

- *Copying* a list into another variable requires a *slicing* operation

```
listC = listA[ : ] # makes a new copy of the list
print( (listC == listA) ) # prints True
print( (listC is listA) ) # prints False
```

Not to worry, the "is" operator is rarely used

Order of Evaluation – What Gets Done First?

- As in math, some operators have higher *precedence*
  - These operators are applied in top-to-bottom order

Category	Examples	Description
Grouping	**() [] {}**	parenthesized expressions, tuples, lists, dictionaries
Exponentiation	*x* ** *y*	*x* raised to the power of *y*
Unary operators	-*x*	negation, etc.
Math operators	*x* / *y*	product, ratio, sum, difference, etc.
Relational operators	*x* >= *y*	equality, inequality, etc.
Boolean operators	*x* **and** *y*	True or False
Conditional evaluation	*x* **if** *tf* **else** *y*	yields *x* or *y*, depending on "*tf*" value

See https://montcs.bloomu.edu/Information/operator-precedence.C-Python-Pascal.html#Python_ops for complete list

Operator Chaining

- A special case of relational operators
  - Not in most programming languages
- Typical usage -

```
a = 7
b = 3
c = -5
if a >= b  and  b >= c:
  # do something...
```

- becomes

```
a = 7
b = 3
c = -5
if a >= b >= c:
  # do something...
```

- Chained relational operators are applied *left* to *right*
- `a`, `b`, `c` can be expressions
  - Normal precedence applies within the expressions
- "Short circuiting" - if the left-hand comparison is true, the right hand comparison isn't even performed
  - Rarely useful or important...

Indentation

- Code blocks - statements that act together
- Python recognizes code blocks by their shared indentation
  - Example:
    "if clauses" and "else clauses" are code blocks with common indentation relative to their "**`if`**" and "**`else`**" lines
  - The **`if`** and **`else`** lines must be lined up with each other
- Good programming editors assist with indentation
- *Warning! DON'T use both TAB characters and spaces to indent! This will confuse Python!*

Conditional Expressions

- A common construct looks like:

```
if condition:
    variableA = expressionT
else:
    variableA = expressionF
```

  - *i.e.* variableA gets one of two values depending on a choice of some sort
- Python offers a *shorthand* form:

```
variableA = \
        expressionT  if condition else  expressionF
```

  - expressions and condition follow the usual precedence rules, with the conditional assignment occurring after everything else

Conditional Example

- "Hailstone sequence" –
  - ```
    if (x_n % 2) == 0:
        x_n+1 = x_n / 2
    else:
        x_n+1 = 3*x_n + 1
    ```
- becomes
  - ```
    x_n+1 = x_n / 2 if (x_n % 2) == 0 else 3*x_n + 1
    ```
- or maybe
  - ```
    x_n+1 = (x_n/2) if ((x_n%2) == 0) else (3*x_n+1)
    ```
    ```