Lecture 7 - Testing with “if”.

Lectures

How if works in Python - https://youtu.be/wRNHlIesz_U
VSCode with Tests - https://youtu.be/_ovuvRjx_bE

From Amazon S3 - for download (same as youtube videos)

How if works in Python
VSCode with Tests

Podcast

Hisotry of Unix and its file system - https://youtu.be/9sHB4r4bqnA

From Amazon S3 - for download (same as youtube videos)

Hisotry of Unix and its file system

Code Examples that we walked through.

The beginning of a test - just a human testing it.

ex1.py:

```
def ex1 ( n ):
    if n <= 10:
        return n+2
    else:
        return 55

x = ex1(5)
print ( f"x={x}" )
```

Better tests - but still by hand.

Comments added.

Tests act as comments on how to call the function.

ex2.py:

```
# ex1 takes the numeric parameter 'n' and it is less than 11
# adds 2, otherwise it returns 55.
def ex1 ( n ):
    if n <= 10:
        return n+2
    else:
        return 55

# Examples of runing ex1
x = ex1(5)
print ( f"x={x}" )

x = ex1(11)
print ( f"x={x}" )
```

Start of automatic testing

Auto tests - partially developed.

ex3.py

```
# ex1 takes the numeric parameter 'n' and it is less than 11
# adds 2, otherwise it returns 55.
def ex1 ( n ):
    if n <= 10:
        return n+2
    else:
        return 55

# Automated Test - Version 1

x = ex1 ( -1 )
if x != 1:
    print ( "FAIL" )

x = ex1 ( 1 )
if x != 3:
    print ( "FAIL" )
```

```
x = ex1 ( 10 )
if x != 12:
    print ( "FAIL" )


x = ex1 ( 20 )
if x != 55:
    print ( "FAIL" )


print ( "PASS" )
```

Note that in ex3.py if a test fails you just get “FAIL” and at the end “PASS” so it can be misleading.

Better print statements but can still be misleading.

This now includes what is “expected” and what was returned.

```
# ex1 takes the numeric parameter 'n' and it is less than 11
# adds 2, otherwise it returns 55.
def ex1 ( n ):
    if n <= 10:
        return n+2
    else:
        return 55

# Automated Test - Version 1

x = ex1 ( -1 )
if x != 1:
    print ( f"FAIL expected 1 got {x}" )

x = ex1 ( 1 )
if x != 3:
    print ( f"FAIL expected 3 got {x}" )


x = ex1 ( 10 )
if x != 12:
    print ( f"FAIL expected 12 got {x}" )


x = ex1 ( 20 )
if x != 55:
    print ( f"FAIL expected 55 got {x}" )
```

```
print ( "PASS" )
```

Test with finished results - missing a case but the basic structure is correct.

ex5.py:

```
# ex1 takes the numeric parameter 'n' and it is less than 11
# adds 2, otherwise it returns 55.
def ex1 ( n ):
    # return -55555 # Uncomment to test automated test code.  All tests will fail.
    if n <= 10:
        return n+2
    else:
        return 55

# Automated Test - Version 2

n_err = 0
x = ex1 ( -1 )
if x != 1:
    print ( f"FAIL expected 1 got {x}" )
    n_err = n_err + 1

x = ex1 ( 1 )
if x != 3:
    print ( f"FAIL expected 3 got {x}" )
    n_err = n_err + 1


x = ex1 ( 10 )
if x != 12:
    print ( f"FAIL expected 12 got {x}" )
    n_err = n_err + 1


x = ex1 ( 20 )
if x != 55:
    print ( f"FAIL expected 55 got {x}" )
    n_err = n_err + 1


if n_err == 0 :
    print ( "PASS" )
else:
    print ( f"{n_err} test(s) FAILED!" )
```

Finished code with tests.

ex6.py:

```
# ex1 takes the numeric parameter 'n' and it is less than 11
# adds 2, otherwise it returns 55.
def ex1 ( n ):
    # return -55555 # Uncomment to test automated test code.  All tests will fail.
    if n <= 10:
        return n+2
    else:
        return 55

# Automated Test - Version 2

n_err = 0
x = ex1 ( -1 )
if x != 1:
    print ( f"FAIL expected 1 got {x}" )
    n_err = n_err + 1

x = ex1 ( 1 )
if x != 3:
    print ( f"FAIL expected 3 got {x}" )
    n_err = n_err + 1

x = ex1 ( 10 )
if x != 12:
    print ( f"FAIL expected 12 got {x}" )
    n_err = n_err + 1

x = ex1 ( 11 )
if x != 55:
    print ( f"FAIL expected 12 got {x}" )
    n_err = n_err + 1

x = ex1 ( 20 )
if x != 55:
    print ( f"FAIL expected 55 got {x}" )
    n_err = n_err + 1

if n_err == 0 :
    print ( "PASS" )
else:
    print ( f"{n_err} test(s) FAILED!" )
```