Objectives

- Conditional statements

Parts of an Algorithm

- Input, Output
- Primitive operations
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

Making Decisions

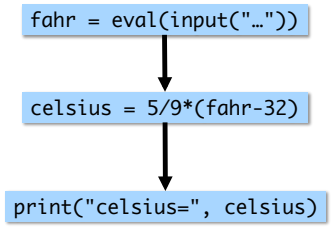
- Sometimes, we do things only if some condition holds (i.e., "is true")
- Examples
  - If the PB is new (has a safety seal)
    - Then, I will take off the safety seal
  - If it is raining and it is cold
    - Then, I will wear a raincoat
  - If it is Saturday or it is Sunday
    - Then, I will wake up at 9 a.m.
    - Otherwise, I wake up at 7 a.m.
  - If the shirt is purple or the shirt is on sale and blue
    - Then, I will buy the shirt

Conditionals

- Sometimes, we only want to execute a statement in certain cases
  - Example: Finding the absolute value of a number
    - |4| = 4
    - |-10| = 10
  - To get the answer, we multiply the number by -1 *only if it's a negative number*
  - Code:

```
if x < 0 :
    abs = x*-1
```

Typical Execution

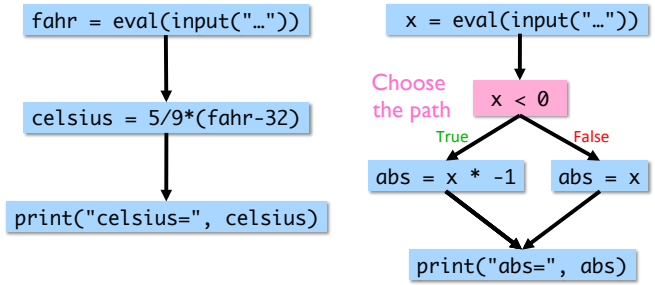


So far, we've thought of programs as a *sequence* of statements.

Statements execute *in order*.

`if` Statements

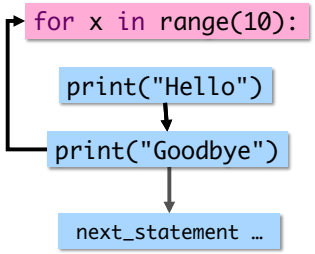
- Change the ***control flow*** of the program



Other Constructs That Change Control Flow

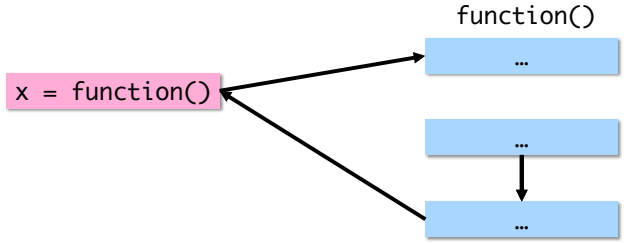
- **`for`** loops
  - Repeats a loop body a fixed number of times before going to the next statement after the **`for`** loop

```
for x in range(10):
    print("Hello")
    print("Goodbye")
next_statement …
```



Other Constructs That Change Control Flow

- Function calls
  - "Go execute some other code and then come back with the result"



Syntax of `if` statement:

Simple Decision

```
if condition :
    statement1
    statement2
    …
    statementn
```

keyword

"then" Body
- Note indentation

English Examples:

if it is raining **:**
    I will wear a raincoat

if the PB is new **:**
    Remove the seal

Conditions

- Syntax (typical, others later):
  - `<expr> <relational_operator> <expr>`
- Evaluates to either `True` or `False`
  - Boolean type

Relational Operators

- Syntax:
  - `<expr> <relational_operator> <expr>`

Low precedence

Relational Operator	Meaning
<	Less than?
<=	Less than or equal to?
>	Greater than?
>=	Greater than or equal to?
==	Equals?
!=	Not equals?

Use Python interpreter

Examples: Using Conditionals

- Determine if a number is even or odd

```
x = eval(input("Enter a number: "))
remainder = x % 2
if remainder == 0 :
    print(x, "is even")
if remainder == 1:
    print(x, "is odd")
```

evenorodd.py

Common Mistake:
Assignment Operator vs. Equality Operator

- Assignment operator: =
- Equality operator: ==

```
x = eval(input("Enter a number: "))
remainder = x%2
if remainder = 0 :
    print(x, "is even. ")
```

Syntax error

Syntax of `if` statement:
Two-Way Decision

English Example:

```
if it is Saturday or Sunday :
    I wake up at 9 a.m.
else :
    I wake up at 7 a.m.
```

keywords

```
if condition :
    statement1
    statement2
    …
    statementn
else :
    statement1
    statement2
    …
    statementn
```

"then" Body

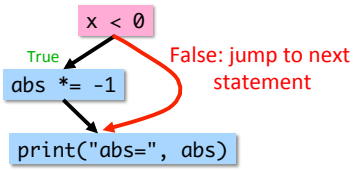
"else" Body

If-Else statements (absolute values)

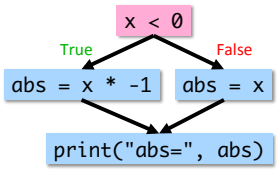
```
abs = x
if x < 0 :
    abs *= -1
print("abs=", abs)
```

```
if x < 0 :
    abs = x * -1
else:
    abs = x
print("abs=", abs)
```

If statement



If-else statement



Examples: Using Conditionals

- Determine if a number is even or odd
- More efficient implementation
  - Don't need to check if remainder is 1 because if it's not 0, it *must* be 1

```
x = eval(input("Enter a number: "))
remainder = x % 2
if remainder == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

Practice: Draw the Flow Chart

```
print("This program determines your birth year")
print("given your age and current year")
print()
age = eval(input("Enter your age: "))

if age > 120:
    print("Don't be ridiculous, you can't be that old.")
else:
    currentYear = eval(input("Enter the current year: "))
    birthyear = currentYear - age
    print()
    print("You were either born in", birthyear, end='')
    print("or", birthyear-1)
print("Thank you.  Come again.")
```

What does this code do?

Nondeterministic Decisions

- Sometimes, we don't want to necessarily know that a specific decision is always made
- For example, games often use randomness to make decisions
  - Roll dice
  - Coin flips
  - Location and behavior of baddies

How can we simulate coin flips?

Flipping Coins

- Simulate by randomly selecting between 0 (heads) and 1 (tails)
- Program: `coinFlip.py`

```
from random import randint

HEADS=0
TAILS=1

# flip the coin
if randint(0,1) == HEADS:
    print("heads")
else:
    print("tails")
```


Practice: Speeding Ticket Fines

- Any speed clocked over the limit results in a fine of at least $50, plus $5 for each mph over the limit, plus a penalty of $200 for any speed over 90mph.
- Our program
  - Input: speed limit and the clocked speed
  - Output: either (a) that the clocked speed was under the limit or (b) the appropriate fine

`speedingticket.py`

Looking Ahead

- Lab 3 due Friday
- Broader Issue: Self-driving Cars