INPUT

CSC 109 - Introduction to programming in Python - Fall 2023

Marcus Birkenkrahe

September 13, 2023

Contents

1 Understanding standard data streams

We want to write a program that

1. Says 'Hello world!'
2. Asks for your name

3. Greets you with your name
4. Tells you how many characters your name has
5. Asks for your age
6. Tells you how old you're going to be in one year

We're going to use this command sequence to learn a few functions useful to get input from the keyboard and manipulate text.

Check the `help` for `input` in the Python reference manual, or in the notebook, enter `?input` to get the *docstring* for the function.

```
>>> help(input)
Help on built-in function input in module builtins:

input(prompt=None, /)
    Read a string from standard input.  The trailing newline is stripped.

    The prompt string, if given, is printed to standard output without a
    trailing newline before reading input.

    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
    On *nix systems, readline is used if available.
```

Figure 1: Python help for keyboard input() function

What does this mean?

1. `input` reads a string from the keyboard or from a file (*stdin*)
2. If `input()` is used, the default `prompt` is missing (`None`)
3. If a prompt is used, it is printed without newline (*stdout*)
4. If CTRL-D (End Of File) is hit, an `EOFError` is raised.

Standard input, output and error are the three data streams: *[insert image `streams.png` here]*

Their standard direction is the screen but they can be redirected anywhere, e.g. into files:

```
touch hello  # create empty file 'hello'
ls -l   # list all files in long format
rm hello 2&>/dev/null  # send 'hello' to the black hole
ls -l
echo "Hello, world" > hello  # put string into new file 'hello'
cat hello  # view 'hello'
```

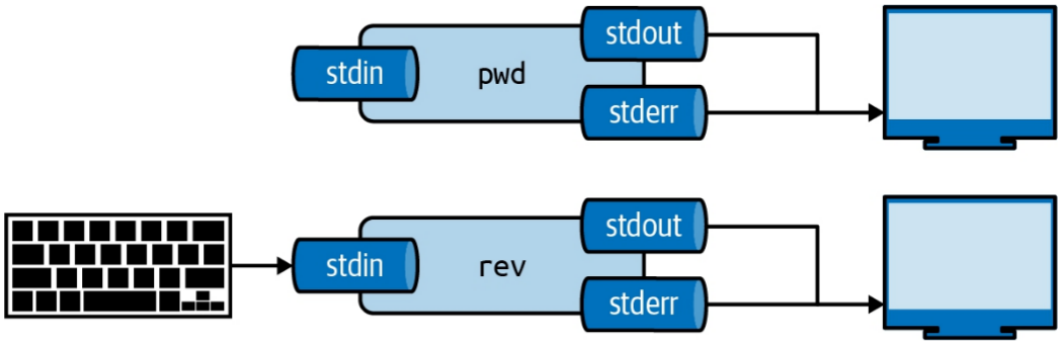

Figure 2: stdin, stdout, stderr for two shell commands

You can try this in the terminal - do you remember how to get to it?

2 Getting input from the keyboard

Step 1: Ask for user's `name` and print out the number of characters in the name.

```
print('hello world')
name = input('What is your name?' )
print('Good to meet you,' + name)
print('Your name has', len(name), 'characters')
```

Why did we not use the `+` operator in the last line? Try using it to put the strings together ('concatenate'):

```
name_length = 12
print('Your name has' + str(name_length) + 'characters')
```

Step 2: ask for user's age and print out age one year from now:

```
print('What is your age?')
age = input()
print('You are going to be ' + str(int(age) + 1) + ' years old')
```

3 Python script infrastructure

Not to forget about the Python script infrastructure:

1. You can save the Python code of your notebook as `.py` file
2. You can run the script on the terminal using Jupyter lab.

4 Getting keyboard input with a prompt

To save code, let's use the ability of `input` to display a `prompt` (as shown in the docstring with `?input`:

1. Put both programs in one code cell.
2. Use `input` to ask for the `name` and the `age`.
3. Print greeting with `name`, length of `name`.
4. Print `age` next year.
5. Sample run (terminal):

```
Hello, world!
What is your name? Marcus
Greetings, Marcus
Your name has 6 characters.
What is your age? 59
You are going to be 60 years old.
```

Figure 3: Testing input with prompt

Step 3: getting `input` with `prompt`:

```
print("Hello world!")
name = input("What is your name? ")
print("Good to meet you, " + name)
print("Your name has ", len(name), " characters")
age = input("What is your age? ")
print("You're going to be " + str(int(age) + 1) + " years old")
```

5 Getting two input values at once

Step 4: getting two input values at once with `split`:

```
print("Hello world!")
input_data = input("Enter name and age separated by a space: ")
name, age = input_data.split()
print("Good to meet you, " + name)
print("Your name has ", len(name), " characters")
print("You're going to be " + str(int(age) + 1) + " years old")
```

Check out the docstring of this new function with: `split?`.

- `split` is a string method - outside of `str` it has no meaning.
- You have to look for `str.split?` to get the docstring.
- Notice that `str.split()?` or `help(str.split())` throw errors.

6 Function preview

Functions in your code are like mini programs. We called six functions: `print`, `input`, `len`, `int`, `str`, `split`:

1. `print` prints its arguments but can also evaluate:

   ```
   print("Hi")
   print(5 + 5)
   ```

2. `input` takes input from the keyboard or from the command line (input stream `stdin`) and either prints it or lets you assign it to a variable (output stream `stdout`):

   ```
   input("What's your name? ") # prints and waits for input
   ```

3. `len` computes the length of its (string) argument and returns an integer:

   ```
   print(len("Birkenkrahe"))
   var = 'Dampfschiffahrtsgesellschaftskapitän'
   print(len(var))  # with the len() function
   print(var.__len__())  # with the str.__len__ method
   ```

4. `str` returns its value as a string:

```
print(str(1000) + " random numbers")
print(str('1000') + " random numbers")
```

5. `split` returns a `list` of words that can be split up among different variables:

```
name = "Marcus 2  Birkenkrahe"
print(name.split()) # default: split on whitespace, ignore ' '
first, last = name.split()  # split name in two parts
print(first,last)
print(first + last)
```

7 A few open questions

- What does the expression `str(int(age) + 1)` do?

  1. `age` is string `input`
  2. `int(age)` converts the string to a number - you cannot do that with any character like "a": `int("a")` throws an error. To convert characters to their Unicode standard, you need to use `ord`:

     ```
     print(int("25"))
     print(ord("a"))
     print(ord("A"))
     ```

  3. `int(age) + 1` adds 1 to whatever number `int(age)` evaluates to:

     ```
     age = "25"
     print(age)
     print(age + " years old")
     print(int(age))
     print(int(age)+1)
     ```

     ```
     25
     25 years old
     25
     26
     ```

  4. `str(int(age) + 1)` converts the result to a string:

     ```
     age = "25"
     print(age)
     ```

```
print(age + " years old")
print(int(age))
print(int(age)+1)
print(str(int(age)+1))
print(str(int(age)+1) + " years old")
```

```
25
25 years old
25
26
26
26 years old
```

- Here is an HTML animation to illustrate these steps (Sweigart, 2023)
- `split(self, / , sep=None, maxsplit=-1)` is a *string method* with two optional (defaulted) arguments - it returns list of words in the string using `sep` as the delimiter, at most `maxsplit` splits are done: elements (note the implicit arguments):

```
print('1,2,3'.split(','))  # default maxsplit = -1 means no limit
print('1,2,3'.split(',',0)) # don't split
print('1,2,3'.split(',',1)) # split once
print('1,2,3'.split(',',2)) # split twice
print('1,2,3'.split(',',3)) # split thrice - nothing more to do
```

- The dot-operator `.` is an *accessor*: it allows you to access anything that's stored inside an object, e.g. the *string* class `str`, or an instance of that class, a particular string.
- What happens when the string to be split does not have substrings?

```
a, b = 'Marcus'.split()
print(a,b)
```

- Why?

```
help(str.split)

Help on method_descriptor:
```

```
split(self, /, sep=None, maxsplit=-1)
    Return a list of the substrings in the string, using sep as the separator stri

      sep
        The separator used to split the string.

        When set to None (the default value), will split on any whitespace
        character (including \\n \\r \\t \\f and spaces) and will discard
        empty strings from the result.
      maxsplit
        Maximum number of splits (starting from the left).
        -1 (the default value) means no limit.

    Note, str.split() is mainly useful for data that has been intentionally
    delimited.  With natural text that includes punctuation, consider using
    the regular expression module.
```

- What does the `/` refer to in the `str.split` docstring:

  ```
  str.split(self, /, sep=None, maxsplit=-1)
  ```

  The `/` is a *parameter separator*: it denotes the end of positional-only parameters. After `self` (the string itself), the parameters `sep` and `maxsplit` can be explicitly assigned:

  ```
  print(str.split('Marcus Birkenkrahe'))
  print(str.split('Marcus_Birkenkrahe','_'))
  print(str.split('Marcus_Birkenkrahe',sep='_'))
  print('Marcus_Birkenkrahe'.split(sep='_'))
  print('Marcus_Birkenkrahe'.split('_'))

  ['Marcus', 'Birkenkrahe']
  ['Marcus', 'Birkenkrahe']
  ['Marcus', 'Birkenkrahe']
  ['Marcus', 'Birkenkrahe']
  ['Marcus', 'Birkenkrahe']
  ```

8 Summary

- Functions are like mini-programs in your program.

- The `print` function displays the value passed to it.
- The `input` function lets users type in a value.
- The `len` function takes a string value and returns an integer value of the string's length.
- The `int`, `str`, and `float` functions can be used to convert data.

9 Glossary

TERM/COMMAND	MEANING
`print`	printing function
`input`	takes input from stdin (e.g. keyboard, file)
`len`	returns length of argument
`str.split`	splits string into substrings
`str.strip`	removes leading and trailing whitespace
`int`, `float`, `str`	data type conversion functions

10 References

- pythontutor.com (2023). Visualize code execution.
- Sweigart, A. (2016). Invent your own computer games with Python. NoStarch. URL: inventwithpython.com.
- Sweigart, A. (2019). Automate the boring stuff with Python. NoStarch. URL: automatetheboringstuff.com.