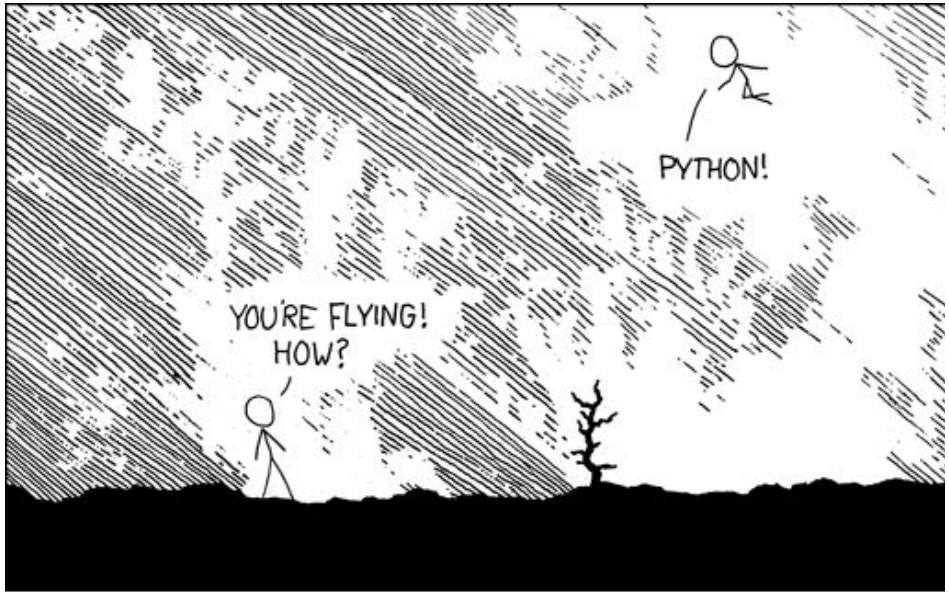


https://xkcd.com/353/

Introduction to Python Programming

me = {'name': 'Phondanai Khanti', 'WORK_PLACE': 'NECTEC', 'Email': 'phondanai.khani@nectec.or.th'}

Agenda

- A Very Brief History of Python
- Expression and Values
- Types
- Variables and Computer Memory
- Designing and Using Functions
- Working with Text
- Making Choices
- A Modular Approach to Program Organization
- Using Methods
- Lists & Dictionaries
- Repeating Code Using Loops
- Object-Oriented Python
- Flask Introduction

Guido van Rossum

December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas.

Wikipedia

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language

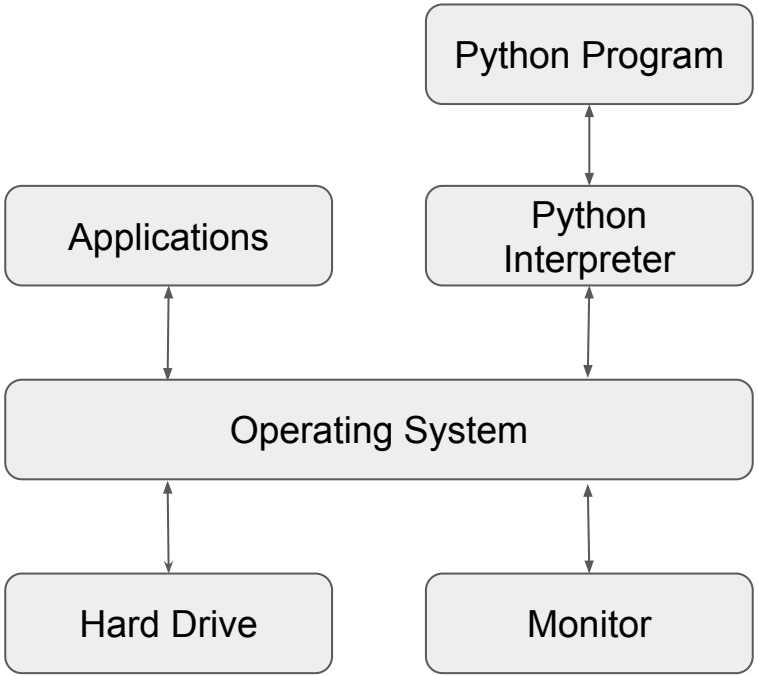
Implementation

- CPython
- IronPython
- Jython
- PyPy
- MyPy
- HyLang

What Python can do?

	Library
Scientific	SciPy, NumPy, Pandas, IPython, Matplotlib
Machine Learning, AI	NLTK,SciKit-Learn,OpenCV, TensorFlow
Web Development, Network Programming	Django, Flask**, Pyramid, Twisted, BeautifulSoup, Requests
DevOps	Ansible, SaltStack
3D & Game	Blender, PyGame
Mobile App	Kivy

How Python run your code?



Python2 or Python3?

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.9	bugfix	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

https://www.python.org/downloads/ - last access 7/7/2021

Syntax, Expressions and Values

Arithmetic

```
>>> 42
42

>>> 42 + 2
44

>>> 42 - 2
40

>>> 42 * 2
84

>>> 42 / 2
21.0

>>> 42 // 4
10

>>> 42 % 4
2

>>> 3 ** 4
81

>>> -42 # negation, unary operator
-42

>>> --42
42
```

Arithmetic : Operator Precedence

Precedence	Operator
Highest Lowest	** - *, /, //, % +, -

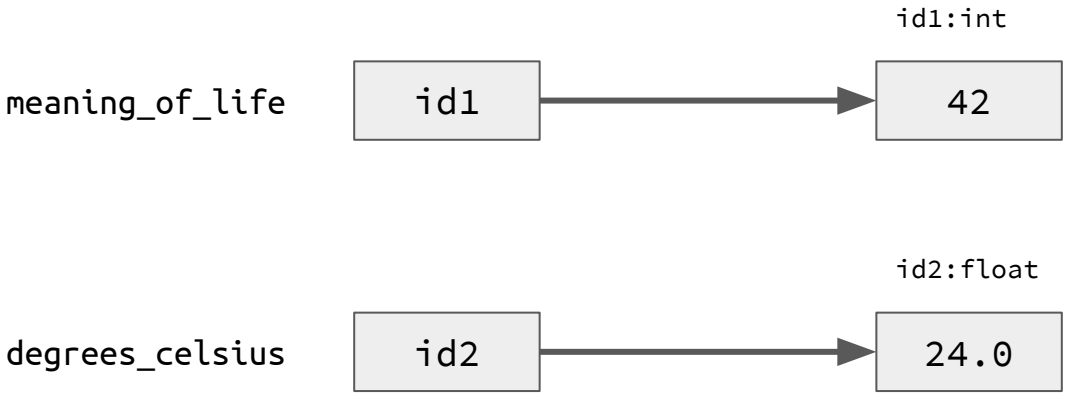
Variables and Computer Memory

```
>>> meaning_of_life = 42          # assignment

>>> degrees_celsius = 24.0

>>> difference = 100 - degrees_celsius # evaluate first then assign

>>> difference
76.0
```

Assignment Statement

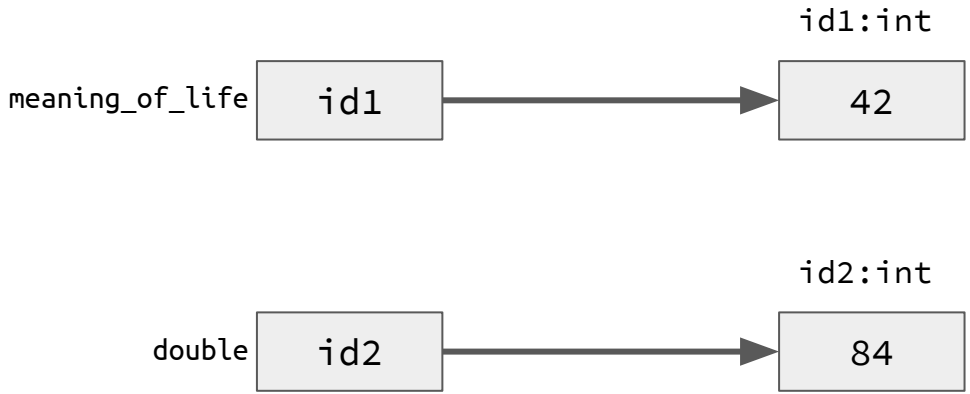
<<variable>> = <<expression>>

Memory Model



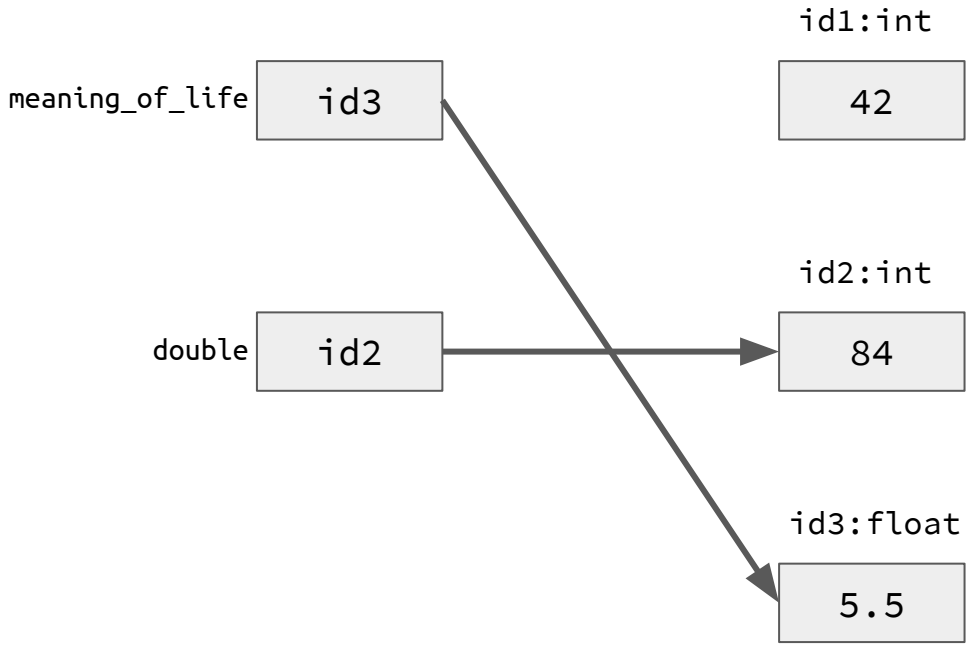
Memory Model (cont’)

```
>>> meaning_of_life = 42

>>> double = meaning_of_life * 2
```



Memory Model (cont’)

```
>>> meaning_of_life = 42

>>> double = meaning_of_life * 2

>>> meaning_of_life = 5.5
```



Something Went Wrong!

```
>>> 42 + foo
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
NameError: name "foo" is not defined

>>> 1 +
  File "<stdin>", line 1
    2 +
      ^
SyntaxError: can't assign to literal
```

Types

Type	Syntax example
bool	True False
bytes	b'Some ASCII' bytes([119, 105, 107, 105])
dict	{'pi': 3.14, 3: False}
float	3.141592653589793
integer	-2 0 65535
NoneType	None
set	{true, 0, 3.14}
str	"Hello world"
tuple	(3.1, 'string', False)

Type category	
Text	str
Numeric	int, float, complex
Sequence	list, tuple, range
Mapping	dict
Set	set
Boolean	bool
Binary	bytes, bytearray

Design and Using Functions

Built-in Functions

abs()	delattr()	hash()	memoryview()	set()	round()	locals()
all()	dict()	help()	min()	setattr()	max()	getattr()
any()	dir()	hex()	next()	slice()	hasattr()	classmethod()
ascii()	divmod()	id()	object()	sorted()	complex()	repr()
bin()	enumerate()	input()	oct()	staticmethod()	__import__()	vars()
bool()	eval()	int()	open()	str()	reversed()	range()
breakpoint()	exec()	isinstance()	ord()	sum()	map()	list()
bytearray()	filter()	issubclass()	pow()	super()	globals()	frozenset()
bytes()	float()	iter()	print()	tuple()	compile()	chr()
callable()	format()	len()	property()	type()	zip()	

https://docs.python.org/3/library/functions.html#built-in-functions - last access 2021/07/08

Functions That Python Provides

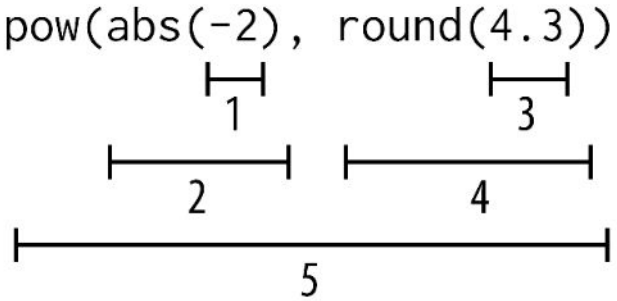
```
>>> abs(-42)
42

>>> abs(5.5)
5.5

>>> pow(2, 4)
16
```

General form

```
<<function_name>>(<<arguments>>)
```

Functions That Python Provides (cont’)

```
>>> pow(abs(-2), round(4.3))
```



```
help()
```

help()

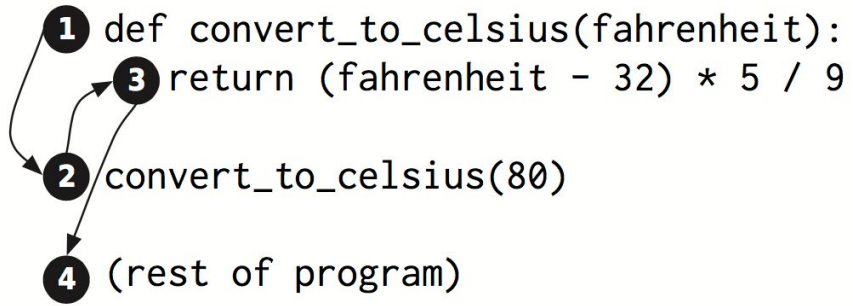
```
>>> help(abs)

Help on built-in function abs in module builtins:

abs(x, /)
    Return the absolute value of the argument.

>>> help(dir)
Help on built-in function dir in module builtins:

dir(...)
    dir([object]) -> list of strings

    If called without an argument, return the names in the current scope.
    . . . . . .
```

Defining Our Own Functions

```
>>> convert_to_celsius(212)
100

>>> convert_to_celsius(78.8)
26.0

>>> convert_to_celsius(10.4)
-12.0
```

Defining Our Own Functions (cont’)

```
>>> def convert_to_celsius(fahrenheit):
…     return (fahrenheit - 32) * 5 / 9
…
>>> convert_to_celsius(80)
26.666666666666668
```




```
def <<function_name>>(<<parameters>>):
    <<block>>
```

Using Local Variables

```
>>> def quadratic(a, b, c, x):
…         first = a * x ** 2
…         second = b * x
…         third = c
…         return first + second + third
…

>>> quadratic(2, 3, 4, 0.5)
6.0
```

```
>>> first
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'first' is not defined

>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
```

Working with Text

```
>>> 'slowpoke'
>>> "psyduck"
>>> "Barry Allen'   # SyntaxError

>>> '''This is
…   multi-line
…   string'''
>>> """Me
…
…   Too!"""

>>> 'This is "slowpoke!"'
'This is "slowpoke!"'
>>> "This is 'psyduck!'"
"This is 'psyduck!'"
```

```
>>> len('slowpoke')
8
>>> len("psyduck")
7
>>> "psyduck" + " golduck"
'psyduck golduck'

>>> len("""Me
…
…   Too!""") # What is the result?

>>> 'slowpoke' + 3   # TypeError

>>> 'slowpoke' * 3   # What about this?
```

Printing Information

```
>>> print(1+1)
2
>>> print("Lorem Ipsum")
Lorem Ipsum
>>> print("One\nTwo\nThree")
One
Two
Three

>>> numbers = """One
…  Two
…  Three"""
>>> print(numbers)
One
Two
Three
```

```
>>> print() # ??????

>>> print(1, 2, 3)
1 2 3

>>> print(1, 2, 3, sep=', ')
1, 2, 3
>>> pokemon_name = "Slowpoke"
>>> pokedex = 79
>>> print("Pokedex #{1} is
{0}".format(pokemon_name,pokedex))
```

Get information from the Keyboard

```
>>> species = input()
Mutant
>>> species
'Mutant'

>>> population = input()
67959000
>>> population
'67959000'
>>> type(population)
<class 'str'>
```

```
>>> population = int(input())
67959000
>>> population += 42
67959042

>>> species = input("Please enter a
species: ")
Please enter a species: Mutant X
>>> print(species)
Mutant X
```

Making Choices : Boolean Type

True

False

Boolean Operators

```
>>> not True
False
>>> not False
True

>>> True and True
True

>>> False or False
False

>>> type(False)
<class 'bool'>
```

```
Relational Operators
>          →    Greater than
<          →    Less than
>=         →    Greater than or equal to
<=         →    Less than or equal to
==         →    Equal to
!=         →    Not equal to

>>> 42 > 32
True
>>> 23.1 == 23
False
>>> 42 != 42
False
```

Comparing Strings

```
>>> 'A' < 'a'
True
>>> 'A' > 'z'
False
>>> 'abc' < 'abd'
True

>>> 'Jan' in '01 Jan 1870'
True
>>> 'Feb' in '01 Jan 1870'
False
```

```
>>> 'a' in 'abc'
True

>>> '' in 'abc'
True

>>> '' in ''
True
```

Choosing Which Statements to Execute

```
If <<condition>>:
    <<block>>
```

What if . . .

```
>>> ph = float(input('Enter the pH level:
'))
Enter the pH level: 6.0
>>> if ph < 7.0:
…       print(ph, "is acidic.")
…
6.0 is acidic.
```

```
>>> ph = float(input('Enter the pH level:
'))
Enter the pH level: 8.5
>>> if ph < 7.0:
…       print(ph, "is acidic.")
…   elif ph > 7.0:
…       print(ph, "is basic.")
…   else:
…       print(ph, "is neutral")
8.5 is basic.
```

Nested if

```
value = input('Enter the pH level: ')

if len(value) > 0:
    ph = float(value)
    if ph < 7.0:
        print(ph, "is acidic.")
    elif ph > 7.0:
        print(ph, "is basic.")
    else:
        print(ph, "is neutral.")
else:
    print("No pH value was given!")
```

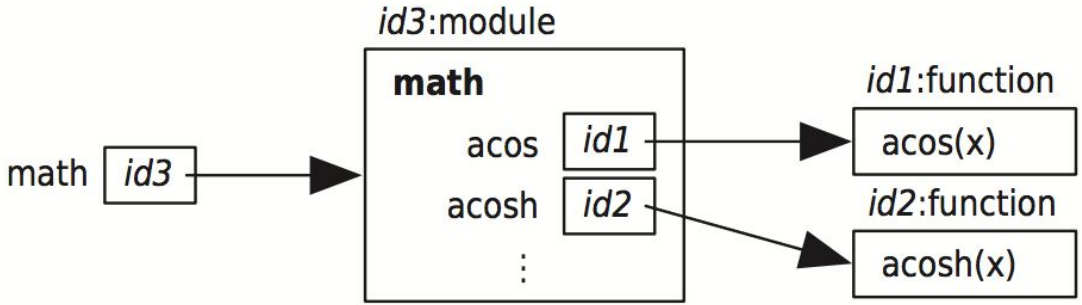
```
Enter the pH level: 6.0
ph 6.0 is basic

Enter the pH level: 8.7
ph 8.7 is basic

Enter the ph level:
No pH was given!
```

A Modular Approach to Program Organization

```
>>> import this
```

Importing Modules

```
>>> import math
>>> type(math)
<class 'module'>

>>> help(math)

>>> sqrt(9) # NameError
>>> math.sqrt(9)
3.0
>>> math.pi = 3 # No! No! NOOOOO!
>>> radius = 5
>>> print('area is', math.pi * radius ** 2)
area is 75
```



Defining Your Own Modules

```
# temperature.py

def convert_to_celsius(fahrenheit):
    """ (number) -> float

    Return the number of Celsius degrees
equivalent to fahrenheit degrees.

    >>> convert_to_celsius(75)
    23.88888888888889
    """
    return (fahrenheit - 32.0) * 5.0 / 9.0
```

```
def above_freezing(celsius):
    """ (number) -> bool

    Return True if temperature celsius degree is
above freezing.

    >>> above_freezing(4.2)
    True
    >>> above_freezing(-2)
    False
    """
    return celsius > 0
```

```
>>> import temperature
>>> temperature.convert_to_celsius(120)
48.88888888888889
>>> temperature.above_freezing(24)
True
```

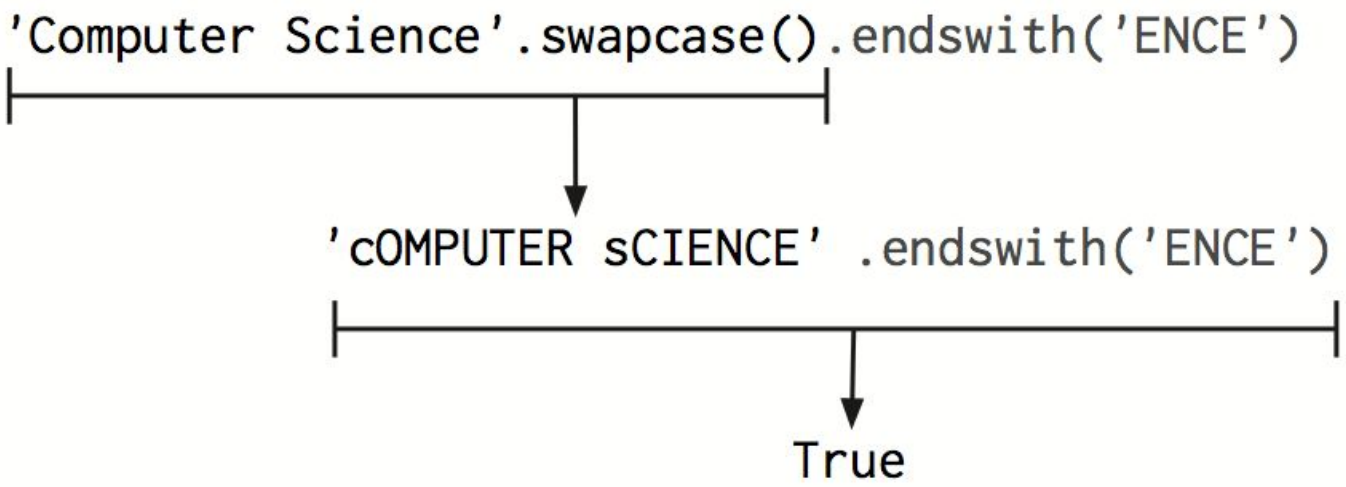
Using Methods

```
>>> help(str)
Help on class str in module builtins:

class str(object)
 |   str(object[, encoding[, errors]]) -> str
```

```
>>> str.capitalize('slowpoke')
'Slowpoke'
>>> str.count("Ever since my baby went away It's
been the blackest day, it's been the blackest
day", 'y')
5

>>> 'slowpoke'.capitalize()
'Slowpoke'
>>> "Ever since my baby went away It's been the
blackest day, it's been the blackest
day".count('y')
5
```

Chaining Method Calls

```
>>> 'Computer Science'.swapcase().endswith('ENCE')
True
```



Using Methods

```
<<expression>>.<<method_name>>(<<arguments>>)
```

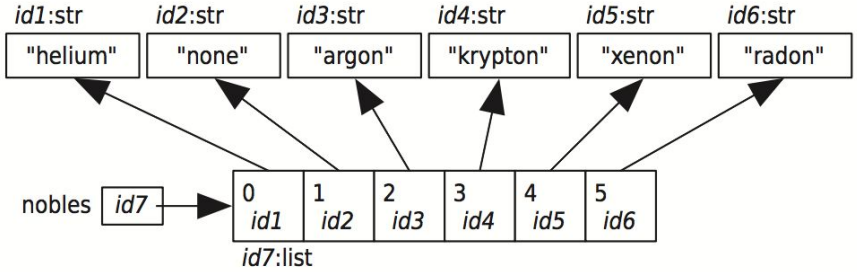
Lists

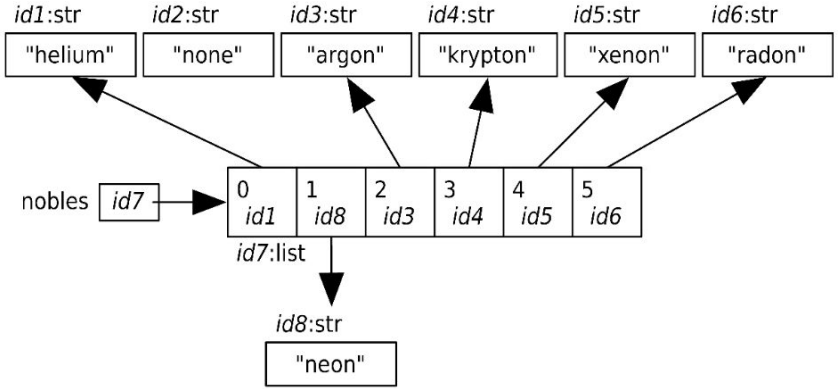
```
# list store sequences
>>> my_list = []      # empty list
>>> your_list = [3, 4, 5]
>>> my_list.append(1)
>>> my_list.append(2)
>>> my_list.append(4)
>>> my_list.append(3)
>>> my_list.pop()
>>> my_list.append(3)

>>> krypton = ['Krypton', 'Kr', -157.2,
-153.4]
>>> krypton[1]
'Kr'
```

```
>>> my_list[0]
1
>>> my_list[-1]
3
>>> my_list[4] # will Raise IndexError
>>> my_list[1:3]
[2, 4, 3]
>>> my_list[::2]
[1, 4]
>>> my_list[::-1]
```

Modifying Lists

```
>>> nobles = ['helium', 'none', argon', 'krypton',
'xenon', 'radon']

>>> nobles[1] = 'neon'
>>> nobles
['helium', 'neon', argon', 'krypton', 'xenon', 'radon']
```

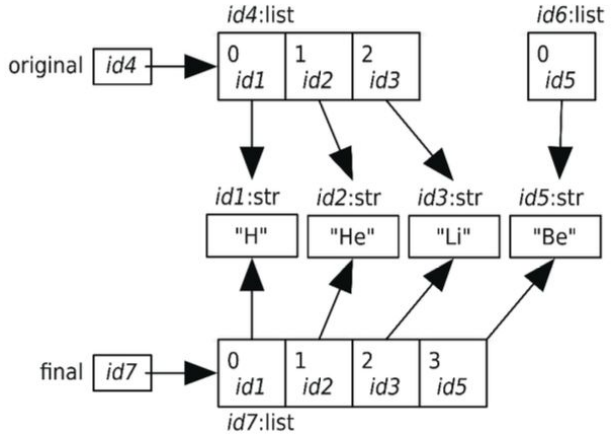




Operations on Lists

```
>>> half_lives = [887.7, 24100.0, 6563.0, 14,
373300.0]
>>> len(half_lives)
5
>>> max(half_lives)
373300.0
>>> min(half_lives)
14
>>> sum(half_lives)
404864.7
>>> sorted(half_lives)
[14, 887.7, 6563.0, 24100.0, 373300.0]
>>> half_lives
[887.7, 24100.0, 6563.0, 14, 373300.0]
```

```
>>> original = ['H', 'He', 'Li']
>>> final = original + ['Be']
>>> final
['H', 'He', 'Li', 'Be']
```



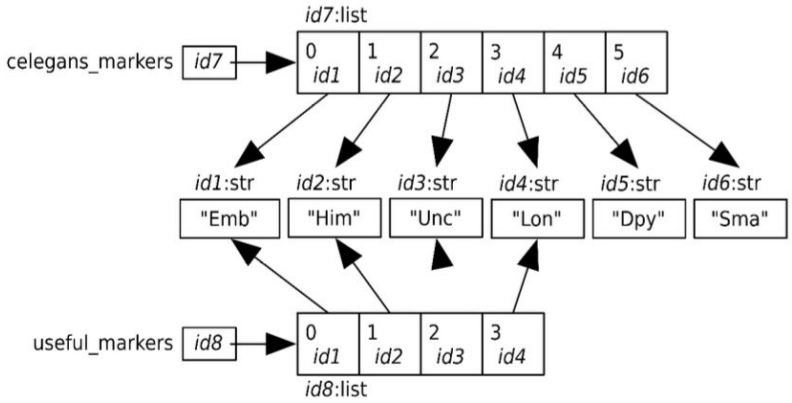
Operations on Lists

```
>>> ['H', 'He', 'Li'] + 'Be' # TypeError

>>> metals = ['Fe', 'Ni']
>>> metals * 3
['Fe', 'Ni', 'Fe', 'Ni', 'Fe', 'Ni']

>>> del metals[0]
>>> metals
['Ni']
```

```
>>> nobles = ['helium', 'neon', argon', 'krypton',
'xenon', 'radon']

>>> 'xenon' in nobles
True


>>> gas = input('Enter a gas: ')
Enter a gas: argon
>>> if gas in nobles:
...     print('{} is noble.'.format(gas))
...
argon is noble.
```

Slicing Lists

```
>>> celegans_markers = ["Emb", "Him", "Unc", "Lon",
"Dpy", "Sma"]

>>> celegans_markers[0:4]
["Emb", "Him", "Unc", "Lon"]
>>> usefule_makers = celegans_markers[0:4]

>>> celegans_copy = celegans_markers[:]
```



List Methods

```
>>> colors = ["red", "orange", "green"]
>>> colors.extend(["black, "blue"])
>>> colors
["red", "orange", "green", "black, "blue"]

>>> colors.append("purple")
>>> colors
["red", "orange", "green", "black, "blue", "purple"]
>>> colors.insert(2, "yellow")
>>> colors
["red", "orange", "yellow", "green", "black", "blue", "purple"]
>>> colors.remove("black")
>>> colors
["red", "orange", "yellow", "green", "blue", "purple"]
```

List of Lists

```
>>> elements = [["Copper", "Cu"], ["Tellurium", "Te"]]
>>> elements[0]
["Copper", "Cu"]
>>> elements[1]
["Tellurium", "Te"]
>>> len(elements)
2

>>> "You're so " + (elements[0][1] + elements[1][1]).capitalize()
"You're so Cute"
```

Storing Data Using Dictionaries

```
>>> empty_dict = {}
>>> my_dict = {"one": 1, "two": 2, "three", 3}
>>> my_dict["one"]
1
>>> list(my_dict.keys())
["two", "one", "three"]
>>> list(my_dict.values())
[2, 1, 3]
>>> for key, value in my_dict.items():
...     print(key, value)
...
one 1
two 2
three 3
```

Repeating Code Using Loops

```
>>> velocities = [0.0, 9.81, 19.62, 29.43]
>>> for velocity in velocities:
...     print("Metric:", velocity, "m/sec;",
...     "Imperial:", velocity * 3.28, "ft/sec")
...
Metric: 0.0 m/sec; Imperial: 0.0 ft/sec
Metric: 9.81 m/sec; Imperial: 32.1768 ft/sec
Metric: 19.62 m/sec; Imperial: 64.3536 ft/sec
Metric: 29.43 m/sec; Imperial: 96.5304 ft/sec
>>>
```

Processing Characters in Strings

```
>>> country = "United Kingdom"
>>> for ch in country:
...     if ch.isupper():
...         print(ch)
...
U
K
```

```
>>> country = "United States of America"
>>> for ch in country:
...     if ch.isupper():
...         print(ch)
...
U
S
A
```

Looping Over a Range of Numbers

```
>>> range(10)
range(0, 10)

>>> for num in range(10):
...     print(num)
0
1
2
3
4
5
6
7
8
9
```

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(3))
[0, 1, 2]
>>> list(range(0))
[]
>>> list(range(1, 5))
[1, 2, 3, 4]

>>> list(range(2000, 2050, 4))
[2000, 2004, 2008, 2012, 2016, 2020, 2024, 2028,
2032, 2036, 2040,2044, 2048]

>>> list(range(2050, 2000, -4))
[2050, 2046, 2042, 2038, 2034, 2030, 2026, 2022,
2018, 2014, 2010, 2006, 2002]
```

Processing Lists Using Indices

```
>>> values = [4, 10, 3, 8, -6]
>>> for num in values:
...     num = num * 2
...
>>> values
[4, 10, 3, 8, -6]
```

```
>>> values = [4, 10, 3, 8, -6]
>>> for i in range(len(values)):
...     values[i] = values[i] * 2
...
>>> values
[8, 20, 6, 16, -12]
```

Nesting Loops in Loops

```
>>> outer = ['Li', 'Na', 'K']
>>> inner = ['F', 'Cl', 'Br']
>>> for metal in outer:
...     for halogen in inner:
...         print(metal + halogen)
...
```

```
LiF
LiCl
LiBr
NaF
NaCl
NaBr
KF
KCl
KBr
```

Controlling Loops Using Break and Continue

```
>>> s = "C3H7"
>>> digit_index = -1
>>> for i in range(len(s)):
...     if s[i].isdigit():
...         digit_index = i
...         break
...
>>> digit_index
1
```

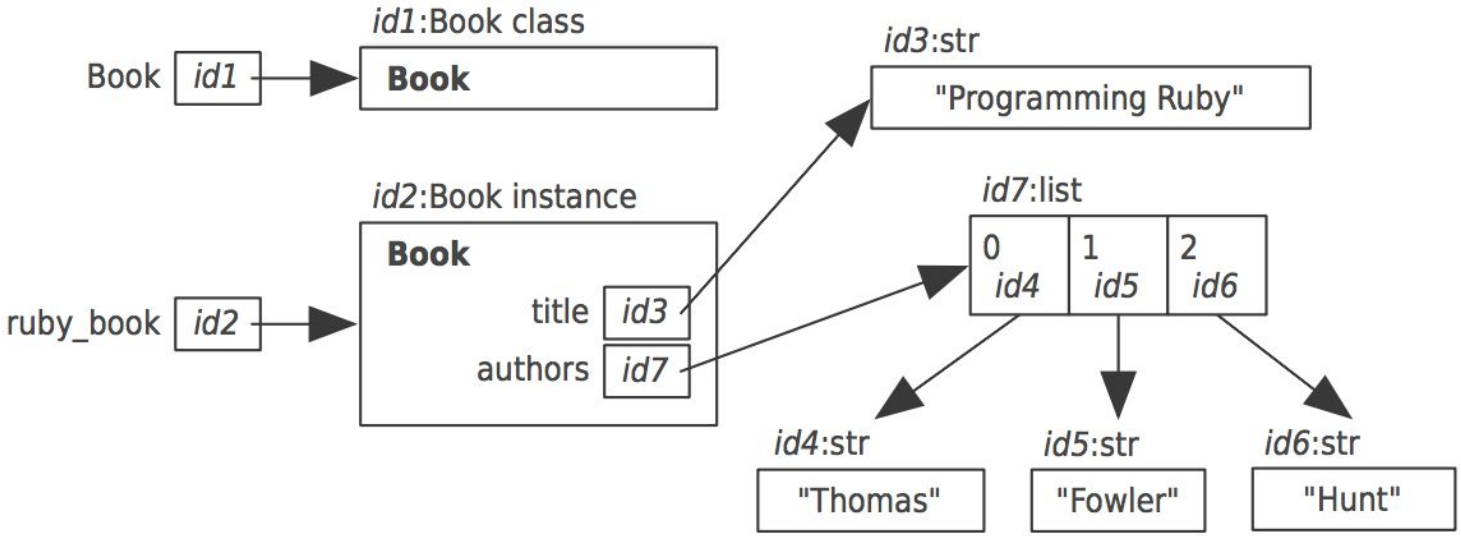
```
>>> s = "C3H7"
>>> total = 0
>>> count = 0
>>> for i in range(len(s)):
...     if s[i].isalpha():
...         continue
...     total = total + int(s[i])
...     count = count + 1
...
>>> total
10
>>> count
2
```

Object-Oriented Programming

```
>>> isinstance('abc', str)
True

>>> help(object)

>>> type(str)
<class 'type'>
```

```
>>> class Book:
…    """Information about a book."""
…

>>> type(Book)
<class 'type'>

>>> ruby_book = Book()
>>> ruby_book.title = "Programming Ruby"
>>> python_book.authors = ['Thomas', 'Fowler',
'Hunt']
```

Object-Oriented Programming (cont'd)



Writing a Method in Class Book

```
>>> str.capitalizing('slowpoke')
'Slowpoke'
>>> 'slowpoke'.capitalize()
'Slowpoke'
```

```
>>> Book.num_authors(ruby_book)
3
>>> ruby_book.num_authors()
3
```

```
class Book:
    """Information about a book."""

    def num_authors(self):
        """ (Book) -> int

        Return the number of   authors of
this book.
        """

        return len(self.authors)
```

Import book.py

```
>>> import book
>>> ruby_book = book.Book()
>>> ruby_book.title = "Programming Ruby"
>>> ruby_book.authors = ['Thomas', 'Fowler', 'Hunt']
>>> book.Book.num_authors(ruby_book)
3
>>> ruby_book.num_authors()
>>> isinstance(ruby_book, book.Book())
True
```

__init__

```
class Book:
    """Information about a book"""

    def __init__(self, title, authors):
        """ (Book, str, list_of_str) -> NoneType

        Create a new book titled title, with list
of author(s)
        """
        self.title = title
        self.authors = authors
```

```
>>> sicp = Book("Structure and Interpretation of
Computer Programs", ["Harold Abelson","Gerald Jay
Sussman","Julie Sussman"])

>>> sicp.title
'Structure and Interpretation of Computer
Programs'

>>> sicp.authors
['Harold Abelson', 'Gerald Jay Sussman', 'Julie
Sussman']
```

Inheritance

```
class Member:
    """A member of university"""

    def __init__(self, name, address, email):
        """(Member, str, str, str) -> NoneType

        Create a new member named name, with home address and email address.
        """

        self.name = name
        self.address = address
        self.email = email

    def hello(self):
        print("Hello {}".format(self.name))
```

Inheritance (cont’d)

```
class Student(Member):
    """A student member at a university."""

    def __init__(self, name, address, email, student_id):
        """(Member, str, str, str, str) -> NoneType

        Create a new student named name, with home address and email address,
        student ID student_id, an empty list of courses taken, and an
        empty list of current courses.
        """

        super().__init__(name, address, email)
        self.student_id = student_id
```

Pip & Virtual Environment

Pip

- Package manager for Python.
  - Install
    - $ pip install Flask
    - $ pip install numpy pandas tensorflow==2.0.0
    - $ pip install -r requirements.txt # install list of package file.
  - Uninstall
    - $ pip uninstall Flask
  - Listing installed packages
    - $ pip list
    - $ pip freeze # output file can be uses by `pip install -r`

Virtual Environment

- Standalone Python environment.
- Creating virtual environment
  - $ python3 -m venv path/to/env
- Using the created environment
  - $ source /path/to/env/bin/activate # Bash/Zsh
  - $ source /path/to/env/bin/Activate.ps1 # PowerShell Core
  - C:\> path\to\env\Scripts\activate.bat # cmd
  - PS C:\> <venv>\Scripts\Activate.ps1 # PowerShell

Flask

web development,
one drop at a time

Flask

- Micro web framework written in Python.
- CKAN is powered by Flask as a core web framework.
- Components.
  - Werkzeug - WSGI toolkits
  - Jinja - Template Engine
  - MarkupSafe- MarkupSafe escapes characters so text is safe to use in HTML and XML.
  - ItsDangerouse - Data serialization library
  - Click - Command line utility.

```
15  dominate==2.4.0          # via -r requirements.in
16  feedgen==0.9.0           # via -r requirements.in
17  flask-babel==1.0.0       # via -r requirements.in
18  flask-multistatic==1.0   # via -r requirements.in
19  flask==1.1.1             # via -r requirements.in, flask-babel, flask-multistatic
20  idna==2.10               # via requests
21  itsdangerous==2.0.1      # via flask
22  jinja2==2.11.3           # via -r requirements.in, flask, flask-babel
23  lxml==4.6.3              # via feedgen
24  mako==1.1.4              # via alembic
```

https://github.com/ckan/ckan/blob/master/requirements.txt

Flask - Extensions

- flask-sqlalchemy: Adds SQLAlchemy support to Flask
- Flask-Migrate: SQLAlchemy database migrations for Flask applications.
- flask-wtf: Simple integration of Flask and WTForms, including CSRF, file upload and Recaptcha integration.
- Flask-SocketIO: Socket.IO integration for Flask applications.
- flask-login: Flask user session management.
- flask-babel: i18n and l10n support for Flask based on Babel and pytz.

Flask - Quick start

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello World"
app.run()

* Serving Flask app "__main__" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Free Online Books

http://greenteapress.com/wp/think-python-2e/

http://www.diveintopython3.net/

Free Flask online resource

- The Flask mega tutorial
- Flask Quickstart


Thank You
Questions?