

Lecture 12

5.2 Functions (review)

5.5 Error Checking and Exceptions

5.2 Functions (review)

Functions serve as an ultimate control structure – they allow a series of complicated instructions to be encapsulated and then subsequently used as a single high-level operation.

Syntax:

```
def fname(parameters):  
    body
```

When creating a new function we need to consider:

- how will it be used
- what should be supplied to the function
- what information will the function return (if any)

5.2 Functions (review)

Example: let's write a program that computes the sum of squares of the first n positive integers, i.e. $\sum_{k=1}^n k^2$

If $n=3$, then $\sum_{k=1}^3 k^2 = 1^2 + 2^2 + 3^2 = 1 + 4 + 9 = 14$

If $n=5$, then $\sum_{k=1}^5 k^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 1 + 4 + 9 + 16 + 25 = 55$

And let's call it `sumOfSquares`

- **what should be supplied for the function?**

- **what should the function return?**

5.2 Functions (review)

Example: let's write a program that computes the sum of squares of the first n positive integers, i.e. $\sum_{k=1}^n k^2$

If $n=3$, then $\sum_{k=1}^3 k^2 = 1^2 + 2^2 + 3^2 = 1 + 4 + 9 = 14$

If $n=5$, then $\sum_{k=1}^5 k^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 1 + 4 + 9 + 16 + 25 = 55$

And let's call it `sumOfSquares`

- what should be supplied for the function?

positive integer value (n)

- what should the function return?

positive integer value

5.2 Functions (review)




Activity state diagram (flowchart) for function `sumOfSquares(n)`

5.2 Functions (review)

Let's write the definition of the function:

```
def sumOfSquares(n):  
    if type(n) != int:  
        print "The value ",n, "is not an integer value.",  
        print "Exiting the program ..."  
        return -1  
    if n < 1:  
        print "The value ",n,"is not positive.",  
        print "Exiting the program ..."  
        return -1  
    sq = 0  
    for k in range(n+1):  
        sq += k*k  
    return sq
```



A function call:

```
sumOfSquares(number)
```



5.2 Functions (review)

Let's write the definition of the function:

```
def sumOfSquares(n):  
    if type(n) != int:  
        print "The value ",n, "is not an integer value.",  
        print "Exiting the program ..."  
        return -1  
    if n < 1:  
        print "The value ",n,"is not positive.",  
        print "Exiting the program ..."  
        return -1  
    sq = 0  
    for k in range(n+1):  
        sq += k*k  
    return sq
```

formal parameter

actual parameter

scope of variable
n
(local scope)

A function call:

```
sumOfSquares(number)
```

5.2 Functions (new)

Optional Parameters

Recall function `range` : `range([start], stop[, step])`



optional parameters

Example:

```
def countdown(start=10, end=1):  
    for count in range(start, end-1, -1):  
        print count
```

This function counts down from 10 to 1 by default, but if the user supplies **start** or **end** values, it will count from the **start** value to the **end** value.

5.5 Error checking and Exceptions

Types of errors we met so far:

`NameError`

`TypeError`

`ValueError`

`AttributeError`

- belong to the `Exception` class.

Exceptions are used to report cases that are out of ordinary.

Usually, the result of an exception is to stop the execution of the code immediately.

For a programmer, the exception describes the immediate cause of a significant problem.

For a non-programmer, seeing such a reported error is not helpful (usually).

5.5 Error checking and Exceptions

Catching an exception (*try statement*):

```
try:  
    body  
except errorClass:  
    recoveryBody
```

Example:

```
...  
n = 0  
while not 1 <= n <= 10:  
    try:  
        n=int(raw_input('Enter a number from 1 to 10:'))  
        if n<1 or n>10:  
            print 'Your number should be from 1 to 10'  
    except ValueError:  
        print 'That is not a number.'
```

What does this program do?

5.5 Error checking and Exceptions

Raising an exception:

Example:

```
...
n = 0
while not 1 <= n <= 10:
    n=raw_input('Enter a number from 1 to 10:')
    if n.isdigit() == False:
        raise ValueError('it is not a number')

    n=int(n)
    if n<1 or n>10:
        print 'Your number should be from 1 to 10'
```

What does this program do?

See one more example in [sumOfSquares-exceptions.py](#)

Homework Assignment

HW8:

Read sections 5.2 and 5.5

Write a program that converts a numeric string (in given base) to the decimal number (base 10). Do the error checking. - exercises 5.26 and 5.35

Limit the range for the base to **2 <= base <=16**