Variables and Statements in Python

Colan Biemer

Variables

A **variable** is a name that points to a value (e.g. a number).

```
sales = 0
```

Printing

```
sales = 0
print(sales)
```

```
0
```

- `print` is a a function, which is called with the parentheses. Inside the parentheses are the **arguments**.

Printing with Strings

```
sales = 10000
print("Sales this year =", sales)
```

```
Sales this year = 10000
```

`"Sales this year ="` is a **string**

Why "String" ?

Every letter ('a', ...), number (0, ...), etc. is a character and string them together to create words, sentences, paragraphs, etc. This is how we ended up with the term *string* to describe text data.

Reassigning Variables

```
sales = 10000
print("sales =", sales)
sales = 100
print("sales =", sales)
sales = -100
print("sales =", sales)
```

```
sales = 10000
sales = 100
sales = -100
```

Comments

```
# Comments start with hashtags. They can be on their own lines like this
sales = 100 # they can also be after statements like this
```

Comments (Continued)

```
tax_rate = 0.0625 # set tax rate to 0.0625
tax_rate = 0.0625 # Default tax in MA
```

Which is a good comment and which is bad? Why?

Comments (Continued)

- Bad comment just repeats what the code already says
- Good comment explains *why*, something the code alone can't tell you
- Write comments for the next person reading your code, including future you

Variable Names

- Names must start with a letter, not a number
- Names can't contain punctuation, other than an underscore (i.e. `_`)
- Names cannot override python keywords (`if`, `else if`, etc. (see table 2-1 in textbook or google it))
- Upper versus lower case matters

Upper Case versus Lower Case

```
sales = 1
Sales = 2
print(sales, Sales)
```

```
1, 2
```

Good Variable Names

Everyone has their own rules, including the textbook. My advice is to err on the side of too descriptive. I also prefer snake case ("word_word") to camel case ("wordWord") because I find snake case to be more readable, but you can use either.

Variable Name Example

```
t = 0.0625
tax_rate = 0.0625
ma_tax_rate = 0.0625
massachussets_tax_rate = 0.0625
massachussets_tax_rate_percentage = 0.0625
```

Which do you think is worst? Which do you think is best?

Getting Input from the User

```
user_name = input("What is your name?") # prompt user for their name
print("Hi,", user_name)
```

```
What is your name? Colan
Hi, Colan
```

Good or bad comment?

Making it Simpler

Some people like to see how condensed they can make a program. Sometimes it makes the program better:

```
print("Hi,", input("What is your name?")) # Does this line need a comment?
```

Ultra Condensed Example

```
import math,os,time,itertools;all((lambda s,t,e,g:(lambda M:(os.system("cls" if os.name=="nt" else "clear"),print("\n".join("".join(M.get((x+r," ") for k in range(60)) for r in range(6,1760,60))),time.sleep(.03)))(s>v for D,o,v in sorted([(b,o+60*a,",.-:;=!*#@"[max(0,6)]) for J in range(0,628,7) for j in [J/100] for f,m in [(math.sin(j),math.cos(j))] for d in [m+2] for i in range(0,628,2) for i in [i/100] for c,l in [(math.sin(i),math.cos(i))] for D in [1/(c*d*m+f*t+6)] for n in [c*d*t-f*s] for x in [int(40+30*D*(l*d*g-c*m))] for y in [int(12+15*D*(l*d*m+c*g))] for N in [int(8*((f*s-c*m*l)*g-c*m*n-f*t-l*m*m))] if 0<=x<80 and 0<=y<23])))(math.sin(f*.07),math.cos(f*.07),math.sin(f*.03),math.cos(f*.03)) for F in itertools.count())
```

Sometimes it isn't comments that are needed, but better variable names and new lines for spacing.

The Moral

Concise code isn't necessarily better code.

Basic Types in Python

A variable in Python has a type.

```
sales = 132
print(type(sales))
```

```
<class 'int'>
```

Integer Type

An **integer** is a number with no decimal places (-100, 1, 100, 1000, etc.)

```
a = 1000  # valid
a = 1,000 # invalid
a = 1_000 # valid
```

Tax Rate Type

```
ma_tax_rate = 0.0625
print(type(ma_tax_rate))
```

```
<class 'float'>
```

Float Type

A `float` is a number with decimals (0.001, -1000.012, etc.). Note that there are minimum and maximum floats:

```
import sys # we'll cover this later
print(sys.float_info.max)
print(sys.float_info.min)
```

```
1.7976931348623157e+308
2.2250738585072014e-308
```

We shouldn't run into these in this class.

String Type

```
print(type("Hello, World!"))
```

```
<class 'str'>
```

Intricacies of Using a String

```
invalid_string = "Here is a quote, "To be or not be...""
```

Two ways around this:

1. Single quotes: `'Here is a quote, "To be or not to be..."'`
2. Backslashes: `"Here is a quote, \"To be or not to be...\""`

Intricacies of Using a String (Continued)

You can represent special characters like new lines and tabs in strings.

```
print("newline\nnewline\n")
print("\t- a")
print("\t- b")
```

```
newline
newline

  - a
  - b
```

Back to Input

```
type(input("What year was Napoleon born?"))
```

Any guesses for what the type will be?

Back to Input

```
type(input("What year was Napoleon born?"))
```

Any guesses for what the type will be?

String. This isn't great if we want to use it as a number.

Converting a String to an Int or a Float

```
example_string = input("What year was Napoleon born?")
example_int    = int(example_string)   # convert to integer
example_float  = float(example_string) # convert to float
```

Abbreviations line "integer" to "int" are everywhere in programming.

Additional Operations with Numbers (int and float)

- `+` - add two numbers (`a = 1 + 3`)
- `-` - subtract two numbers (`a = 1 - 3`)
- `*` - multiply two numbers (`a = 1 * 3`)
- `/` - divide two numbers (`a = 1 / 3`)
- `**` - raise to the power of `a = 1 ** 3` (which is 1 to the power of 3)
- There is also `//` and `%`, but we'll cover them later

First IPO Program!

Review: What does IPO stand for?

Calculate the Sum of Two User Input Numbers

```
# help me program this...
```

- What is the input?
- What is the processing?
- What is the output?

Calculate the Sum of Two User Input Numbers

```
a = int(input('Give me a number')) # input
b = int(input('Give me another'))  # input

total = a + b                      # processing

print(total)                       # output
```

Was `int` the right choice?

Another One: Let's Calculate the Percent Difference of Two Prices

```
# help me, and let's separate into IPO steps again
```

Percent difference equation: $|a - b| / ((a + b)/2)$

For this we need the python function `abs`, which returns the absolute value of a number (e.g. `abs(-3) -> 3` and `abs(3) -> 3`).

Another One: Let's Calculate the Percent Difference of Two Prices

```
price1 = float(input("First price"))                # input
price2 = float(input("Second price"))               # input

numerator = abs(price1 - price2)                    # processing
denominator = (price1 + price2) / 2                 # processing
percent_difference = numerator / denominator        # processing

print("Percent difference =", percent_difference) # Output
```

Another One: Show the Minimum and Maximum of Two Prices

```
# min(a, b) returns minimum of a and b
# max(a, b) returns maximum of a and b
```

Another One: Show the Minimum and Maximum of Two Prices

```
price1 = float(input("First price"))  # input
price2 = float(input("Second price")) # input

minimum = min(price1, price2)         # processing
maximum = max(price1, price2)         # processing

print("Minimum =" minimum)            # output
print("Maximum =" maximum)            # output
```

Last One: Simple Algebra Problem

As input, I want the user to tell me how many items they bought and what the total price was. I will output the average price per item, and I want it rounded to two decimal places.

We need this python function: `round([number], [decimal places])`

Last One: Simple Algebra Problem

```
number_of_items = float(input("How many items did you buy?"))
cost = float(input("What was the total cost?"))

cost = cost / number_of_items # Reassigning allowed in Python
rounded_average = round(cost, 2)

print("Average cost per item:", rounded_average)
```

String Operations: Concatenation

String concatenation puts two strings together:

```
a = "one"
b = "two"
c = a + b # concatenation
print(c)
```

```
onetwo
```

String Operations: Concatenation

```
a = "one"
b = "two"
a += b # shorthand (we can also use this shorthand for numbers)
print(a)
```

```
onetwo
```

String Operations: Methods

```
a = "one"
b = a.capitalize()
print(b)
```

```
One
```

Method

A `method` is a function that belongs to an object, like `capitalize` for strings in Python. You can access them with `.`.

Another example is to make a string all upper case:

```
a = "one"
print(a.upper())
```

```
ONE
```

Summary

- A **variable** is a name pointing to a value, and you can reassign it
- `print` outputs value
- `input` reads text from the user and always returns a string
- Comments start with `#`
- Three types so far: `int` (no decimals), `float` (decimals), `str` (text)
- Numbers use `+`, `-`, `*`, `/`, and `**`
- String concatenation puts two or more strings togher
- A **method** is a function belonging to an object, called with `.` (e.g. `a.upper()`)
- Useful functions so far: `abs`, `min`, `max`, `round`, `type`, `print`

Next Time

Every program so far runs top to bottom. Next we'll write programs that make decisions about whether to do one thing or another.

Booleans and Decision Making