ECS 32A - Introductory Topics

Aaron Kaloti

UC Davis - Summer Session #1 2020

UC DAVIS
COMPUTER SCIENCE

First Remarks

This Slide Set is Not Comprehensive

- Much of the first three lectures (i.e. the Week 1 lectures) are not on these slides. Make sure to consult the lecture capture outlines that I post to Canvas here.

Note on Interpreter Mode Formatting

- In these slides, whenever I show Interpreter mode interactions, I use a Python interpreter that looks like Python IDLE (i.e. has the `>>>` prompt) but not like the REPL mode in the Mu editor (doesn't say "In" or "Out"). This shouldn't affect your understanding of Python.

print()

Escaping Quotation Marks

```
>>> print("He said, \"It's great.\"")
He said, "It's great."
>>> print('He said, "It\'s great."')
He said, "It's great."
```

Arithmetic Shorthands

Comparison

No Shorthand	With Shorthand
`x = x + 2`	`x += 2`
`x = x - 3`	`x -= 3`
`x = x * 7`	`x *= 7`
`x = x / 4`	`x /= 4`
`x = x // 4`	`x //= 4`
`x = x % 3`	`x %= 3`

Example: Safe Division[1]

```
val1 = int(input("Enter first value: "))
val2 = int(input("Enter second value: "))
if val2 == 0:  # if the user IS trying to divide by zero
    print("You can't divide by zero!")
else: # if the user is NOT trying to divide by zero
    quotient = val1 / val2
    print("{} / {} = {}".format(val1, val2, quotient))
```

safe_division.py

1. From the 06/25 lecture.

Example: Compared to 100[1]

Version #1

- Asks the user to enter an integer and tells them how that integer compares to 100.

```
num = int(input("Enter a number: "))
if num < 100:
    print(num, "is less than 100.")
if num == 100:
    print(num, "is equal to 100.")
if num > 100:
    print(num, "is greater than 100.")
```

1. From the 06/25 lecture.

Example: Compared to 100

Version #2 (Buggy Version)

- During the 06/25 lecture, it would seem that I incorrectly explained the purpose of this version. With this wrong version, if the user enters an integer that is less than 100, they will be told that the integer is both less than 100 *and* greater than 100. This is because the `else` is only "chained" to the `if num == 100` and not to the `if num < 100`.

```
num = int(input("Enter a number: "))
if num < 100:
    print(num, "is less than 100.")
if num == 100:
    print(num, "is equal to 100.")
else:
    print(num, "is greater than 100.")
```

Example: Guess the Secret Number

Version #1

```
secret_number = 18
guess = int(input("Guess the secret number: "))
if guess == secret_number:
    print("Congratulations! You guessed it.")
if guess != secret_number:
    print("WRONG")
```

Example: Guess the Secret Number

Version #2

- Tell them if their guess was too high or too low.

```
secret_number = 18
guess = int(input("Guess the secret number: "))
if guess == secret_number:
    print("Congratulations! You guessed it.")
# if guess != secret_number:
#     print("WRONG")
if guess < secret_number:
    print("Your guess was too low.")
if guess > secret_number:
    print("Your guess was too high.")
```

Example: Guess the Secret Number

Version #3

- Guess *one* of the *two* secret numbers.

```
secret_num1 = 17
secret_num2 = 20
guess = int(input("Guess one of the secret numbers: "))
if guess == secret_num1 or guess == secret_num2:
    print("You guessed a secret number!")
else:  # if guess != secret_num2
    print("You FAILED!")
```

Example: Age Range[1]

```
age = int(input("Enter your age: "))
if age <= 10:
    print("You are at most 10 years old.")
elif age <= 20:
    print("You are at most 20 years old.")
elif age <= 30:
    print("You are at most 30 years old.")
else:  # could be elif age > 30
    print("You are more than 30 years old.")
```

age_range.py

1. From the 06/25 lecture.

`pass`

- Use `pass` keyword if want empty body.

```
if x == 5:
    pass
else:
    pass
```

Chained Conditionals

Example #1

- If the `if x == 5` case is entered, the `elif x < 8` won't even be considered.

```
x = int(input("Enter number: "))
if x == 5:
    print("x equals 5.")
elif x < 8:
    print("x is less than 8.")
else:
    print("x is greater than or equal to 8.")
```

```
Enter number: 5
x equals 5.
```

```
Enter number: 4
x is less than 8.
```

```
Enter number: 7
x is less than 8.
```

```
Enter number: 8
x is greater than or equal to 8.
```

Chained Conditionals

Example #2

```
y = 30
if y == 30:
    print("AAA")
if y == 30:
    print("BBB")
elif y == 30:
    print("CCC")
else:
    print("DDD")
```

```
AAA
BBB
```

Chained Conditionals

Example #3

```
y = 30
if y == 30:
    print("XXX")
else:
    print("YYY")
if y == 30:
    print("BBB")
elif y == 30:
    print("CCC")
else:
    print("DDD")
```

```
XXX
BBB
```

Chained Conditionals

Rules

- Use of `if` indicates start of new chain.
- A chain can have at most one `else`, i.e. this fails:

```
if x == 3:
    pass
else:
    pass
else:
    pass
```

- A chain can have infinitely many `elif`'s, even if doesn't make sense:

```
if x == 8:
    pass
elif x == 12:
    pass
elif x == 12:  # will never be triggered
    pass
elif x == 12:  # will also never be triggered
    pass
```

Nesting

Example #1

```
val = int(input("Enter integer: "))
if val <= 50:
    if val == 20:
        print("val equals 20.")
    else:
        print("val is <= 50 but not equal to 20.")
else:
    if val == 80:
        print("val equals 80.")
    else:
        print("val is > 50 but not equal to 80.")
```

```
Enter integer: 27
val is <= 50 but not equal to 20.
```

```
Enter integer: 20
val equals 20.
```

```
Enter integer: 58
val is > 50 but not equal to 80.
```

```
Enter integer: 80
val equals 80.
```

```
Enter integer: 50
val is <= 50 but not equal to 20.
```

Nesting

Example #2

```
choice = input("Do you own a water bottle? (Enter 'y' or 'n'.) ")
if choice == "y":
    color = input("What color is it? ")
    if color == "green":
        print("Your water bottle is green.")
    else:
        print("Your water bottle is not green.")
elif choice == "n":
    print("You do now own a watter bottle.")
```

```
Do you own a water bottle? (Enter 'y' or 'n'.) n
You do now own a watter bottle.
```

```
Do you own a water bottle? (Enter 'y' or 'n'.) n
You do now own a watter bottle.
```

```
Do you own a water bottle? (Enter 'y' or 'n'.) y
What color is it? green
Your water bottle is green.
```

```
Do you own a water bottle? (Enter 'y' or 'n'.) blah
```

```
Do you own a water bottle? (Enter 'y' or 'n'.) y
What color is it? blue
Your water bottle is not green.
```

Logical Operators

- Can specify that multiple conditions have to be true with the logical operator `and`:

```
if num >= 3 and num <= 10:
    print("num is between 3 and 10, inclusive.")
else:
    print("num is either less than 3 or greater than 10.")
```

- Can specify that *at least* one of a set of conditions has to be true with the logical operator `or`:

```
if num == 2 or num == 5 or num == 7:
    print("num is either 2 or 5 or 7.")
else:
    print("num is neither 2 nor 5 nor 7")
```

- Can specify that the opposite of a condition has to be true with the logical operator `not`:

```
if not num == 2:  # same as if num != 2
    print("num does not equal 2")
```

Example: Basic Calculator

- Write a program that asks the user for three things:
  - one number
  - another number
  - a mathematical operation to perform, expressed as a string, e.g. "add", "subtract", "multiply", "divide". It can also be entered as all upper-case letters, e.g. "ADD", "SUBTRACT".
- The program should then output the result of the operation.

```
num1 = int(input("Enter first integer: "))
num2 = int(input("Enter second integer: "))
op = input("Enter operation: ")
if op == "add" or op == "ADD":
    print("{} + {} = {}".format(num1, num2, num1 + num2))
elif op == "subtract" or op == "SUBTRACT":
    print("{} - {} = {}".format(num1, num2, num1 - num2))
elif op == "multiply" or op == "MULTIPLY":
    print("{} * {} = {}".format(num1, num2, num1 * num2))
elif op == "divide" or op == "DIVIDE":
    print("{} / {} = {}".format(num1, num2, num1 / num2))
else:
    print("Invalid operation: {}".format(op))
```

Logical Operators

Short-Circuit Evaluation

- Recall:
  - With `and`: once a false condition is found, can ignore the conditions that follow.
  - With `or`: once a true condition is found, can ignore the conditions that follow.
- Usually doesn't affect anything; beneficial sometimes.

Example #1

- Only the highlighted condition (`3 < 5`) will be checked.

```
if 3 < 5 or 8 != 9:
    print("...")
```

Logical Operators

Short-Circuit Evaluation

Example #2

- Only the highlighted condition will be checked.

```
if 8 > 12 and "abc" == "abc":
    print("...")
```

Logical Operators

Short-Circuit Evaluation

Example #3: Example Where it Matters

- Results in crash:

```
if 5 / 0 == 3:
    print("...")
```

- Doesn't result in crash:

```
if 7 == 7 or 5 / 0 == 3:
    print("...")
```

Upcoming

- These next slides address two things:
  - How do the conditions of conditional statements work?
  - How do strings work with `<`, `<=`, etc.?

Boolean Values

What Relational Operators Output

```
>>> 10 < 100
True
>>> 55 > 100
False
>>> 100 > 55
True
>>> 99 <= 100
True
>>> 100 <= 100
True
>>> 100 < 100
False
>>> 100 == 100
True
>>> 100 != 100
False
```

Boolean Values

- `True` and `False` are boolean values; they represent a new type.

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
>>> x = True
>>> type(x)
<class 'bool'>
>>> type(5 < 6)
<class 'bool'>
```

- Putting them in quotes makes them strings, not booleans.

```
>>> type("True")
<class 'str'>
```

Conditional Statements: Layout

```
if condition:
    body
elif condition:
    body
elif condition:
    body
...
else:
    body
```

where `condition` evaluates to, or is, a boolean value.

Example

- These work:

```
if True:
    print("This will always be printed.")
if False:
    print("This will never be printed.")
```

String Comparisons

Motivating Example

- Not surprising that this works:

```
>>> "abc" == "abc"
True
>>> "abc" == "def"
False
>>> "abc" != "def"
True
```

- Might be surprising:

```
>>> "abc" < "abcd"
True
>>> "abc" < "Abcd"
False
>>> "ABC" < "AAD"
False
```

- What dictates if `True` or `False` is output in the above?

String Comparisons

- Focus question: How can a string be "less than" another string?

Rules

- A character can be "less than" another character. (see next slide)
- Strings are compared character-by-character. When two unequal characters are found, the character that is "less than" the other determines the string that is "less than" the other.

Example

- `"c"` is "less than" `"d"`.

```
>>> "abc" < "abd"
True
```

String Comparisons

Comparing Characters

- To determine if character X is less than character Y, apply the following rules[1]:
  - If X and Y are both lowercase letters or both uppercase letters, use the alphabet.
  - Uppercase letters are "less than" lowercase letters.
  - *Absence* of a character is "less than" every other character.

1. There are other rules (that you don't need to know), dictated by something called the ASCII table, which you should learn in at least one of ECS 36A, ECS 32C, or ECS 50.

String Comparisons

Example #1

```
if "ryan" < "bob":
    print("AAA")
elif "a" == "aaa":
    print("BBB")
elif "gray" <= "grey":
    print("CCC")
else:
    print("DDD")
```

```
CCC
```

String Comparisons

Example #2

```
name1 = "Aaron"
name2 = "Ryan"
name3 = "Bob"
if name1 == name2 or name1 == name3:
    print("AAA")
if name2 == "ryan":
    print("BBB")
elif name3 > name1:
    print("CCC")
elif name2 > name3:
    print("DDD")
```

```
CCC
```