Week 1 Review

Prints, Inputs, and Variables

Console: Computer interface that we're running python in.

2 ways to run python:

1. Create a python file, write our code in it, and run the file with `python myFile.py` in our console.
2. **Interactive mode**: Just type `python` into console, write + execute code one line at a time.

Print: function that prints a string to our console

```
print("hello")
```

Comments: lets me write notes in my code, but the computer won't execute it

Variables: lets me store values and gives it a name so we can use it more than once

```
# Create a new variable called "name", and give it a value.
name = "Tiffany"
# This will print "hello," with a new line, and then name
print("hello,")
print(name)
```

What can we do with print?

```
# We can use the + sign to combine multiple strings first.
print("Hello Ms. " + name)
```

```
# This also works! We gave print 3 string inputs instead of 1.
# NOTE: print automatically added a space in between each input
print("Ms.", name, "Welcome!")
```

Function: a block of code that we packaged and named so we can reuse it

- `print` is a function someone else wrote, we can use it without worrying about how print works

Parameter: Variables that are inputs to a function

Argument: Values that we assign to parameters when we call the function

Positional Argument:

- Argument gets passed into function without specifying which parameter it's for
- Computer will assume arguments are given in the same order as parameters in function definition

Keyword Argument:

- When we call a function, we tell it exactly which argument is for what parameter

```
# Print has an input called "end" that we can change. By defa
ult, it is a new line.
# We don't want print to add a new line at the end.
# Let's make it an empty string instead of the default.
print("hello, ", end="")
print(name)
# Now it prints "hello, " and name without a new line.

# Print also has an input called "sep", which is inserted bet
ween each input string.
# By default it's a space. Let's change that too.
```

```
print("hello", name, "welcome", sep=", ", end="!")
# Now it prints "hello, Anika, welcome!"
```

- `end=` is an example of a keyword argument
- `"hello, "` is an example of a positional argument
- `end` is the name of a parameter
- `""` is the argument we're passing into the `end` parameter

Input: function that prints a prompt in console, and waits for user to respond

```
name = input("What's your name? ")
# name = whatever the user typed (but as a string)
```

Functions

So far we've used functions other people wrote for us (`print`, `input`).

How to write my own function?

```
def hello():
    print("hello")
    print("doing complicated stuff...")

hello()
hello()
hello()
```

```
def hello(to):
    print("hello", to)

hello("Alice")         # positional argument
```

```
hello("Bob")           # positional argument
hello(to="Charlie")    # keyword argument
```

```
def hello(to="world"):
    print("hello", to)

hello()             # Use default argument "hello world"
hello("Alice")      # positional argument, "hello Alice"
hello(to="Bob")     # keyword arugment, "hello Bob"
```

return: output something from the function so we can use it where we called the function

```
def hello(to):
        # Don't print here, just return a string
        return f"hello {to}!"

name = input("What's your name? ")
welcome_message = hello(name)
# Now print the welcome message
print(welcome_message)
```

Main() Function

Functions have to be defined first before we can use it!

```
# Code starts executing here
hello()
# Uh-oh...hello doesn't exist yet! it's defined below
# Console shows:
# NameError: name 'hello' is not defined.

def hello():
```

```
    print("hello")
    print("doing complicated stuff...")
```

If we have a bunch of function definitions, we have to scroll all the way down a file to find where we actually start executing code.

main(): Put our top level code into this function, and call it after all the other functions are defined.

```
def main():
        # Do it all in here!
    name = input("What's your name? ")
    hello(name)
    hi(name)
    welcome(name)

def hello(to="world"):
    print("hello", to)

def hi(to="world"):
    print("hi", to)

def welcome(to="world"):
    print("welcome", to)

# Call main after all the functions have been defined
main()
```

Strings

String: A type of object in Python that stores characters

Format String (f-string): special string that lets us insert variables anywhere in a string

```
# Concatenating a bunch of strings is not convenient
print("hello, " + name + ", welcome!")
# f-strings make it easy
print(f"hello, {name}, welcome!")
```

What can we do with strings?

- `strip()`: remove spaces at beginning and end of string
- `title()`: capitalize first letter of each word
- `split(sep=)`: splits string by a `sep` argument into multiple strings
- `upper()` and `lower()` : makes everything upper or lower case

```
name = "    aLicE sMith   ".strip().title()
print(name)
# prints "Alice Smith" with no spaces

name = "Alice, Smith"
first, last = name.split(", ")
# first = "Alice"
# last = "Smith"

name = "aLiCe sMiTh"
print(name.upper())    # ALICE SMITH
print(name.lower())    # alice smith
```

Lots more built in functions for strings documented here:

https://docs.python.org/3/library/stdtypes.html#string-methods

Ints, Floats, and Math

So far we can handle strings in Python. What if we need to handle numbers?

- **Int**: Integer type. Stores whole numbers, positive and negative
- **Float**: Floating point type. Stores numbers with decimals.

```
x = 1
y = 2

z = x + y       # addition, z = 3
z = y - x       # subtraction, z = 1
z = x * y       # multiplication, z = 2
z = x / y       # division, z = 0.5 (a float!)
```

We can do the same operations to floats. Note: integer division results in a float!

Casting: changing from one type to another

- NOTE: In python, a variable can be any type!

```
x = input("What's x? ")      # x is a string (123)
y = input("What's y? ")      # y is a string (456)

z = x + y             # What happens when I add two strings?
print(z)              # z = 123456, z is a string type

z = int(x) + int(y)   # Cast strings to int type
print(z)              # z = 579, z is an int type

z = float(z)
print(z)              # z = 579.0, z is a float type
```

Nested Function Calls: using output from one function as parameter for another function directly

```
x = int(input("What's x? "))
y = int(input("What's y? "))
z = x + y
```

round(number, ndigits): rounds a `number` to `ndigits` after zero. If `ndigits` is left blank, returns a whole number as int.

```
x = 3.14159
y = round(x)          # y = 3 (int)
z = round(x, 2)       # z = 3.14 (float)
```

Formatting Numbers As Strings

How to print numbers nicely? Use f-strings.

```
x = 123456789.123
# prints commas to separate thousands
print(f"{x:,}")     # "123,456,789.123"

# use f string to round decimals
y = 123.456789
print(f"{y:.2f}")   # "123.46"
```

Functions Again

```
def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n
```

```
main()
```