Programming with Python
Part 2: Functions / Methods

Functions

- Remember math class? A function did what?

- In programming we use functions too! A function is a mini program that runs and returns a value to your program.

- `print()` is a function

The Print Function

print() is used to display output to the user

```
1  print("This is some text to output to the user.")
2
3
```

Using variables in print()

- `print()` is a function
- The `print()` function can also be used with variables and a combination of text and variables

```
1    message = "This is a Python course"
2    message1 = "What is your name?"
3
4    print(message)
5    print(message1 + " = Justin")
6
7    #Output
8    #This is a Python course
9    #What is your name? = Justin
10
```

type() Function

- Another built in Python function

- Returns the data type of the given data or variable

```
In [5]: type(611)

Out[5]: int
```

```
In [6]: type("This is a message!")

Out[6]: str
```

Errors in Python

- When programmers code errors are bound to happen
- Three types of errors
  - Syntax
  - Run Time
  - Logic

Syntax Error

- Error in the rules of the programming language

  e.g using a keyword as a variable name

```
while = 3
```

- Failing to include a required item like a ":"

```
def hello_funtction()
    print("hello world")
```

- Illegal operation

```
if x=3:
```

Run time error

- An error that causes a program to crash
- Properly syntax but an illegal operation
- E.g Divide by zero

```
x = 3/0
```

- Infinite loop

```
while True:
    print("hello")
```

Logical Error

- Hardest type of error to detect
- Code is correct, but does not produce the expected results

Example:

- The programmer thought **`x = 2 + y`** was the correct equation
- when **`x = y - 2`** was correct.
- The program will run, but produce unintended results.

The 'input' Keyword

User Input

- The main task programs need to perform is gathering input from the user

- In Python we use the `input()` function to gather input from the user

```
1  name1 = input("enter your name: ")
2  print(name1)
3
```

The program pauses and waits for user input.

Multiple Input Prompts

- By storing inputs in variables the program can prompt the user for multiple pieces of information.

- Simply use input statements in variable assignment statements.

```
1    #input multiple prompts
2
3    print("Please enter five grades. One at a time.")
4
5    grade1 = input("Grade 1:")
6    grade2= input("Grade 2:")
7    grade3= input("Grade 3:")
8    grade4 = input("Grade 4:")
9    grade5 = input("Grade 5:")
10
```

Input Data Type

- Input always returns a data type of **string**

- Sometimes Python needs some help with data types and we have to convert the type, such as from **Str**ing to **Int**eger for example.

- remember:
    - `input()` always returns type **`str`** (string)

Type Conversion

- Python assigns the data type automatically. But what if Python is using the data in a way we don't want?
- We must convert it using a **type conversion** where we **cast** the data type
- To cast a data type – use the specific Type keyword (like `int` or `str`) in front of the variable or data

```
1  sum_grades = int(grade1)+int(grade2)+int(grade3)+int(grade4)+int(grade5)
2
```

with a sum of grades stored as a number and calculated from 5 grades, the average can be calculated

average = sum / number of data points

```
1  average = sum_grades / 5
2
3  print("The average is: " + str(average))
4
```

What data type is average?
What does str(average) do?

Print formatting

- Besides string addition- we can concatenate in the print function with commas
- comma (**,**) method
- comma method combines
- printable items with a space
- Notice you can combine the comma with string addition

```
1   #input multiple prompts
2
3   print("Please enter five grades. One at a time.")
4
5   grade1 = input("Grade 1:")
6   grade2= input("Grade 2:")
7   grade3= input("Grade 3:")
8   grade4 = input("Grade 4:")
9   grade5 = input("Grade 5:")
10
11  sum_grades = int(grade1)+int(grade2)+int(grade3)+int(grade4)+int(grade5)
12
13  average = sum_grades / 5
14
15  print("The average is:" , str(average)+ ".")
16
17
```

String Fromating

There are method to format strings in Python.

`.capitalize()` - capitalizes the first character of a string

`.lower()` - all characters of a string are made lowercase

`.upper()` - all characters of a string are made uppercase

`.swapcase()` - all characters of a string are made to switch case

upper becomes lower and vice versa

`.title()` - each 'word' separated by a space is capitalized

User Defined Functions

- User defined functions are those functions which are created by the user and are not inbuilt in Python.
- System functions are those inbuild in Python e.g print(), input(), type() etc
- User defined functions are created by the user to do a specific tasks

User defined Functions

- Each user defined function must have the following:
  - Starts with the keyword **def**
  - Followed by the name of the function, with no spaces in between
  - Followed by parenthesis **( )** then a colon **:** immediately after
  - Inside the parenthesis, it can have one or more parameters. This is optional
  - The body of the function
  - The **return** keyword(also optional)
  - Indentation must be adhered to

Arguments and Parameters

- After a function you might have noticed there are always **( )**
- Sometimes there is data inside the parenthesis sometimes not
- Anything passed in the ( ) is called an **argument**

or a **parameter** when defining a function

- The `input()` function has 1 parameter: the message string to show the user
- `input()` takes 1 argument such as "what is your name? ", a string

Simple Functions

- To create a function use the `def` keyword.
- Each line of the function under the definition MUST be indented. This is how Python knows it is part of the function.
- To call a function, type the name of the function like a built-in function.
- function call: `get_name()`

```
1  def get_name():
2      name = input("What is your name?")
3      print("The name you entered is", name + ".")
4
5  print("This is a sample funciton")
6  get_name()
7
```

Function with Return

This simple function adds numbers, but **returns** a value to the function call

```
1   def add_numbers():
2       num1= input("Enter a number:")
3       num2 = input("Enter a second number:")
4       num3 = int(num1) + int(num2)
5       return num3
6
7   print("Welcome to the Magic!")
8   num4= add_numbers()
9   print(num4)
10
```

- In the `add_numbers()` function the user is asked for 2 numbers which are added together.
- The `return` keyword is used to send the result back to the `num4` variable

```
1  def add_numbers():
2      num1= input("Enter a number:")
3      num2 = input("Enter a second number:")
4      num3 = int(num1) + int(num2)
5      return num3
6
7  print("Welcome to the Magic!")
8  num4= add_numbers()
9  print(num4)
10
```

Code Reuse

We call the function more than once. This is the real power of a function

```
1  print("Welcome to the Magic!")
2  num4= add_numbers()
3  num5 = add_numbers()
4  print(num4 + num5)
5
```

So now the function is called twice and all 4 numbers are added together.

But the addition code was only written once.

Functions With Parameters

- Built-in functions like `print()` and `type()` take arguments.
- User-defined functions can define parameters that will be accepted

```
1  # define the function
2  def format_name(name_input):
3      print("the name you entered is", name_input + ".")
4
5  name1 = input("What is your name?: ")
6
7  #Call the function
8  format_name(name1)
9
```

Multiple Parameters

Separate each parameter with a comma.

```
1    # define the function
2    def format_name(name_input,age_input):
3        print("the name you entered is", name_input + ". You are", age_input, "years old.")
4
5    name1 = input("What is your name?: ")
6    age1 = input("What is your age?: ")
7
8    #Call the function
9    format_name(name1, age1)
10
```

The number of parameters passed must exactly match the number in the def statement or Python will throw an error. *Unless a default value is defined*.