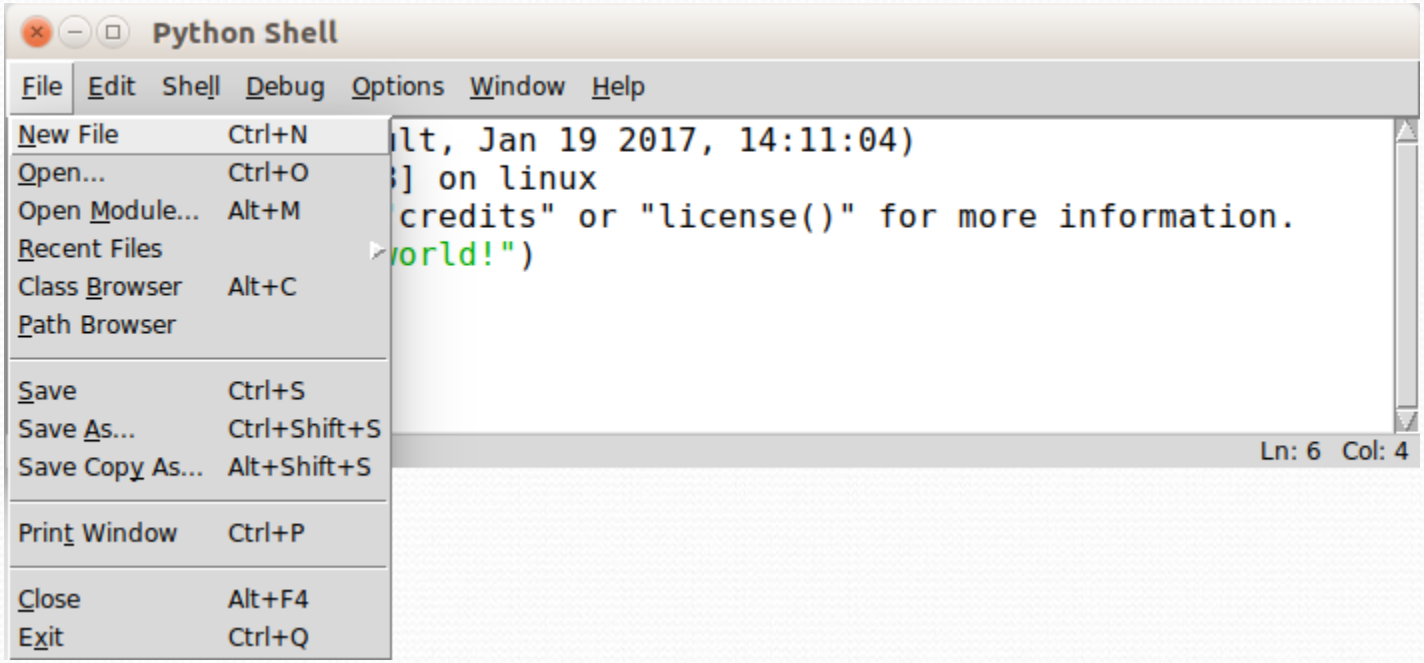
CS 115, Autumn 2021

Lecture 3: User Input

Thanks to Marty Stepp and Stuart Reges for parts of these slides

Creating a Python Program File



Creating a Python Program File

hello3.py

File Edit Format Run Options Window Help

```
print("Hello, world!")
print("This program produces")
print("Three lines of output")
```

Ln: 4 Col: 0

When Run -> Run Module is selected:

Python Shell

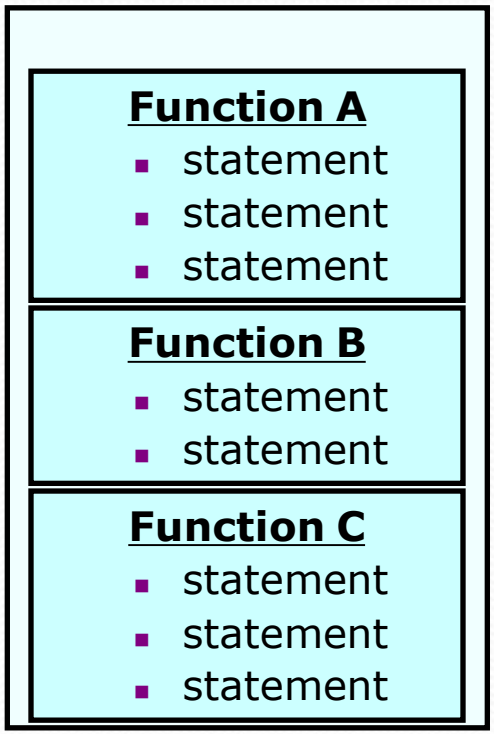
File Edit Shell Debug Options Window Help

```
>>>
 RESTART: /home/python/ch01-files/hello3.py
Hello, world!
This program produces
Three lines of output
>>>
```

Ln: 14 Col: 4

functions

- **function**: A named group of statements.
  - denotes the *structure* of a program
  - eliminates *redundancy* by code reuse
  - **procedural decomposition**: dividing a problem into functions
- Writing a function is like adding a new command to Python.



Declaring a function

Gives your function a name so it can be executed

- Syntax:

```
def name():
    statement
    statement
    …
    statement
```

- Example:

```
def print_warning():
    print("This product causes cancer")
    print("in lab rats and humans.")
```

Calling a function

Executes the function's code

- Syntax:

  name`()`

  - You can call the same function many times if you like.

- Example:

```
print_warning()          #separate multiple words with underscores
```

  - Output:

```
This product causes cancer
in lab rats and humans.
```

Comments

- **comment**: A note written in source code by the programmer to describe or clarify the code.
  - Comments are not executed when your program runs.
- Syntax:

```
# comment text
```

- Examples:

```
# This is a one-line comment.

# This is a very long
# multi-line comment.
```

Comments example

```
# Suzy Student,
# CSc 110, Fall 2019
# Displays lyrics

# first line
print("When I first got into magic")
print("it was an underground phenomenon")
print()

# second line
print("Now everybody's like")
print("pick a card, any card")
```

Interactive programs

interactive program: Reads input from the console.

- While the program runs, it asks the user to type input.
- The input typed by the user is stored in variables in the code.
- Can be tricky; users are unpredictable and misbehave.
- But interactive programs have more interesting behavior.

Variables

- **variable**: A piece of the computer's memory that is given a name and type, and can store a value.
  - Like preset stations on a car stereo, or cell phone speed dial:

  - Steps for using a variable:
    - *Declare/initialize* it - state its name and type and store a value into it
    - *Use* it - print it or use it as part of an expression

Declaration and assignment

- **variable declaration and assignment**:
  Sets aside memory for storing a value and stores a value into a variable.
  - Variables must be declared before they can be used.
  - The value can be an expression; the variable stores its result.

- Syntax:

  name = expression

  - **name = "Merlin"**

  - **last_name = "Cat"**

Using variables

- Once given a value, a variable can be used anywhere its value can be used:

```
cat = "Sir Purrcival"       # cat is "Sir Purrcival"

print(cat)  # outputs the text: Sir Purrcival
```

- You can assign a value more than once:

cat	"Sir Purrcival Cat"

```
cat = "Sir Purrcival"  # "Sir Purrcival" here


cat = "Sir Purrcival Cat" # now cat is "Sir Purrcival Cat"
```

input

- **`input`**: A function that can read input from the user.
- Using an `input` object to read console input:

  name `= input(`**prompt**`)`

  - Example:

    ```
    name = input("type your name: ")
    ```

    - The variable `name` will store the value the user typed in

Exercise: Welcome

- Write a program that asks the user their name and prints out a welcome message.

- Expected output:

```
What is your name? Merlin
Hello Merlin!
```

Exercise:

- Ask the user how old they are and then print out how many years it will be until they can retire. For this problem, we will assume everyone retires at 65.

- Expected output:

```
How old are you? 29
36 years until retirement!
```

Working with numbers

- Internally, computers store everything as 1s and 0s

```
104    → 01101000
'hi'   → 0110100001101001
'h'    → 01101000
```

- How are `h` and `104` differentiated?

- **type**: A category or set of data values.
  - Constrains the operations that can be performed on data
  - Many languages ask the programmer to specify types
  - Examples: integer, real number, string

Python's number types

Name	Description	Examples
`int`	integers	`42, -3, 0, 926394`
`float`	real numbers	`3.1, -0.25`
`complex`		

Expressions

- **expression**: A value or operation that computes a value.
  - Examples:

```
1 + 4 * 5
(7 + 2) * 6 / 3
42.0
```

  - The simplest expression is a *literal value*.
  - A complex expression can use operators and parentheses.

Arithmetic operators

- **operator**: Combines multiple values or expressions.
  - `+` addition
  - `-` subtraction (or negation)
  - `*` multiplication
  - `/` division
  - `//` integer division (a.k.a. leave off any remainder)
  - `%` modulus (a.k.a. remainder)
  - `**` exponent

- As a program runs, its expressions are *evaluated*.
  - `1 + 1` evaluates to `2`

`input` example – error!

```
def main():
    age = input("How old are you? ")

    years = 65 - age
    print(years, "years until retirement!")
```

- Console (user input underlined):

```
How old are you? 29
```

age 29

```
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    print(65 - age)
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

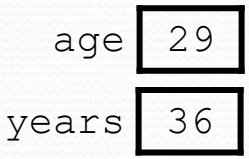
`input` example - fixed

```
def main():
    age = int(input("How old are you? "))

    years = 65 - age
    print(years, "years until retirement!")
```

- Console (user input underlined):

```
How old are you? 29
36 years until retirement!
```



Variables and expressions

- Once given a value, a variable can be used in expressions:

```
x = 3                   # x is 3
y = 5 * x - 1           # now y is 14
```

- You can assign a value more than once:

```
x = 3                   # 3 here

x = 4 + 7               # now x is 11
```

Assignment and algebra

- Assignment uses = , but it is not an algebraic equation.
  - = means, *"store the value at right in variable at left"*
    - The right side expression is evaluated first, and then its result is stored in the variable at left.
- What happens here?

```
x = 3
x = x + 2     # ???
```

Receipt question

Improve the receipt program using variables.

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    print("Subtotal:")
    print(38 + 40 + 30)

    print("Tax:")
    print((38 + 40 + 30) * .08)

    print("Tip:")
    print((38 + 40 + 30) * .15)

    print("Total:")
    print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 + 30) * .08)

main()
```

Printing a variable's value

- Use a comma to print a string and a variable's value on one line.
  - ```
    grade = (95.1 + 71.9 + 82.6) / 3.0
    print("Your grade was", grade)

    students = 11 + 17 + 4 + 19 + 14
    print("There are", students,
          "students in the course.")
    ```
  - Output:

    ```
    Your grade was 83.2
    There are 65 students in the course.
    ```
    ```

Receipt answer

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    subtotal = 38 + 40 + 30          # int
    tax = subtotal * .08             # float
    tip = subtotal * .15             # float
    total = subtotal + tax + tip     # float

    print("Subtotal:", subtotal)
    print("Tax:", tax)
    print("Tip:", tip)
    print("Total:", total)

main()
```