

Python Lesson 2: Basics

vanderbi.it/py

Steve Baskauf



Digital Scholarship and Communications Office (DiSC)

- Unit of the Vanderbilt Libraries
- More online at: vanderbi.lt/disc
- Email: disc@vanderbilt.edu


Object name recommendations

- Be descriptive (what the object is or does)
- snake_case is specified by PEP 8:
 - `ordinary_relational_processes`
- camelCase is frequently used. Examples:
 - `bookList, alphabetizeParticipants` (lower CC for variables, functions)
 - `DocumentDescription, PageHeader` (upper CC for classes)
- Don't ever put spaces in any kind of name, even if you can get away with it.

Simple object types

- **string** literals. Enclose in quotes. Examples:
 `"lol", 'bye bye, birdie', "can't"`
 special characters with backslash: `'\n'`
- **number** literals (no quotes). Examples:
 `35`
 `0.999`
 `6.02`
- **boolean** (no quotes):
 `True` or `False`

Assignment to a variable

- The equals sign (=) assigns a value to a variable
- It's like a left arrow: 

```
user_name = "smithjr"  
is_door_open = False  
eulers_number = 2.7182818
```

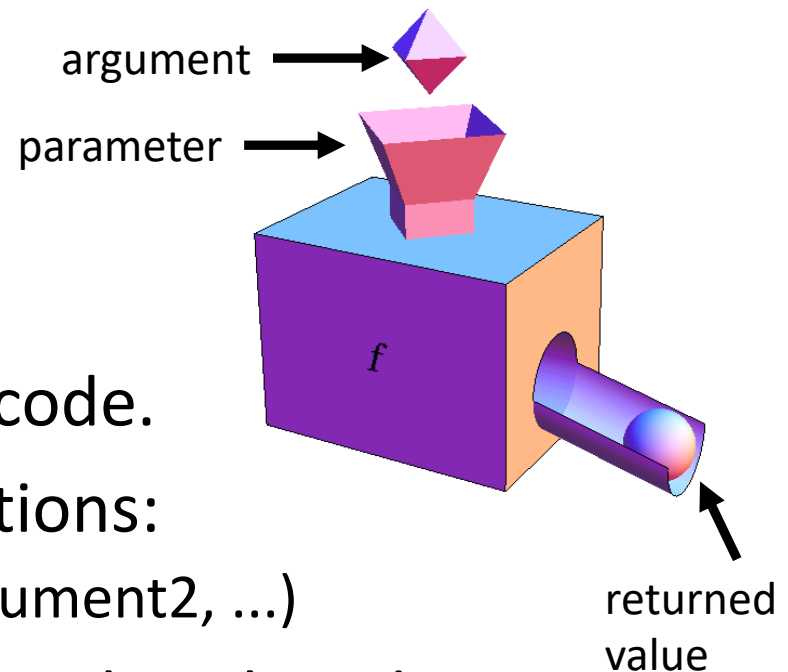
```
user_name = last_login_name  
sum = number_widgets + 3
```

```
too_many = sum > 10  
student_count = student_count + 1
```

- Variables can store many kinds of objects (not just simple ones like numbers and strings)

Try this...

Functions



- A **function** defines a block of code.
- We pass **arguments** into functions:
 - `functionName(argument1, argument2, ...)`
- It's good to name functions by what they do.
Example:

```
my_latte = make_latte(beans, milk, water)
```

- Functions can be:
 - built-in
 - defined by you in your code
 - defined by somebody else in a module

Try this...

Defining and calling functions

```
# here is where the function is defined
def multiplication(first_number, second_number):
    answer = first_number * second_number
    return answer
```

4 spaces standard

```
# here is where the function is called
num1 = 3
num2 = 5
answer = multiplication(num1, num2)
print(answer)
```

parameters

arguments

- Notes:

- The hash (#) character is used for comments
- Variables used for parameters and arguments can differ
- Indented code blocks: standard for Python is 4 spaces
- Don't forget colon before code block!!!
- About white space elsewhere

Try this...

- Notes:
 - Use a function when you need to repeat a task more than once
 - Use a function to keep your code in small enough blocks that it's easy to understand what's going on (importance of naming!)
 - Is it better to pile up functions inside of functions (compare first and second example)?

Modules (how)

- reusable code stored in a separate file (has **.py** extension like other Python programs)
- loaded into script using **import** statement
- dot notation for indicating a function is from a module (don't need the **.py** extension)

```
import simple_math
```

```
sum = simple_math.addition(num1, num2)
```

- abbreviating module names:

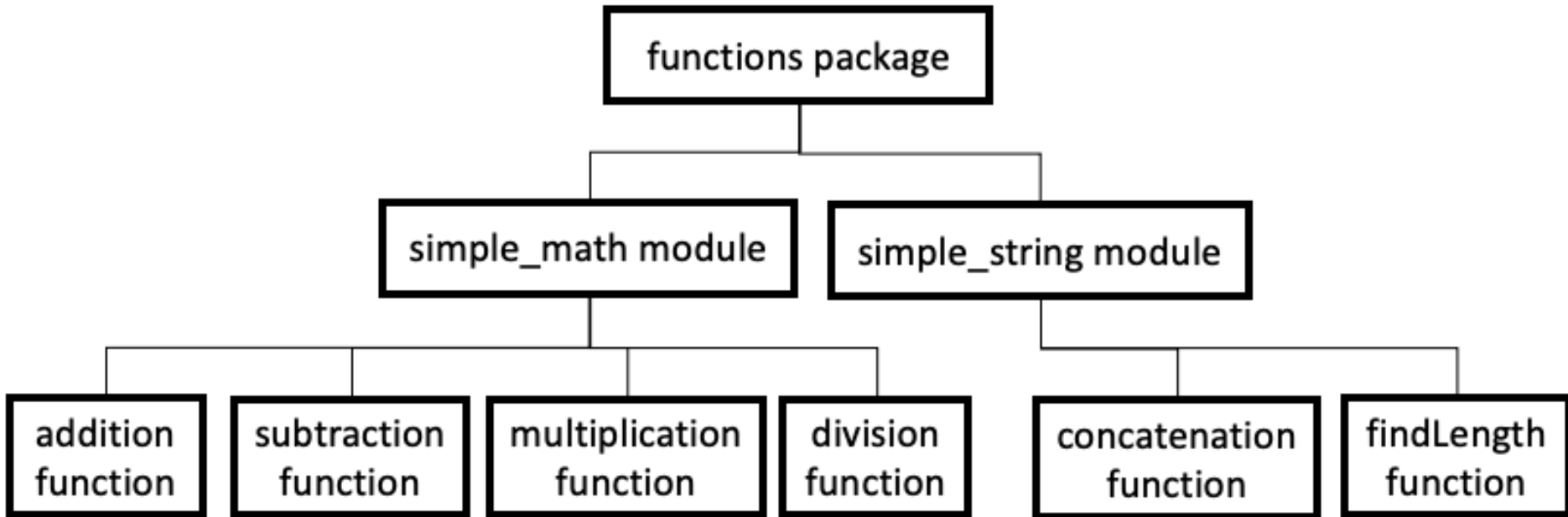
```
import simple_math as m
```

```
sum = m.addition(num1, num2)
```

Modules (where)

- some modules are part of the **standard library** (part of every Python installation)
- modules NOT in standard library must be loaded using a package manager (PIP for command line, GUI Tools menu on Thonny)
- Anaconda has most libraries already loaded
- You can make your own modules (but probably won't)
 - about the file name gotcha!

Packages (what)



- Packages are a high-level organizational tool for grouping related modules.
- Hierarchical dot notation:

`package.module.function()`

Packages (how)

- abbreviating module names:

```
from functions import simple_math
import functions.simple_string as st
```

```
answer = simple_math.subtraction(10, 3)
print(answer)
```

```
firstName = 'Donald'
lastName = 'Duck'
combinedString = st.concatenation(firstName,
lastName)
print(combinedString)
```

- There are linked DIY instructions if you want to try making your own packages and modules

Input function

- Example:

```
name = input("What's your name? ")  
print('Hello ' + name + '! How are you?')
```

- Content comes in as a string, so conversion is required if you want to input numbers. See example.

Conditional execution (Try this)

```
name = input('What is the name of the character? ')
is_micky = name == 'Mickey Mouse'
print(name)
print(is_micky)
```

```
if is_micky:
    print('You are a Disney character')
print('That is all!')
```

- The comparison operator (==) is different from the assignment operator (=) and produces a boolean.
- The **if** statement evaluates a boolean
- If **True**, the following indented code block is executed. (Don't forget colon!). Same indent is always executed.
- Notice how I named the variable to make the code readable.

else and **elif**

- **else** defines the default code block if no condition is satisfied.
- **elif** combines **else** and **if**; use to check additional conditions.
- Python does NOT have the **switch-case** structure common in other languages.

Try this...

- Examine and try **if...else...** and **if...elif...else...** examples.
- Notice how indentation is used to control which code blocks are conditionally executed and which ones are always executed.
- Notice that the program is really dumb. It only does what you say and doesn't really have any idea what a Disney character is.