Lecture 13 - Functions in Python

YouTube

Intro - Functions - https://youtu.be/uaZQGlWs3eA
Example Functions - https://youtu.be/KrD_3d4s6AM

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

Intro - Functions
Example Functions

Why Functions are Important

1. Code re-use
2. Testability
3. Organization and Psychology
4. Readability and Documentation

Building a better "language" out on top of the existing language. Your "new" language is the set of actions (verbs) that you can apply to the problem at hand.

Code Reuse

For example you want a function that performs a calculation or set of operations.

Testing

There are two kinds of functions - pure means that it has no side effects. There are languages that implement "pure" functions an no other kind - like F# and Haskell. If you are to build pure functions in Python - you have to make certain that they don't have an side effects.

Examples of functions

```
##############################
def f1 ( a, b, c ):
    return a+b*c

x1 = f1 ( 2, 3, 4 )
print ( f"x1={x1}" )
```

```
##############################
def f2 ( a, b, c=10 ):
    return a+b*c

x1 = f2 ( 2, 3 )
print ( f"x1={x1}" )

##############################
def f3 ( *a ):
    t = 0
    for x in a:
        t = t + x
    return t

x1 = f3 ( 2, 3, 4, 5 )
print ( f"x1={x1}" )


##############################
def insulation ( w=15, l=47, q=15 ):
    tw_f=9
    tw_i=6
    tl=16
    sq_in = ( tw_f*12 + tw_i ) * ( tl * 12 )
    print ( f"sq in for studio {sq_in}" )

    sq_in_pack = w * l * q
    packs = sq_in / sq_in_pack
    return packs

ii = insulation ( )
print ( f"packs needed {ii}" )


##############################

# Accessing a global value - a "Non-Pure" fucntion with side effects

a = 4
b = 2

def hard_to_test ( ) :
    global a, b
    a = ( a + 2 ) % 5
    return b + a


x1 = hard_to_test()
print ( f"hard_to_test = {x1} first run" )
```

```
x1 = hard_to_test()
print ( f"hard_to_test = {x1} 2nd run " )

x1 = hard_to_test()
print ( f"hard_to_test = {x1} third run" )

x1 = hard_to_test()
print ( f"hard_to_test = {x1} forth run" )

x1 = hard_to_test()
print ( f"hard_to_test = {x1} fifth run" )

x1 = hard_to_test()
print ( f"hard_to_test = {x1} sixth run" )

x1 = hard_to_test()
print ( f"hard_to_test = {x1} seventh run" )
```