

Outline for October 5, 2023

Reading: §2,5

Assignments: Homework 1, due October 12, 2023

Handout: “Truth Tables: and, or, and not”

1. Decision structures [*if0.py*]
 - (a) If statement
 - (b) Executes once, based on condition
 - (c) Syntax
2. Conditions
 - (a) Resolves to boolean value
 - (b) Literal booleans: True (1), False (0)
 - (c) Testable as `true` or `false`
 - (d) Relational operators
 - i. Use two arithmetic expressions connected with relational operators to create a boolean
 - ii. Relational operators: `>`, `>=`, `<`, `<=`, `==`, `!=`
 - iii. Precedence: resolved after arithmetic operators
 - iv. `6 > 2 + 3`; `"UCD" == "Sac State"`
3. Two-way decisions [*if1.py*]
 - (a) `if ... else` statements
 - (b) One condition, two possible code blocks
 - (c) Syntax
 - (d) `else` very powerful when the positive condition is easy to describe but not the negative
 - (e) String comparison example
4. Multi-way decisions [*if2.py*]
 - (a) Can execute code based on several conditions
 - (b) `elif` (else if)
 - (c) Syntax
 - (d) `else` only reached if all previous conditions false
 - (e) Nested if statements
5. Conditional expressions [*condexp.py*]
6. Iteration
 - (a) Definite loops: execute a specific (definite) number of times
 - (b) Indefinite loops: execute until a general condition is false
7. While loops [*while.py*]
 - (a) Contrast with `for`
 - (b) `break` causes program to fall out of loop (works with `for` too) [*loop1.py*]
 - (c) `continue` causes program to start loop over immediately (works with `for` too) [*loop1.py*]
8. For loops
 - (a) General form: `for i in iterator`
 - (b) *Iterator* is either list or something that generates a list

(c) Very common form: `for i in range(1, 10)`

9. `range()` in detail [*for.py*]

(a) `range(10)` gives 0 1 2 3 4 5 6 7 8 9

(b) `range(3, 10)` gives 3 4 5 6 7 8 9

(c) `range(2, 10, 3)` gives 2 5 8

(d) `range(10, 2, -3)` gives 10 7 4

10. Exception `Keyboard Interrupt` — user hit the interrupt key (usually control-C)

11. Program: counting to 10 [*toten.py*]

12. Program: sum the first 10 squares [*sumsq.py*]

13. Program: Fibonacci numbers [*fib.py*]