Lecture 1: Expressions and Variables

CS 51P

August 30, 2023

Types

A **type** is a set of values and plan for representing/interpreting those values in binary

int

- Values: 0, 1, -10, 34022, …
- Operations: +, -, /, *

  ** (exponent),

  % (remainder)

  // (truncated division)

string

- Values: "Hi!", "", "2.0",…
- Operations: + (concatenation)

  * (duplication),

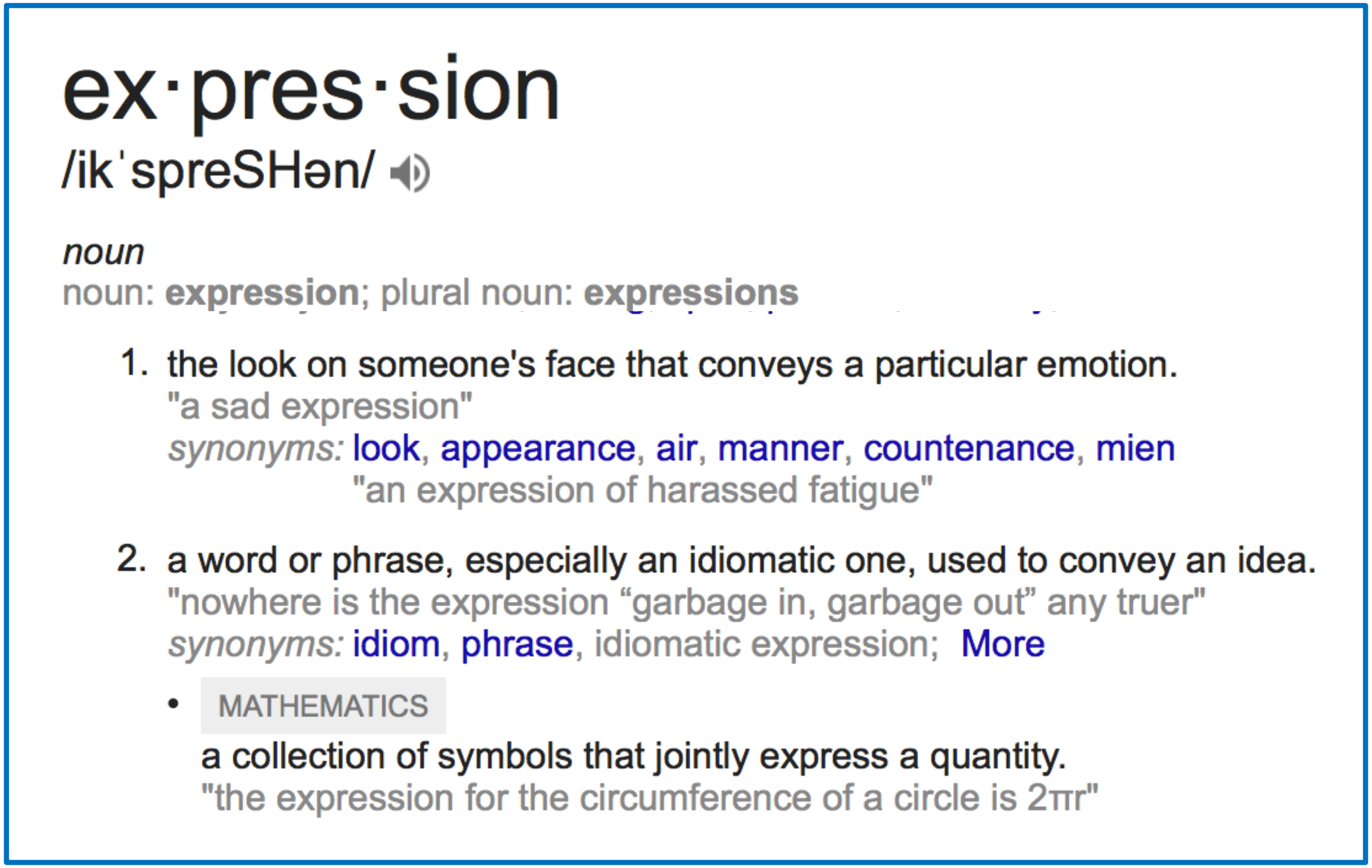
All values have types

Common types: int, float, str, bool

Terminology

- Value
- Type
- Operator
- Expression

Expressions



ex·pres·sion

/ikˈspreSHən/

noun

noun: **expression**; plural noun: **expressions**

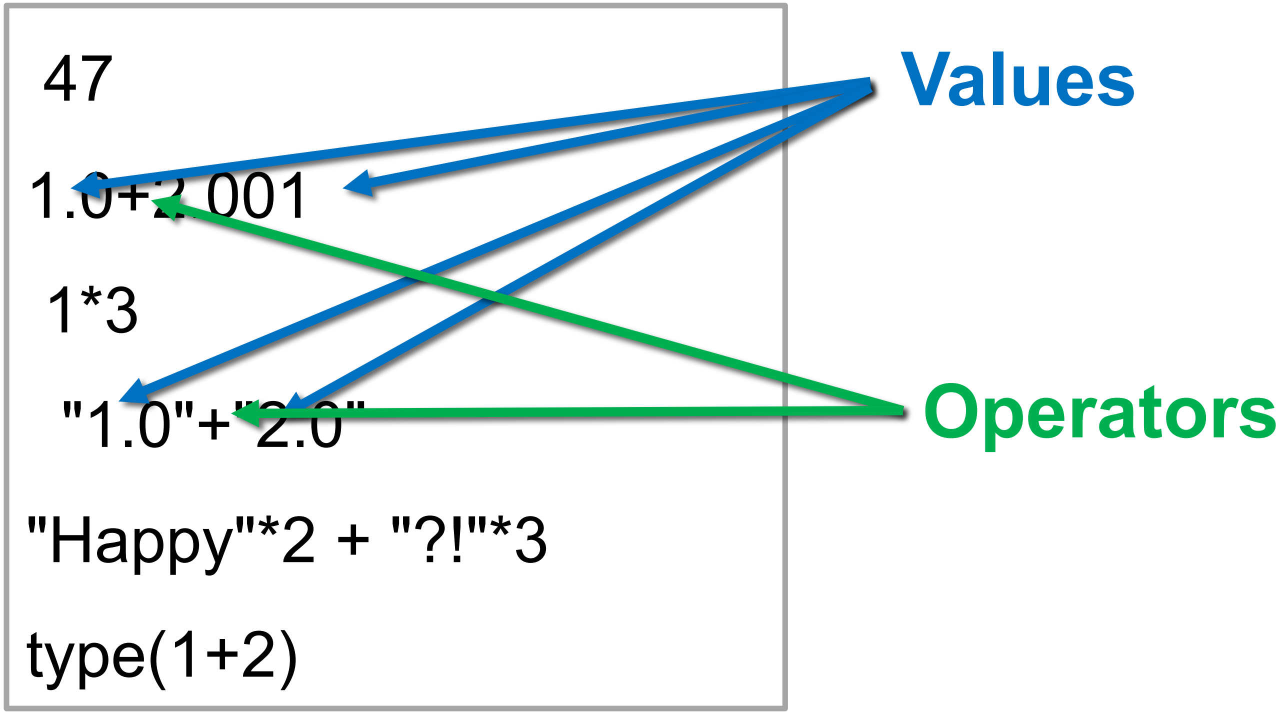
1. the look on someone's face that conveys a particular emotion.
   "a sad expression"
   synonyms: look, appearance, air, manner, countenance, mien
   "an expression of harassed fatigue"

2. a word or phrase, especially an idiomatic one, used to convey an idea.
   "nowhere is the expression “garbage in, garbage out” any truer"
   synonyms: idiom, phrase, idiomatic expression; More

   - MATHEMATICS
     a collection of symbols that jointly express a quantity.
     "the expression for the circumference of a circle is 2πr"

Expressions

- Expressions represent a value
- Python evaluates expressions (similar to a calculator)



```
47
1.0+2.001
1*3
"1.0"+"2.0"
"Happy"*2 + "?!"*3
type(1+2)
```

Values

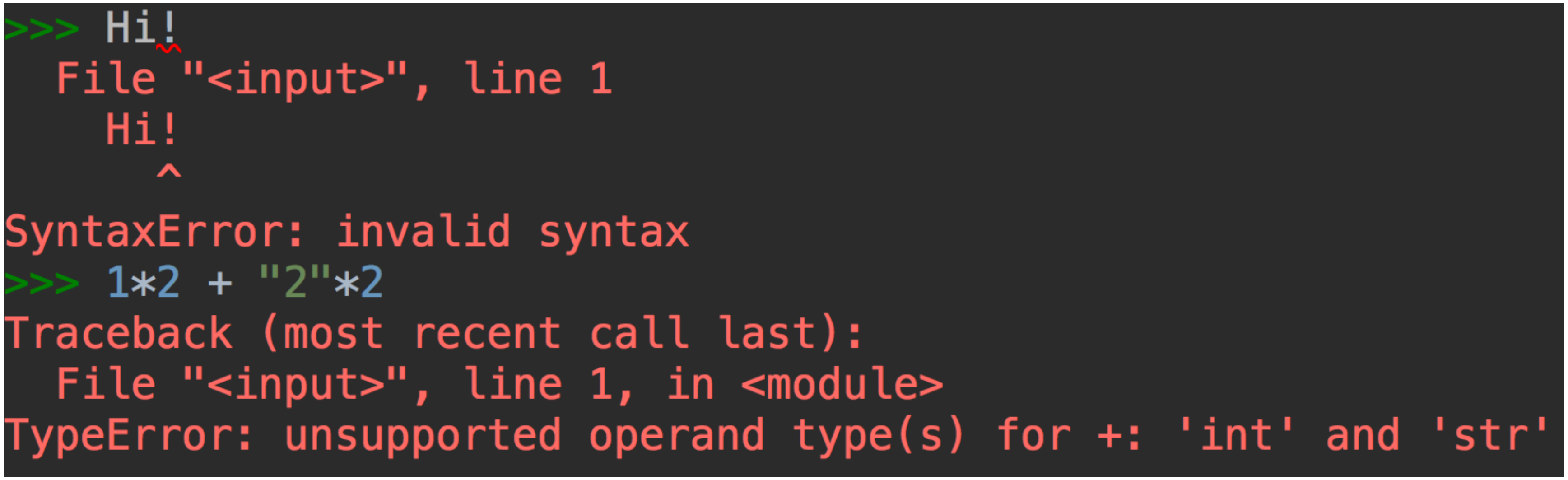
Operators

Exercise 1: Expressions

```
1/2
13
4 + 3 * 2
( "A"*2 + "?"*3 ) * 2
14 % 5
5 ** 2
Hi!
1*2 + "2"*2
```

Errors

- Three types of errors:
  - ValueError: unsupported operand value(s)
  - TypeError: unsupported operand type(s) for +: 'int' and 'str'
  - SyntaxError: invalid syntax



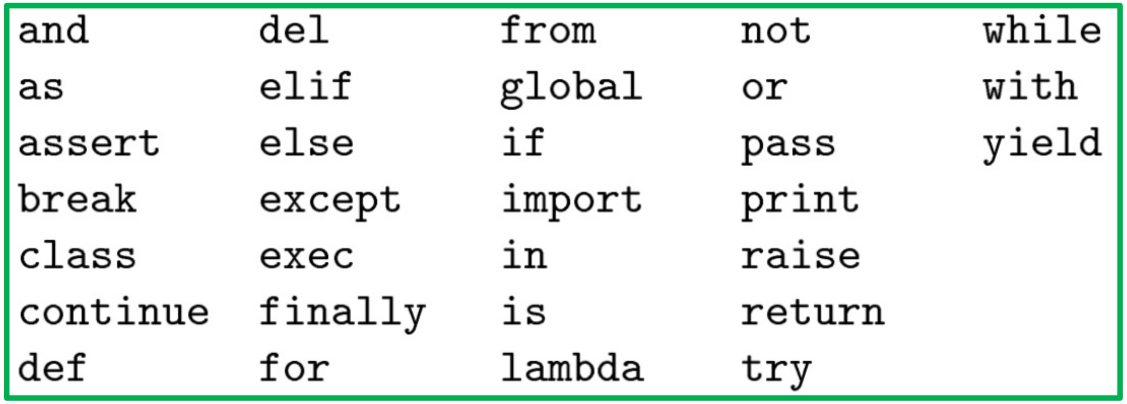
```
>>> Hi!
  File "<input>", line 1
    Hi!
      ^
SyntaxError: invalid syntax
>>> 1*2 + "2"*2
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Exercise 2: Expressions and Errors

```
3 * "5"

str(3) * int("5")

1 / 2 * "Hello"

"1.0" + 2.0

int("2") * "2"

1 ** 2.5

str("2") * 4
```

Variables

- A variable is a name that refers to a value
  - names should be meaningful
  - by convention words separated by an underscore
  - names cannot be a keyword (e.g. *print*), cannot include spaces, must begin with a letter

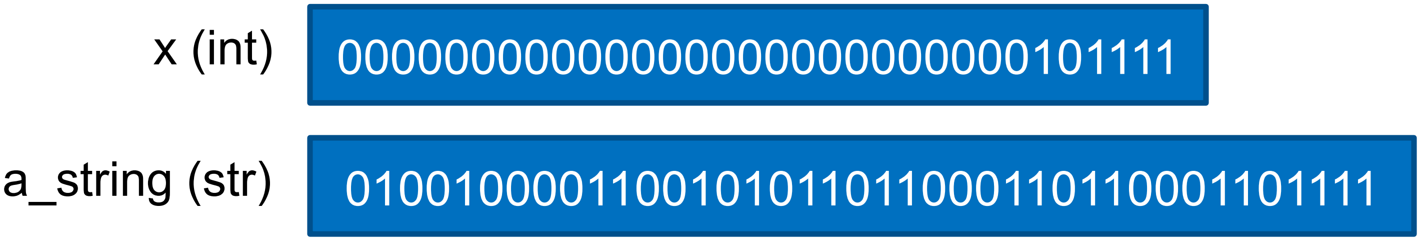


and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	
class	exec	in	raise	
continue	finally	is	return	
def	for	lambda	try	

Assigning variables

- Can assign a value to a variable

```
x = 47

a_string = "Hello"
```



x (int) 00000000000000000000000000101111

a_string (str) 0100100001100101011011000110110001101111

Assigning variables

- Can assign a value to a variable
- Right hand side can be any expression (anything that is, or that evaluates to, a value)

```
x = 13

a_string = 1*str(2) + "2"*2

x_type = type(1+2.001)
```

Variables and Expressions

- a variable evaluates to the value stored in that variable
- variables can be used in expressions

```
my_num = 13

new_num = my_num + 34
```

Example: Word Problem

If you run a 10 kilometer race in 43 minutes 30 seconds, what is your average time per mile? (Hint: there are 1.61 kilometers in a mile).

Example: Writing a Program

If you run a 10 kilometer race in 43 minutes 30 seconds, what is your average time per mile? (Hint: there are 1.61 kilometers in a mile).

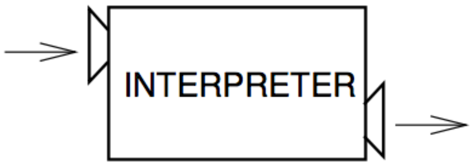
Exercise 3: Writing a Program

If you run a 10 kilometer race in 43 minutes 30 seconds, what is your average speed in miles per hour? (Hint: there are 1.61 kilometers in a mile).

Demo: Running a Program

High-level languages

```
print("Hello World!\n")
```



```
pushq   %rbp
movq    %rsp, %rbp
subq  $32, %rsp
leaq  L_.str(%rip), %rax
movl  $0, -4(%rbp)
movl  %edi, -8(%rbp)
movq  %rsi, -16(%rbp)
movq  %rax, %rdi
movb  $0, %al
callq _printf
xorl  %ecx, %ecx
movl  %eax, -20(%rbp)
movl  %ecx, %eax
addq  $32, %rsp
popq  %rbp
retq
```

Figure 1.1: An interpreter processes the program a little at a time, alternately reading lines and performing computations.

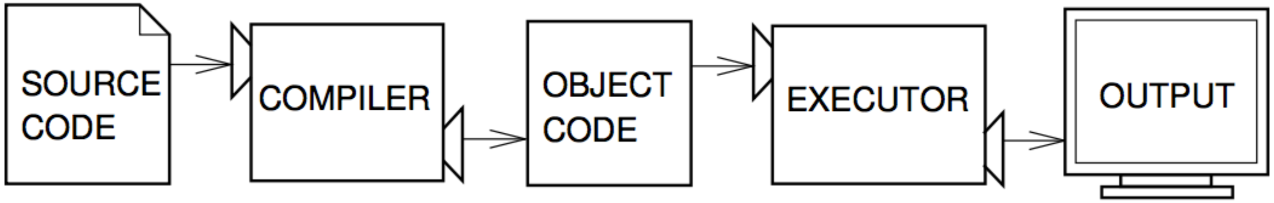


Figure 1.2: A compiler translates source code into object code, which is run by a hardware executor.

Input an Output

- print() is a function that allows you to display text on the screen

- input() is a function that allows you to get a value from your user

Example

How would you change this program so that it asks the user for their time in minutes and seconds and then prints out their average time per mile for the 10K race?

Exercise 4: Interactive Programs

Write a program that asks the user for word and a number, then prints the word followed by an exclamation point and a space that many times. For example:

```
Enter a word:
    happy
Enter a positive integer:
    3

happy! happy! happy!
```