Intro to Python Class 3

Review

- Lists - appending, indexing
- Conditions - if, elif, else
- Loops - while, for

What we will cover this morning

- Functions
- Method calls
- (Even) More Data Types
- Python builtin functions

Functions

A named section of code that performs a specific task

This is also referred to as calling a function

We have already made a function call
when using the `type`, `int`, or `float` functions

```
a = '3'
print(type(a))
a = float(a)
```

Function calls

```
a = 3
print(type(a))
```

A function can take arguments

In the example above, the variable `a` is passed as an argument to the function `type`

Arguments can also be called parameters

```
# Some more function call examples

int('32')
str(32)
```

Creating a Function

The following example is a function definition.
This allows us to create our own functions

```
def print_plus_5(x):
    print(x + 5)
```

The function definition has the following parts

- The `def` keyword signifies we are defining a function
- The name of the function being defined - `print_plus_5`
- The arguments in parentheses - `x`
- The function body, which is a block of indented code that executes when the function is called. - `print(x + 5)`

What to name your arguments

Similar to for loops, you decide how to refer to your arguments

We just used x as our argument, but I can replace that with duck and the function will work the same

```
def print_plus_5(duck):
    print(duck + 5)
```

When naming your functions and your arguments, you should use terms based on the task at hand

Function returns

A function can also return a value

Return allows us to save the result and use it later

```
# function without return
def plus_5(x):
    x + 5

y = plus_5(4)
print(y)

#function with return
def return_plus_5(x):
    return x + 5

y = return_plus_5(4)
print(y)
```

Functions with no arguments

A function does not have to take arguments,
as in the following example:

```
def grass():
    print('The grass is green.')

grass()
```

This is useful when the function does some work but doesn't need any parameters.
i.e. The function is intended to always do the same thing

Functions with more than one argument

A function can also take more than one argument separated by commas. For example:

```
def personal_info(name, age):
    return "My name is " + name + " and I am " + str(age) + " years old."

my_information = personal_info("Amy", 26)
print(my_information)
```

Let's Develop It

Pick one of the following exercises

- Write a function that a bar could use to check to see if a customer is of drinking age.

```
isOfDrinkingAge(age)
```

- If it is a saturday in spring and it is above 65 degrees, Kermit plants flowers. Write a function to check whether he will plant flowers today.

```
willKermitPlantFlowers(dayOfWeek, season, currentTemperature)
```

- Zach knows something is wrong with his basement if the temperature is below 50 or the humidity is above 70%. Write a function that can check whether his basement has a problem.

```
basementIsOk(temperature, humidity)
```

Function Composition

Functions can also call other functions.

You can do this to break up tasks into small pieces and make your code easier to follow.

Function composition is when the output of one function acts as the input of another

```
def get_name():
    return input("What is your name?")

def get_email_address():
    return input("What is your address?")

def register():
    print("Please fill out the following information to register:")
    name = get_name()
    email = get_email_address()
    print(name, "is registered under the email address", email)
```

Method Calls

Methods are also called by name but are associated with a data type.

We have already called methods for strings and integers

```
a = 4
name = 'caleb'
sentence = 'the quick brown fox did the thing with the thing'
a_list = [1, 2, 3]

print(name.capitalize())
print(sentence.count('the'))

a_list.append(4)
print(a_list)
```

Data Type: Dictionaries

A dictionary is a data type of key/value pairs, defined with `{}`

```
menu_categories = {
    'food': 'stuff you eat',
    'beverage': 'stuff you drink',
}
```

Think of words in a dictionary. The words are keys and the definitions are values.

Indexing on Dictionaries

Dictionaries aren't literally just for definitions. They represent a group of mappings. A mapping might be: menu items -> costs.

We can also index on dictionaries, but using the key instead of integers like in a list

```
menu = {
    'tofu': 4,
}

tofu_cost = menu['tofu']
```

Indexing on a key that doesn't exist results in a KeyError

Dictionary Methods

Some of the most essential methods are `keys`, `values` and `items`

```
menu = {
    'tofu': 4,
    'pizza': 8,
    'baguette': 3,
}
print(menu.keys())
print(menu.values())
print(menu.items())
print(menu.get('pizza'))
print(menu.get('water'))
print(menu.get('juice', 5))
```

If you aren't certain a key is present, you can use the `get` method

`get` will return None if the key isn't present.

If you pass a second value to get, it returns that value when the key is not found.

The in operator

The `in` operator is used to determine if an element is in a given data type

For dictionaries, the keys are searched for the element.

```
color = [255, 255, 0]
if 0 in color:
    print("0 is in the color")

menu = {'tofu': 4}
print('tofu' in menu)
print(4 in menu)
```

Builtins for data types

Python provides several functions that help us work with these data types.

`len()`	Given a data type, return its length
`range()`	Create a list of integers in the range provided.
`sorted()`	Given a data type, returns a sorted copy of that data type
`enumerate()`	Returns a list of (index, element) from the list

Examples of using Builtins

```
# Using len() - Determines length
print(len([1, 2]))
print(len("Hello"))

# range() - Quickly creates a list of integers
print(range(5))
print(range(5, 10))
print(range(0, 10, 2))

# sorted() - Sort a given list
grades = [93, 100, 60]
grades = sorted(grades)
print(grades)
```

String Formatting Operators

```
print('To Do:')
to_dos = ['work', 'sleep', 'work']
# python 2
for item in to_dos:
    print('item is: %s' %(item))

# both
name = 'Sally'
print('Her name is {}'.format(name))

# python 3
print(f'Her name is {name}')
```

Function Practice

Write a function to...

```
# ...add 10 to any number
# ...divide any number by 100
# ...add any 2 numbers
# ...subtract any 2 numbers
# ...convert celsius to farenheight
# ...convert farenheight to celsius
# ...convert inches to centimeters
# ...convert centimeters to inches
# ...calculate sales tax for any given total amount
# ...calculate the area of a triangle (1/2 base * height)
# ...calculate the volume of a rectangular prism (length * width * height)
```

Questions?