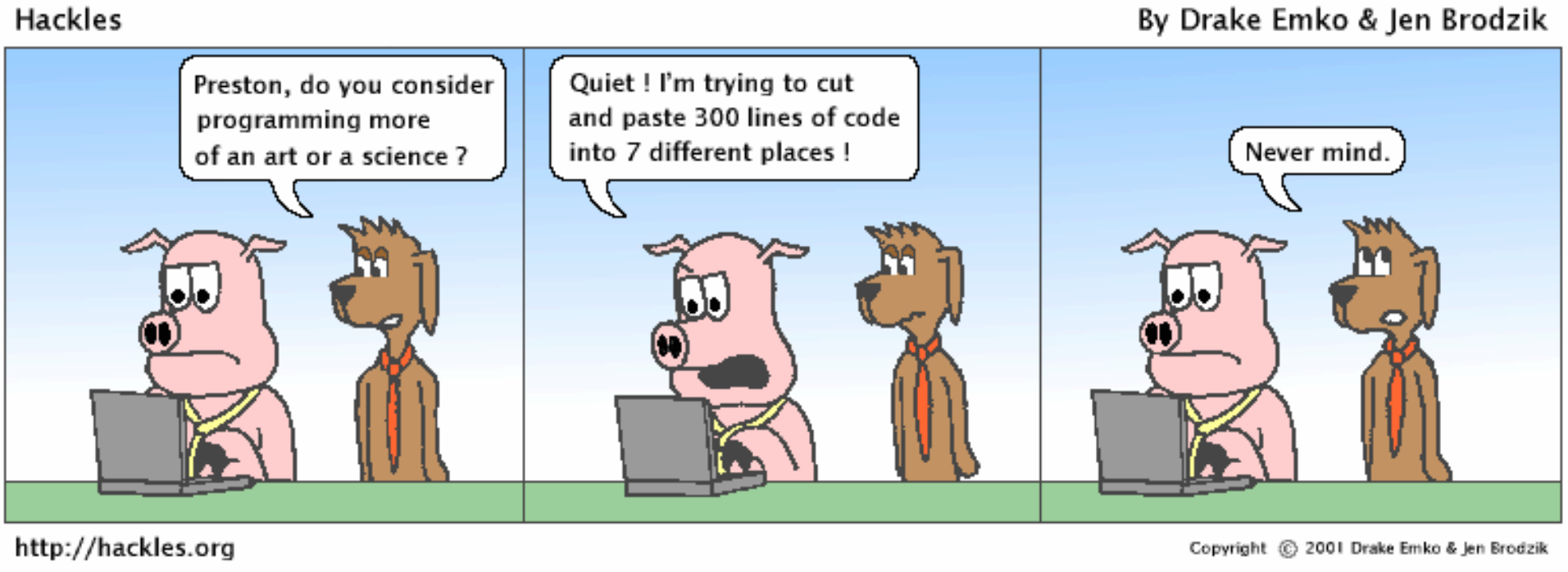
CS 115, Autumn 2021

Lecture 4: User Input; variables; expressions



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

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`

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	11

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

`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'
```

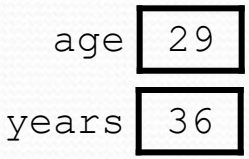
`input` example - fixed

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

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

- Console (user input underlined):

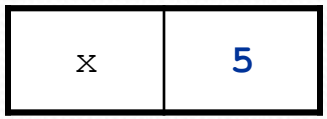
```
How old are you? 29
36 years until retirement!
```



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    # ???
```



Exercise: Receipt

- Write a program that prompts a user for three meal prices and outputs:
  - the subtotal
  - amount of tax that would be charged for them (rate of 8%)
  - the tip owed (rate of 15%)
  -
  - the total owed (subtotal + tax + tip)

Printing a variable's value

- Use a comma to print a string and a number 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.
    ```
    ```

Printing a variable's value

- You can also print a number variable and string with + but you have to tell Python to treat the number variable as a string with `str`
  - ```
    grade = (95.1 + 71.9 + 82.6) / 3.0
    print("Your grade was " + str(grade))

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

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