Module 3: Functions and Methods

A function is a block of computer code that performs some action, may accept input arguments, and may return a value.

Built-In Functions

Some functions are a standard part of Python. We've encountered several of these functions so far.

What are the functions we've learned so far?

1.	
2.	
3.	
4.	
5.	
6.	

The full list of functions built into Python 3 is here: https://bit.ly/37Xxxmw

👉 **DO THIS:** What does the *len()* function do?

👉 **HINT:** When you start reading reference documentation you will encounter stuff you haven't learned yet. Focus on the bits you do know. Sometimes you'll need to dig into those new things, but it's perfectly ok to take a mental note that it's something to learn someday, and move on.

Library Module Functions

Python has a rich library of modules that you can include (import) into your programs to add more functions.

We will use a new command called *import* to load functions from a library module

The *math* Module

Reference: https://tinyurl.com/pyhi-math

Try these code fragments in the REPL:

Fragment	Evaluation Result

`from math import sqrt`
`sqrt(9)`	

`from math import pi`
`pi`	

💡 **NOTE:** `math.pi` is not a function, it is a constant value defined in the *math* library.

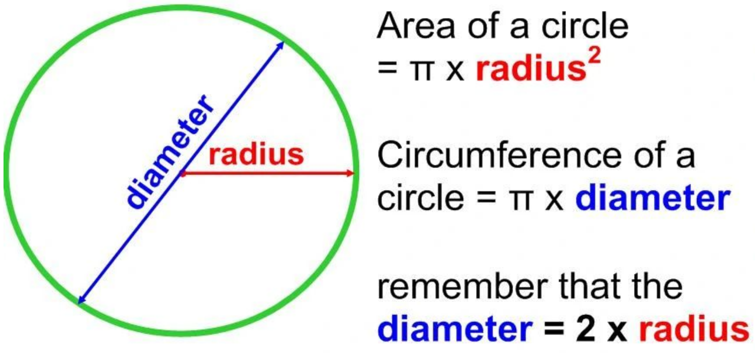
💡 **TAKEAWAY:** use the `import` statement to access a library module from your program.

Programming Example (pizza.py)

Enter and run the following program:

```
from math import pi

# Ask the user for the size (diameter) of the pizza.
ans = input("How big is your pizza (diameter in inches)? ")
d = int(ans)

# Calculate the radius from the diameter.
r = d/2

# Calculate the area from the radius.
a = pi * r**2

# Display the result.
print("That is", round(a, 1), "square inches of cheesy goodness")
```

For reference, here's the math that this program uses:



Area of a circle = π x **radius**^{2}

Circumference of a circle = π x **diameter**

remember that the **diameter = 2 x radius**

⁉ **QUESTION:** A 14" pizza is how many times bigger than a 10" pizza?

The *random* Module

Reference: https://tinyurl.com/pyhi-random

Try these code fragments in the REPL:

Fragment	**Evaluation Result**

`from random import randint`
`randint(1, 6)`	

`randint(1, 2)`	

💡 **NOTE:** in the REPL, you can press UP to recall previous command

⁉️ **QUESTION:** Where might `random.randint(1, 6)` be useful?

⁉️ **QUESTION:** Where might `random.randint(1, 2)` be useful?

Programming Example (diceroll.py)

Enter and run the following program:

```
from random import randint
print("Rolling two dice ...")
a = randint(1, 6)
b = randint(1, 6)
print("You rolled a", a, "and a", b)
print("Your total is", a+b)
```

👉 **DO THIS:** Once your program works, pair up with one classmate and play the Dice Roll game. You each roll a pair of dice and see who got the higher roll. First person to win two rolls wins the game.

User Defined Functions

You can make your own functions!

Functions can make your program more readable by grouping together all the statements needed to do one action. Plus functions allow you to reuse code you've written before.

Programming Example (pizza2.py)

Let's make another pass on the pizza.py program, this time using functions.

Enter and run the following program:

```
from math import pi

def input_int(prompt):
     '''Prompt for input, return the int value.'''
     ans = input(prompt)
     return int(ans)

def circle_area(diameter):
     '''Return the area of a circle calculated from its diameter.'''
     radius = diameter/2
     area = pi * radius**2
     return area

d = input_int("How big is your pizza (diameter in inches)? ")
a = circle_area(d)
print("That is", round(a, 1), "square inches of cheesy goodness")
```

A Python function:

- starts with the keyword *def*
  - is followed by the function name
  - then followed by a list of function parameters in parenthesis
  - and the line ends with a colon (:)
- the body of the function is indented
- a *return* statement is used to return a value back to the calling program

The triple-quote string is a third type – in addition to single(') and double(") quoted strings. The special thing about triple-quote strings is that they can be several lines long. A bare string at the start of a function that describes the function is called a *docstring*. Using docstrings is a good Python practice.

Object Methods

The things we've been talking about – like *int* and *str* – are called *objects*. Objects have *methods* which are actions – kind of like functions – but they work on the object.

str Methods

reference: https://tinyurl.com/pyhi-str

Try the following code fragments in the REPL to see how this works.

Fragment	**Evaluation Result**

`a = "Hello"`
`a.upper()`	

`"Hello".upper()`	
`"Hello".lower()`	
`"hello world".capitalize()`	
`"hello".upper().replace("LO", "P!")`	

The following code fragments show methods that help us inspect strings. Try them in the REPL:

Fragment	**Evaluation Result**
`"hello".startswith("h")`	
`"hello".startswith("H")`	
`"abc".isalpha()`	
`"123".isalpha()`	
`"abc123".isalpha()`	
`"abc".isdigit()`	
`"123".isdigit()`	
`"abc123".isdigit()`	

💡 **NOTE:** *True* and *False* are a new data type called a *bool*. It is used to store the result of yes/no decisions.