Statements

Thomas Schwarz SJ

Statements

- Computers execute instructions
  - An instruction (or sequence of instructions) corresponds roughly to a statement
- Python statements are put on separate lines
- You can (but should not) put more statements on a line
  - In which case you separate the statements with semi-colons :
    - ;

statements.py - /Users/thomasschwarz/Documents/My website

```
print('1'); print('2'); print(3);
```

Statements

- Types of statements we have already seen
  - print statements
  - import statements
  - assignment statements
  - conversion statements

Statements

- It is possible to combine several computation steps into a single statement
  - `print(2**2**2)`
    - Combines a calculation and a print operation
- `value = float(input('Enter a distance in yards'))`
  - The result of `input` is a string
  - The string is converted to a floating point number
  - And the resulting floating point number is assigned to the variable value

Statements

- Example (continued)
  - `print(value * METERS_PER_YARD)`
  - We multiply (the contents of) `value` with the value in the constant `METERS_PER_YARD`
- The result is stored in a temporary variable
  - Not seen nor accessible by the programmer
- The result is then handed over to the print function in order to be put out

Conversion Pattern

- Just as languages have idiomatic expressions, so do computer programs
  - Here is a program that converts Celsius to Fahrenheit
  - Conversion formula is

$$t_{\text{fahrenheit}} = \frac{9 * t_{\text{celsius}}}{5} + 32$$

Conversion Pattern

- When you begin to program, you are usually at a loss of what to do
  - A trick:
    - Execute the task with paper and pencil
    - Then explain to your Teddy Bear what you just did
    - Then emulate what you just explained as a program

Conversion Pattern

- Here is what we do:
  - We get a temperature in Celsius
  - This is a number
  - We apply the formula
  - We print out the result

Conversion Pattern

- One statement at a time:

```
my_input = input("Enter the temperature in Celsius: ")
temp_celsius = float(my_input)
temp_fahrenheit = 9*temp_celsius/5+32
print("The temperature is", temp fahrenheit, "Fahrenheit.")
```

Conversion Pattern

- We can combine statements if we want

```
celsius = float(input("Enter the temperature in Celsius"))
print("The temperature is", 9*celsius/5+32, "Fahrenheit.")
```

- We could go further, but this would trade in clarity for shortness

Conversion Pattern

- Another example:
  - Europe measures gasoline consumption using liters per kilometers
  - US uses miles per gallons
  - Let's write two conversion programs
    - MPG -> LPHK

Conversion Pattern

- Derivation of formula:
  - x miles per gallon, λ liters per gallon, μ kilometers per mile
  - $\frac{1}{x}$ gallons per mile
  - $\frac{\lambda}{\mu x}$ liters per kilometer
  - $\frac{\lambda}{0.01\mu x}$ liters per 100 kilometers

Conversion Pattern

- Now we can use the same pattern

```
LITERS_PER_GALLON = 3.78541178
KILOMETERS_PER_MILE = 1.609344

mpg = float(input('please enter fuel consumption in miles per gallon: '))
lphk = (100*LITERS_PER_GALLON)/(KILOMETERS_PER_MILE * mpg)
print('The fuel consumption is', lphk, 'liters per kilometer')
```

Conversion Pattern

- This gives us an output that is too verbose

```
= RESTART: /Users/thomasschwarz/Documents/My website/Classes/BPMumbai2022/Modules/Stat
ements/consumption.py
please enter fuel consumption in miles per gallon: 43
The fuel consumption is 5.470106583367089 liters per kilometer
```

- We can overcome this by rounding the result of the conversion
  - You can find this by searching the web
    - “Python3 rounding”

Conversion Pattern

- Better version:

```
LITERS_PER_GALLON = 3.78541178
KILOMETERS_PER_MILE = 1.609344

mpg = float(input('please enter fuel consumption in miles per gallon: '))
lphk = (100*LITERS_PER_GALLON)/(KILOMETERS_PER_MILE * mpg)
lphk = round(lphk, 2)
print('The fuel consumption is', lphk, 'liters per kilometer')
```

- I am reusing the variable lphk