# Lesson 1

## Setting up Python

* Checking version

$ python –-version

* Python vs python3
* Python interactive shell

$ python3

## Uniquely Python

* Interpreted language
* Dynamically typed language
* Use of whitespace

## Print()

* String

>>> print("Hello, world!")

* Evaluated expression

>>> print(3+5)

* Combo

>>> print("3 + 5 =",3+5)

* Variable

>>> myVar = "I'm a variable!"

>>> print(myVar)

## Functions

* Hello()

>>> def hello():

... print("Hello, world!")

...

>>> hello()

* Arguments

>>> def greet(name):

... print("Hello,",name)

...

>>> greet("Zack")

* What can you do?
  + Uppercase (strings as arguments)

>>> def uppercase(myString):

... print(myString.upper())

...

>>> uppercase("Learning to program")

* + Double (numbers as arguments)

>>> def double(num):

... print(num\*2)

...

>>> double(16)

* + Sum (multiple arguments)

>>> def sum(num1, num2):

... print(num1+num2)

...

>>> sum(3,5)

* + Pretty sum (combining knowledge)

>>> def sum(num1, num2):

... print("num1 + num2 =",num1+num2)

...

>>> sum(3,5)

## Using Variables

* Variables are like a placeholder for ‘stuff’.
  + Strings

>>> myStringVar = "I'm a string"

>>> print(myStringVar)

* + Numbers

>>> myNumberVar = 4

>>> print(myNumberVar)

* They can be used in the same ways that ‘stuff’ can be used.
  + Numbers

>>> anotherNumberVar = 7

>>> print(4+anotherNumberVar)

* + Strings

>>> anotherStringVar = "happy birthday!"

>>> print(anotherStringVar.upper())

* They’re more flexible. Their values can change.

>>> myChangingVar = 4

>>> print(myChangingVar)

>>> myChangingVar = 7

>>> print(myChangingVar)

* Their values can even be reset with respect to themselves.
  + Like this

>>> x = 5

>>> print(x)

>>> x = x + 7

>>> print(x)

* + Or like this

>>> y = 3

>>> print(y)

>>> y = y + y

>>> print(y)

* Variables are a great way to store user input
  + Strings

>>> var = input("give a string input: ")

>>> print(var)

>>> type(var)

* + String of a number

>>> var = input("give a number input: ")

>>> print(var)

>>> type(var)

* + Actual number

>>> var = eval(input("give a number input: "))

>>> print(var)

>>> type(var)

* + Convert a string to a number

>>> var = input("give a number input: ")

>>> print(var)

>>> type(var)

>>> var = eval(var)

>>> print(var)

>>> type(var)

## Loops

* Loops allow us to repeat actions

>>> for i in range(5):

... print("Penny")

...

* Loops allow us to repeatedly act on a variable.
  + Note that we need to declare x outside of the loop since it should only be done once.
  + This is also important because of scope. Things declared inside a loop can’t be called outside a loop.

>>> x = 0

>>> for i in range(10):

... x = x + 3

... print(x)

...

* Loops use their own variable, but we can use that variable ourselves
  + In programming, we generally start counting at 0.

>>> for i in range(10):

... print(i)

...

* We can use a variable to count by adjusting one small detail:

>>> for i in range(10):

... print(i+1)

...

* Try squaring some numbers!

>>> for i in range(10):

... print(i\*i)

...

* Add up the numbers 1 – 20.

>>> sum = 0

>>> for i in range(20):

... sum = sum + (i+1)

...

>>> print(sum)

* Print statements can be helpful for checking that your loops are doing what you expect.

>>> sum = 0

>>> for i in range(20):

... print("current sum:",sum)

... sum = sum + (i+1)

... print("adding",i+1,"to sum")

... print("new sum:",sum)

... print("===================")

...

* Add up the numbers 5 to 12

>>> sum = 0

>>> for i in range(8):

... sum = sum + (i+5)

... print("+",(i+5))

...

>>> print(sum)

## Combining the Basics

* Temperature converter with input from terminal

>>> def convertCtoF():

... myCelciusTemp = eval(input("Enter the temperature in Celsius: "))

... myFahrTemp = (9/5)\*myCelciusTemp+32

... print(myCelciusTemp, "degrees Celsius is", myFahrTemp, "degrees Fahr")

...

>>> convertCtoF()

* Temperature converter with argument

>>> def convertCtoF(myCelciusTemp):

... myFahrTemp = (9/5)\*myCelciusTemp+32

... print(myCelciusTemp, "degrees Cel is", myFahrTemp, "degrees F")

...

>>> convertCtoF(15)

* Function within a loop within a function

>>> def doubling(x):

... return x\*2

...

>>> def repeateddoubling(y):

... for i in range(10):

... y=doubling(y)

... print (y)

...

>>> repeateddoubling(3)

# Lesson 2

## Variables

* We’ve already seen that variable assignments look like this:

x = 7

* Python allows us to make multiple assignments in one line:

x, y = 7, “k”

* See what this looks like combined with input commands:

x, y, z=input("x: "), input("y: "), input("z: ")

* The name that we give to a variable (or a function) must begin with a letter or an underscore, which may be followed by any sequence of letters, digits, or underscores. (No spaces allowed.) The only exceptions are the following ‘keywords’ that have specific purposes within the language of Python:
  + and
  + as
  + assert
  + break
  + class
  + continue
  + def
  + del
  + elif
  + else
  + except
  + False
  + finally
  + for
  + from
  + global
  + if
  + import
  + in
  + is
  + lambda
  + None
  + nonlocal
  + not
  + or
  + pass
  + raise
  + return
  + True
  + try
  + while
  + with
  + yield

## Loops and Lists

* For loops, so far we have seen the following syntax:

for i in range(10):

[loop behavior]

* range(10) basically tells python that the variable i is going to loop through values 0 – 9. However, we can tell python that the variable I should loop through a specific set of values like this:

for i in [1,3,7,8,9]:

[loop behavior]

* Note that

for i in range(5):

print(i)

* Behaves the same way as

for i in [0,1,2,3,4]:

print(i)

* This sort of variable [1,3,5,2] is called a list. We can create a list like this:

myList = [1,3,5,2]

* If we are looping through a list, the list doesn’t need to be written out in the loop syntax. We can call a list that already exists:

myList = [1,3,5,2]

for i in myList:

print(i)

## Writing Programs

* While it’s not technically necessary in Python, using a main block is common practice for many modern languages. Think of it as a function that contains the gist of your program. Like other functions, it’s just grouping a bunch of commands together. Put the commands of your program into a main block, and end your file with the command to run main.

def main():

[commands go here]

main()

* Comments are a key part of programming. You can’t be expected to remember all the details of everything you write, and other people looking at your project can’t be expected to know exactly what’s going on. You can include comments in your program by starting them with a #. # indicates to the interpreter that everything from # to the end of the line is for the programmer’s reference and can be ignored. You should still strive to write code that is clear and self-explanatory to a degree, but use comments to your advantage.

# I am a comment.

# I will be ignored by the interpreter.

* Python programs are stored in .py files. Name your programs accordingly.
* Here is a program that uses these principals:

# program.py

# prints "I’m a program!" to the terminal.

def main():

print("I’m a program!")

main() # This runs the function main()

* To run a program, navigate to the appropriate directory, use the command python3 and specify the program you want to run.

python3 program.py

## Exercises

* Write a program that prints “Hello, world!” to the terminal.

hello.py

* Write a program that computes the value of an investment after 10 years.
  + How do we calculate an investment with interest over time?
  + Which values do we need for this calculation?

futVal.py

* Modify futVal.py so users can choose how many years to calculate the investment for.

futVal2.py

* Write a program that asks for 3 values, stores them in a list, then uses a loop to sum them.

totalValues.py

* Write a program that converts Fahrenheit temperatures to Celsius.

convertFC.py

* Write a program that asks the user for 5 temperatures in Fahrenheit, then prints a table with each Fahrenheit temp on the left, and the associated Celsius temp on the right. Use a list and a loop!
  + What does a table look like? Type it out on word if it helps.
  + The string “\t” evaluates to a tab in Python.

convertFCTable.py

* Modify convertFCTable.py to use a function for the conversion.

convertFCTable2.py

# Lesson 3

## Ints vs Floats

* Ints and Floats are different data types for representing numbers. An int can represent an integer, while a float can represent a number with a decimal value.
* It takes a lot more work for a computer to store a float than an int.
* Python is dynamically typed, so it will generally figure out what kind of data type is best. In general, it will treat a number as an int until it becomes clear that decimals are needed.
* It is a good idea to understand the difference between these and when they are used, because they will sometimes behave differently.
* **Python’s number operations:**

|  |  |
| --- | --- |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Float division |
| \*\* | Exponentiation |
| abs() | Absolute value |
| // | Integer division |
| % | Remainder |

* Addition:
  + Int + int = int

>>> 9 + 3

12

* + Once there’s a float operand, the result is a float

>>> 9 + 3.0

12.0

>>> 9.0 + 3

12.0

>>> 9.0 + 3.0

12.0

* Subtraction:
  + Int - int = int

>>> 9 - 3

6

* + Once there’s a float operand, the result is a float

>>> 9 - 3.0

6.0

>>> 9.0 - 3

6.0

>>> 9.0 - 3.0

6.0

* Multiplication
  + Int \* int = int

>>> 9 \* 3

27

* + Once there’s a float operand, the result is a float

>>> 9 \* 3.0

27.0

>>> 9.0 \* 3

27.0

>>> 9.0 \* 3.0

27.0