Lecture 2 - converting values

Why Python instead of Java?

Python pays better.

Overview

You may want to refer back to the record video of this is lecture if you want to go back to it. We are going to cover a bunch of stuff - that is in a lot of chapters in the book, 1, 2, 3, some 4, some 5, some 6 and some 7 all in one set of examples.

Also the lecture notes are online in the lect-02 github. https://github.com/Univ-Wyo-Education/F21-1010/tree/main/class/lect/Lect-02

There is a ./conv directory that has a series of steps where you can go back to this and see the code as I develop it.

The "Process" that we often use

1. Collect Requirements
2. Produce a "design" / or a Requirements Document
3. Convert Requirements into a Prototype
4. Take the Prototype and build test data from it
5. Build the "application" from the Prototype
6. Build automated tests
7. Document results

Demo - of this in browser.

A lot of what happens when you program seems so simple - until you have to learn a non-human language. Programs are formal languages. English is an informal language. For example I can make a sentence that most of you will not understand, at first, but with some explanation I can show that it is using proper English crammer.

"The old man the boat."

In this context the old is a type of person. "man" is to get on board the boat and operate it. It is a verb.

So… The sentence is roughly equivalent to “The old people get on the boat and operate it.”

Python is a formal language. It uses a rigorous syntax. As humans we are not used to this.

One of the realities of development is that you are not using “one” tool. vim - for editing, or notepad++ on windows (don’t use notepad it will mess you up). VSCode - for debugging - and building and running projects. Python - command line for running programs.
Jupiter Notebooks (Iron Python) for mixing code and output in a human readable form. Different types of files. .py for python, .txt for text, .csv for comma separated values, .mk for markdown, images in .svg and .jpg and .png etc.

Software Engineering

This is a very “software engineering” approach to code development. Learning to code effectively is a “process”. Creativity tends to be innate - it is a talent. Programming is a set of skills.

Topics Covered

1. Files and Directories
2. Editing
3. Operators, `*` is multiply.
4. Other operators like `+`, `-`, `/`, `%`, and unary `-`. There are more.
5. `def` code reusability
6. Float, int and string data types
7. Basic testing
8. Functions - parameters - return values
9. if
10. Comparison for equality, `==` operator. Also `!=` not equal.
11. if / else
12. ‘:’ starts a block
13. Indentation
14. a = a + 1 - not algebra
15. Files
16. Import of files
17. Input
18. Output
19. Formatting of output

Requirements

Implement a python function that will convert from miles to kilometers and return that value.

Implement a program that will use the function, prompt for input in miles and then print out the result in kilometers.

Step 1

Convert from miles to kilometers.

Conversion generally is ((X + k1) * C) + k2

In our case k1 and k2 are 0. So we just get X * C

Demo - lookup conversion from miles to kilometers

```
# Step 1 - constants

miles = 3
conv = 1.60934
km = miles * conv
```

Demo - of this as a visualization

Step 2 - Input with error

```
# Step 2 - will error with type error

print ( "Enter Miles" )

miles = input()

conv = 1.60934
km = miles * conv
```

```
print ( "km = {}".format(km) )
```

Step 3 - Fixed error / Types

```
# Step 3 - inline after fixing type

print ( "Enter Miles" )

miles_str = input()
miles = int(miles_str)
conv = 1.60934
km = miles * conv

print ( "km = {}".format(km) )
```

Step 4 - Make a function

```
# Step 4 - After making a function

def mi_to_km ( mi ):
    conv = 1.60934
    km = mi * conv
    return (km)

print ( "Enter Miles" )

miles_str = input()
miles = int(miles_str)

km = mi_to_km(miles)

print ( "km = {}".format(km) )
```

Step 5 - Make Reusable Code

step-5.py:

```
# Step 5 - with function and a test.

import mi_to_km

print ( "Enter Miles" )

miles_str = input()
miles = int(miles_str)

km = mi_to_km.mi_to_km(miles)

print ( "km = {}".format(km) )
```

conv/mi_to_km.py:

```
# mi_to_km converts from miles as an integer or float to kilometers.
def mi_to_km ( mi ):
    conv = 1.60934
    km = mi * conv
    return (km)

# Automated Test
if __name__ == "__main__":
    n_err = 0
    x = mi_to_km ( 3 )
    if x !=  4.82802:
        n_err = n_err + 1

        print ( "Error: Test 1: conversion not working, expected {} got {}".format (  4
    x = mi_to_km ( 0 )
    if x != 0:
        n_err = n_err + 1
        print ( "Error: Test 2: conversion not working, expected {} got {}".format ( 0,

    if n_err == 0 :
        print ( "PASS" )
    else:
        print ( "FAILED" )
```

Step 6 - Add documentation

This is really a little step in this program - but a really important one for this class..

```
# Author: Philip Schlump
# Email: pschlump@uwyo.edu

# Main program to read in values and convert from miles (mi) to kilometers (km)

# Step 5 - with function and a test.

import mi_to_km

print ( "Enter Miles" )

miles_str = input()
miles = int(miles_str)

km = mi_to_km.mi_to_km(miles)

print ( "km = {}".format(km) )
```

This is the BEST time ever to be in this field

Blockchain, Growth, IoT, AI / Machine Larning!

https://youtu.be/eAn_oiZwUXA

Copyright