Programming

- computer program: sequence of instructions
- describes how to perform a task

Creating Programs

- to run, a program has to be in an executable format
- machine code
- very difficult to write by humans

- we write programs in a programming language
- source code
- we use programs to convert source code to machine code

Programming Languages

- lots of languages with different characteristics
- C, C++, Java, C#, Swift, Go
- Python, Ruby, Perl, PHP, Tcl
- JavaScript, TypeScript, Rust
- Lisp, Haskell, Ocaml, F#, Scala, Clojure, Erlang
- R, SQL
- …

Python

- created by Guido van Rossum
- in early 1990s

- major changes in 2008: Python 3

Monty Python

- named after a British comedy group from the 1970s

Popularity

- web and enterprise applications (IEEE):

  The Top Programming Languages 2017

- for teaching programming (ACM):

  Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities

Popularity - 2

- developing projects (GitHub):

  GitHub Octoverse 2017

- questions and discussions (StackOverflow):

  The Incredible Growth of Python

Who's Using It?

- Youtube, Google
- Dropbox
- Instagram
- Pinterest
- Reddit
- NASA
- IL&M
- ...

Application Areas

- web applications
- data science
- scientific computation
- system administration
- …

Source Files

- extension for source files: `.py`
- running a source file:

  ```
  python SOURCE_FILE.py
  ```

Interactive Mode

- REPL: Read Eval Print Loop
- ask a question, get an answer

- shows prompt, waits for input
- evaluates input
- prints result
- shows prompt, waits for input
- …

Python REPL

- run:

```
python
```

- and you see the prompt:

```
Python 3.6.2 ...
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" ...
>>>
```

Jupyter

- interactive environment for many languages
- Python, R, Julia, JavaScript, Haskell, C++, …

- on the console: `jupyter-console`
- in the browser: `jupyter-notebook`

Development Environments

- any text editor will do

- PyCharm
- Eclipse PyDev
- Spyder
- IDLE
- …

Expressions

- an expression describes a computation
- evaluating it results in a value

- examples:

$$35 + 7$$

$$2^5$$

$$14!$$

Expressions in Jupyter

- type expression, get result

```
In [1]: 35 + 7
Out[1]: 42
```

```
In [2]: 13 * 3
Out[2]: 39
```

```
In [3]: 6 + 7 * 4
Out[3]: 34
```

Expression Components

- literals: values written directly
- operators: addition, multiplication, …

- only a literal:

```
42
```

- literals connected with operators:

```
13 * 3
```

Syntax Errors

- source code has to follow language rules

```
In [4]: 6 +* 7
  File "<ipython-input-4-0aa372dde964>", line 1
    6 +* 7
       ^
SyntaxError: invalid syntax
```

Assignment

- assignment: associate a value with a name
- variable: named value

- variables can be used in expressions
- value substitutes variable

Assignment in Python

- syntax:

```
name = expression
```

1. evaluate expression
2. associate resulting value with name

Statements

- assignment is a statement
- it doesn’t return a result
- not a question

- a source file consists of statements
- and comments: from # until end of line

Assignment Examples

```
In [5]: midterm = 85

In [6]: final = 78

In [7]: total = midterm * 0.45 + final * 0.55

In [8]: total
Out[8]: 81.15
```

Assignment and Equality

- assignment is not equality!

```
In [9]: x = 41

In [10]: x = x + 1

In [11]: x
Out[11]: 42
```

Name Rules

- start with letters
- can contain letters, digits and underscore
- no punctuation or white-space
- case sensitive: $A \neq a$

Missing Variable

- what happens if:

```
total = midterm * 0.3 + assignment * 0.3 + final * 0.4
```

```
In [12]: total = midterm * 0.3 + assignment * 0.3 + final * 0.4
---------------------------------------------------------------
NameError                                 Traceback (most recent
<ipython-input-12-a3053fe74ae0> in <module>()
----> 1 total = midterm * 0.3 + assignment * 0.3 + final * 0.4

NameError: name 'assignment' is not defined
```

Types

- every value has a type
- how data is to be interpreted

- numeric: integer (int), real (float)
- literal: if no decimal point then int, else float

- text: string of characters (str)
- literal: surrounded by double or single quotes

Type Examples

literal	type
`42`	`int`
`3.14159`	`float`
`'Hello'`	`str`
`"42"`	`str`

String Delimiters

- a string starting with " is only ended by "
- a string starting with ' is only ended by '

```
"I said 'hello'."

'I said "hello".'
```

Multiline Strings

- putting a newline into a string: `\n`

```
'Mountain sheep are sweeter,\nvalley sheep are fatter.'
```

- multi-line strings: three quotes (double or single)

```
"""Mountain sheep are sweeter,
valley sheep are fatter."""
```

Arithmetic Operators

- addition: `x + y`
- subtraction: `x - y`
- multiplication: `x * y`
- division: `x / y`

- integer division: `x // y`
- division remainder (mod): `x % y`

- exponentiation: `x ** y`

Arithmetic Operator Examples

operator	expression	result	type
`+`	`6 + 7`	`13`	`int`
`*`	`6 * 7`	`42`	`int`
`/`	`15 / 6`	`2.5`	`float`
`//`	`15 // 6`	`2`	`int`
`%`	`15 % 6`	`3`	`int`
`**`	`4 ** 3`	`64`	`int`

String Concatenation

- addition on strings → concatenation

```
In [13]: 'Hello,' + 'world!'
Out[13]: 'Hello,world!'

In [14]: name = 'Eric'

In [15]: greeting = 'Hello,' + ' ' + name + '!'

In [16]: greeting
Out[16]: 'Hello, Eric!'
```

Type Errors

- operand types must match operation

```
In [17]: birth_year = 1991

In [18]: age = 2017 - birth_year

In [19]: 'Python is ' + age + ' years old.'
----------------------------------------------------------------
TypeError                                 Traceback (most recent
<ipython-input-19-10e3e8904f0b> in <module>()
----> 1 'Python is ' + age + ' years old.'

TypeError: must be str, not int
```

Functions

- take input: parameters (also called “arguments”)
- produce output: return values

Function Examples

- abs: absolute value - 1 parameter
- min: minimum - 2 parameters
- max: maximum - 2 parameters
- round: 2 parameters (value and precision)
- len: length - 1 parameter

Function Usage Examples

```
In [20]: abs(-3)
Out[20]: 3

In [21]: min(midterm, final)
Out[21]: 78

In [22]: max(midterm, final)
Out[22]: 85

In [23]: round(total, 1)
Out[23]: 81.2

In [24]: len(greeting)
Out[24]: 12
```

Functions as Operands

- functions can be operands in expressions
- replace function expression with its return value

```
In [25]: abs(-3) + 3
Out[25]: 6

In [26]: min(3, -3) + max(3, -3)
Out[26]: 0
```

Parameter Expressions

- function parameters are expressions

```
In [27]: min(3 * 9, 4 * 8)
Out[27]: 27

In [28]: min(abs(-10), abs(3))
Out[28]: 3
```

Type Conversions

- functions to convert values between types

```
In [29]: str(42)
Out[29]: '42'

In [30]: int('42')
Out[30]: 42

In [31]: int(42)
Out[31]: 42
```

Type Conversion Errors

- what's the result of `int('Eric')`?

- a syntax error?
- a type error?

```
In [32]: int('Eric')
-----------------------------------------------------------------
ValueError                                Traceback (most recent
<ipython-input-32-f84d53442c9b> in <module>()
----> 1 int('Eric')

ValueError: invalid literal for int() with base 10: 'Eric'
```

Input and Output

- interaction with the user
- output: print a string to the screen

```
print(message)
```

- input: read a string from the keyboard

```
variable = input(prompt)
```

Output Example

- a program to print a message

```
print('Hello, world!')
```

Output Example - 2

- a program to get an input and produce an output

```python
name = input('What is your name? ')
message = 'Hello, ' + name + '!'
print(message)
```

Simple Flow

- get inputs from user
- process inputs and produce results
- output results

Simple Flow Example

```python
response = input('In which year were you born? ')
birth_year = int(response)
age = 2017 - birth_year
message = 'You are ' + str(age) + ' years old.'
print(message)
```

Libraries

- library: collection of code
- functions, constants, …
- grouped into packages

- import into your code

Importing Libraries

- syntax 1:

```
from LIBRARY import NAME
```

- syntax 2:

```
import LIBRARY

# use names as: LIBRARY.NAME
```

Import Example - 1

- importing a constant

```
In [33]: from math import pi

In [34]: pi
Out[34]: 3.141592653589793

In [35]: r = 4.2

In [36]: area = pi * r ** 2

In [37]: area
Out[37]: 55.41769440932395
```

Import Example - 2

- importing a function

```
In [38]: from math import pi, sqrt

In [39]: sqrt(area / pi)
Out[39]: 4.2
```

Math Library Example

```python
# Given the radius, calculate the area of a circle.

from math import pi

response = input("What's the radius of the circle? ")
radius = float(response)
area = pi * radius ** 2
message = 'The area is: ' + str(area)
print(message)
```

Math Library Example - 2

```python
# Given the area, calculate the radius of a circle.

import math

response = input("What's the area of the circle?" )
area = float(response)
radius = math.sqrt(area / math.pi)
message = 'The radius is: ' + str(radius)
print(message)
```