Introduction to Programming

Unit 2: Variables, values (and user IO)

Programming is about managing and working with "data".

"Data" is any information you're dealing with.

Variables

Variables are where you "store" data in your programs.

Values

Values are the data itself.

Mad Lib

Let's learn about variables and values by playing Mad Lib.

Can someone give me:

- Two adjectives
- One game
- Three nouns
- One plural noun
- Two verbs ending in "ing"

Let's play Mad Lib

A vacation is when you take a trip to some (***adjective***) place with your (***adjective***) family. Usually you go to some place that is near a/an (***noun***) or up on a/an (***noun***). A good vacation place is one where you can ride (***plural noun***) or play (***game***) or go hunting for (***noun***). I like to spend my time (***verb ending in "ing"***) or (***verb ending in "ing"***).

Before we write the code, let's write the pseudocode.

What is pseudocode?

Pseudocode is a made up programming language that helps humans come up with the structure for their code.

Pseudocode

Pseudocode looks like code, but it's intended for humans and not computers.

Sample pseudocode program

```
if it's 6pm on a Monday or a Thursday
then go to the Python class
```

Sample pseudocode program

```
as long as there is another episode available
then keep watching TV
```

Pseudocode can look however we want since we make it up, but as we learn more Python our pseudocode will start to look a more like Python.

Later our pseudocode will start to look more like computer code

```
if it's 6pm on a Monday or a Thursday
then go to the Python class
```

will turn into this:

```
if time equals 6pm and (day equals Monday or Thursday) then
    go to the Python class
```

which will turn into this:

```
if time == 6 and (day == Monday || day == Thursday):
    go_to("Python class")
```

Pseudocode code will help us design and write our code.

But remember that pseudocode is not code itself.

Let's practice.

Mad Lib steps for humans

- Ask your friend for:
  - Two adjectives
  - One game
  - Three nouns
  - One plural noun
  - Two verbs ending in "ing"
- Then we add the words into the sentence
- Finally we say the sentence

Mad Lib steps for computers

- We ask the user to enter the needed words:
  - adjective 1, adjective 2
  - game 1
  - noun 1, noun 2, noun 3
  - plural noun 1
  - verb ending in "ing" 1, verb ending in "ing" 2
- Then we add the words into the sentence
- Show the sentence to the user

Let's write the pseudocode code for our Mad Lib program.

Mad Lib pseudocode

```
ask the user for adjective 1.
ask the user for adjective 2.
ask the user for game 1.
ask the user for noun 1.
ask the user for noun 2.
ask the user for noun 3.
ask the user for plural noun 1.
ask the user for verb ending in "ing" 1.
ask the user for verb ending in "ing" 2.

... continues on next slide
```

Mad Lib pseudocode (cont.)

```
... continuation from previous slide

combine "A vacation is when you take a trip to some "
  with adjective 1
  and with " place with your "
  and with adjective 2
  and with " family. Usually you go to some place that is near a/an "
  and with noun 1
  and with " or up on a/an "
  and with noun 2
  and with ". A good vacation place is one where you can ride "
  and with plural noun 1
  and with " or play "
  and with game 1
  and with " or go hunting for "
  and with noun 3
  and with " I like to spend my time "
  and with verb ending in "ing" 1
  and with " or "
  and with verb ending in "ing" 2
  and with "."
  into the final mad lib.

show the final mad lib to the user.
```

There's no magic in computer programming.

If you didn't code your computer to perform a task, it won't do it.

We must specify each step we want the computer to perform in our code.

The same applies to our pseudocode.

Now let's use words like "input", "output", and "store" in our pseudocode.

Mad Lib pseudocode

```
ask for an adjective and store it in adjective 1.
ask for an adjective and store it in adjective 2.
ask for a game and store it in game 1.
ask for a noun and store it in noun 1.
ask for a noun and store it in noun 2.
ask for a noun and store it in noun 3.
ask for a plural noun and store it in plural noun 1.
ask for a verb ending in "ing" and store it in verb ending in "ing" 1.
ask for a verb ending in "ing" and store it in verb ending in "ing" 2.

... continues on next slide
```

Mad Lib pseudocode (cont.)

```
... continuation from previous slide

combine "A vacation is when you take a trip to some "
  with adjective 1
  and with " place with your "
  and with adjective 2
  and with " family. Usually you go to some place that is near a/an "
  and with noun 1
  and with " or up on a/an "
  and with noun 2
  and with ". A good vacation place is one where you can ride "
  and with plural noun 1
  and with " or play "
  and with game 1
  and with " or go hunting for "
  and with noun 3
  and with " I like to spend my time "
  and with verb ending in "ing" 1
  and with " or "
  and with verb ending in "ing" 2
  and with "."
  and store it in the result.

output the result to the screen.
```

Let's turn our pseudocode code into Python

Other types of values

Python has more than just string values.

Other types of values

There are booleans, numbers, lists, dictionaries, tuples, and many others.

Why are the multiple types of values?

Different types of values have different capabilities.

We'll talk about strings, booleans, and numbers today

- Strings represent text which can be printed to the screen.
- Integers and floats represent numbers which you can do arithmetic on.
- Booleans represent the answer to yes/no questions.

Booleans

There are two boolean *values*: `True` and `False`.

Booleans

We'll talk more about booleans when we learn about `if` statements.

Numbers

There are two kinds of numbers in Python: integers and floating point numbers.

Integers

Integers represent whole/natural numbers. For example, `1`, `7`, `42`, `0`, and `-453`

Floats

Floats represent real numbers. For example, `3.14`, `54.654543`, `1.0`, and `-2394.8`

Different capabilities

- Strings represent text which can be printed to the screen.
- Integers and floats represent numbers which you can do arithmetic on.
- Booleans represent the answer to yes/no questions.

Syntax

- Strings are surrounded by double-quotes.
- Integers are numbers.
- Integers are numbers with a decimal.
- Booleans are just `True` and `False`.

Operators

Arithmetic operators

Operator	Name	Example	Result
`**`	Exponentiation	`2 ** 3`	`8`
`*`	Multiplication	`43 * 7`	`301`
`/`	Division	`45 / 7`	`6.428571428571429` *
`//`	Floor division	`45 // 7`	`6`
`%`	Modulus	`45 % 7`	`3`
`+`	Addition	`54 + 543`	`597`
`-`	Subtraction	`654 - 32`	`622`

String operators

Operator	Name	Example	Result
`+`	Concatenation	`"abc" + "xyz"`	`"abcxyz"`
`*`	Multiplication	`"ab" * 4`	`"abababab"`
`%`	Format	`"a: %s, b: %s" % ("1", "2")`	`"a: 1, b: 2"`

Comparison operators

Operator	Name	Example	Result
`==`	Equal	`7 == 7`	`True`
`!=`	Not equal	`7 != 7`	`False`
`>`	Greater than	`43 > 33`	`True`
`>=`	Greater than or equal to	`43 >= 43`	`True`
`<`	Less than	`32 < 546`	`True`
`<=`	Less than or equal to	`32 <= 31`	`False`

Boolean operators

Operator	Example	Result
`and`	`True and True`	`True`
	`True and False`	`False`
	`False and False`	`False`
`or`	`True or True`	`True`
	`True or False`	`True`
	`False or False`	`False`
`not`	`not True`	`False`
	`not False`	`True`

Complex boolean expressions

- `(2 < 43 or 453 >= 23 or 3 > 1) and (43 > 65)`
- `34 <= (3 + 34) and not (43 < 2 or 3 != 3)`
- `True and not False or False`
- `True and not True or True`
- `True and not (True or True)`

Casting

Casting

Casting is the act of converting a value of one type to another type.

Casting

If you have a string but you need to use it as an integer, you have to cast it.

Casting

- `str` casts a value to a string
- `int` casts a value to an integer
- `float` casts a value to a float

Problem set