Week 2 Review

Math

More math operators:

- Built in math operations: https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex
- `math` library has even more: https://docs.python.org/3/library/math.html

Library: A collection of code (functions, types...etc) that can be reused and shared.

- i.e. Someone else wrote the `math` library, we can `import` it into our code and use it.

```
x = 5 / 2       # x = 2.5
x = 5 // 2      # x = 2 (floored division)

x = abs(-5)             # x = 5, absolute value
x = max(2, 3, 4, 5)     # x = 5, return the max value
x = min(2, 3, 4, 5)     # x = 2, return the min value
x = pow(2, 3)           # x = 8, raise to the given power
x = 2**3                # same thing as pow! 2 to the 3rd pow
er = 8

import math
x = math.ceil(4.5)      # x = 5, round up
x = math.floor(4.5)     # x = 4, round down
```

Modulo: x modulo y = remainder of x divided by y

- NOTE: in Python, modulo has same +/- sign as denominator

```
# Today is a Tuesday (2)
x = 2 % 7               # 2 modulo 7, x = 2

# 15 days from now is a Wednesday (3)
x = (2 + 15) % 7        # 17 modulo 7, x = 3

# 15 days ago was a Monday (1)
x = (2 - 15) % 7        # -13 modulo 7, x = 1
```

Booleans and Conditionals

Booleans: A type that's either `True` or `False`

Conditional: Check whether some statement is `True` or `False`

if: do this if our conditional is `True`

elif: (else if) do this if the previous `if` was `False`, and the current conditional is `True`

else: do this if all the `if` and `elif` conditionals before were `False`

Comparisons:

- `<`, `>`, `<=`, `>=`
- `==`: equal to
- `!=`: not equal to
- `and`, `or`

```
# Create a function called "compare", that takes inputs "x" a
nd "y".
def compare(x, y):
        if x < y:
            print("x is less than y")
        if x > y:
            print("x is greater than y")
        if x == y:
            print("x is equal to y")

compare(x=1, y=2)
# Checks x < y -> True
# Prints "x is less than y"
# Also checks if x > y and if x == y!

compare(x=2, y=1)
# Checks x < y -> False
# Checks x > y -> True
# Prints "x is greater than y"
# Also checks if x == y!

compare(x=2, y=2)
# Prints "x is equal to y"
```

```
def compare(x, y):
    if x < y:
        print("x is less than y")
    elif x > y:        # Only evalutes if x < y is False
        print("x is greater than y")
    else x == y:       # Only evalutes if x < y is False AND
x > y is False
        print("x is equal to y")

compare(x=1, y=2)
# Checks x < y -> True
# Prints "x is less than y", and then skips to end function

compare(x=2, y=1)
# Checks x < y -> False
# Checks x > y -> True
# Prints "x is greater than y" and skips to end of function
```

```
compare(x=2, y=2)
# Checks x < y -> False
# Checks x > y -> False
# Prints "x is equal to y"
```

Example: Grades

```
def getGrades1(score):
    if score >= 90 and score <= 100:
        print("Grade: A")
    elif score >= 80 and score < 90:
        print("Grade: B")
    elif score >= 70 and score < 80:
        print("Grade: C")
    elif score >= 60 and score < 70:
        print("Grade: D")
    else:
        print("Grade: F")

def getGrades2(score):
      # We can chain comparisons
    if 90 <= score <= 100:
        print("Grade: A")
    elif 80 <= score < 90:
        print("Grade: B")
    elif 70 <= score < 80:
        print("Grade: C")
    elif 60 <= score < 70:
        print("Grade: D")
    else:
        print("Grade: F")

def getGrades3(score):
    if score >= 90:
        print("Grade: A")
    elif score >= 80:
            # Don't need to check if score < 90 because
            # we already know it's true if the first conditio
n is False
        print("Grade: B")
    elif score >= 70:
        # Don't need to check if score < 80 because
        # we already know it's true if the above conditional
is False
        print("Grade: C")
    elif score >= 60:
        print("Grade: D")
    else:
        print("Grade: F")

# These 3 functions do the same thing!
getGrades1(70)
getGrades2(70)
getGrades3(70)
```

Comparing Strings

Also use the `==` and `!=` operators

```
answer = input("Do you agree? ")
if answer == "yes":
    print("Agreed")
else:
    print("Not agreed")
```

If I type `"YES"` it will print `"Not agreed"` ... that doesn’t seem right.

Cleaning up our input to make the program smarter.

```
# Stripping white spaces and lower case everything
answer = input("Do you agree? ").strip().lower()

# Version 1
if answer == "yes" or answer == "y":
    print("Agreed")
else:
    print("Not agreed")

# Version 2
if answer.startswith("y"):
    print("Agreed")
else:
    print("Not agreed")
```

Even or Odd

Variable Scope: where in the code a variable exists

- a parameter variable (`n`) only exists in its function `is_even`

```
def main():
    x = int(input("What's x? "))
    # n doesn't exists here!
    ans = is_even(x)
    print(f"Is {x} even? {ans}")

def is_even(n):
        # x doesn't exists here!
    if n % 2 == 0:
        return True
    else:
        return False

main()
```

Match/Case

What if I want to do something variable matches a value, and do something else if the variable matches a different value?

Match: select variable we want to match

Case: Checks if the match variable equals this value. If it does, execute the code block below it.

```
name = input("What's your n
ame? ")

if name == "Harry":
    print("Gryffindor")
elif name == "Hermione":
    print("Gryffindor")
elif name == "Ron":
    print("Gryffindor")
elif name == "Draco":
    print("Slytherin")
else:
    print("Who?")
```

```
name = input("What's your n
ame? ")

match name:
    case "Harry":
        print("Gryffindor")
    case "Hermione":
        print("Gryffindor")
    case "Ron":
        print("Gryffindor")
    case "Draco":
        print("Slytherin")
    case _:
        print("Who?")
```

Case can also match multiple values like this

```
match name:
    case "Harry" | "Hermione" | "Ron":
        print("Gryffindor")
    case "Draco":
        print("Slytherin")
    case _:
        print("Who?")
```