Lecture 1a
Intro to Python

Why program?

Computers Aren’t That Smart (yet)

Computers will follow orders precisely

Do cool stuff

Program:

communicate a computational process

Why Python?

- Powerful open source programming language
- Clear and readable syntax
- Natural expression of procedural code
- Popular

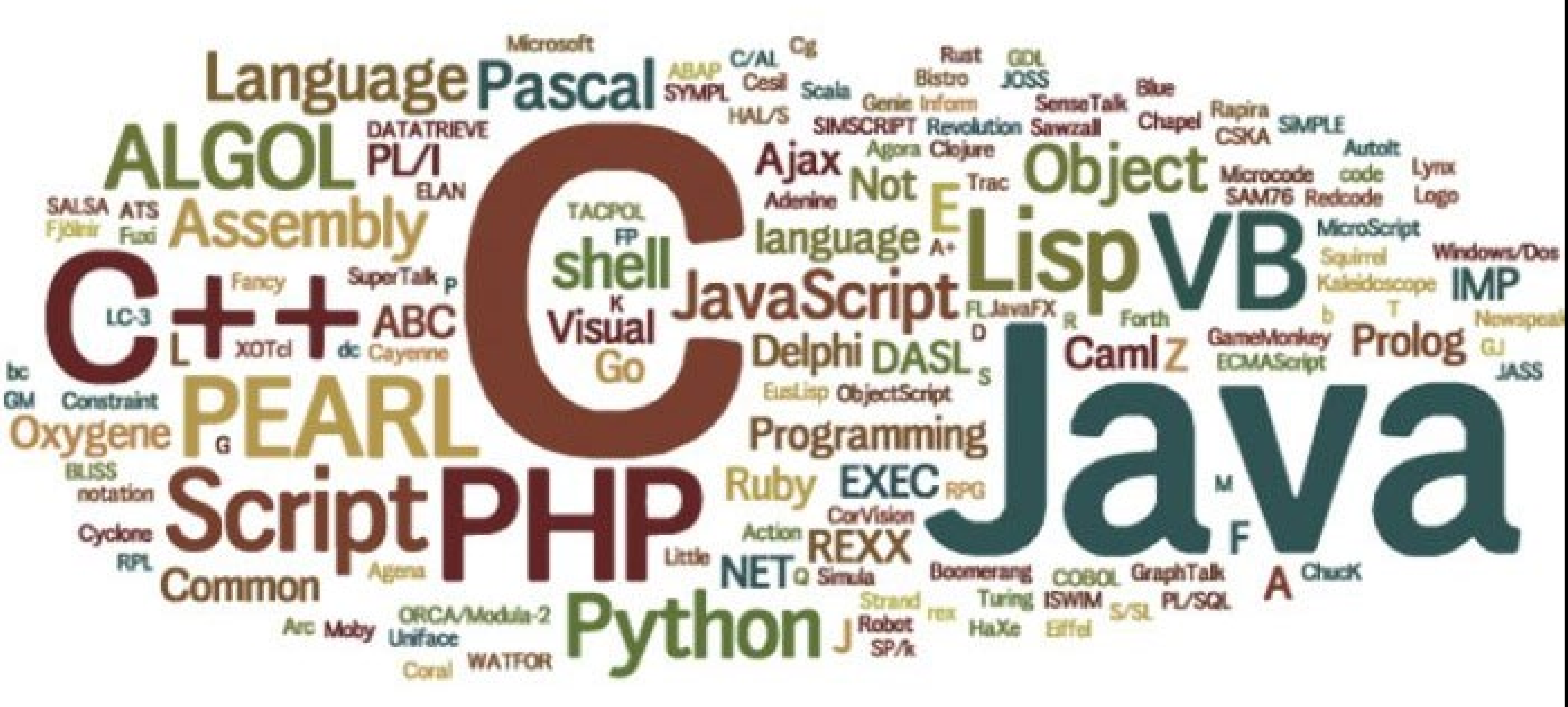
Read an article about …

Simplicity

Computational Thinking

Futile to Teach a Language

(Here today, gone tomorrow)



Elements of Programming

1. Primitives

- Numbers:

  `4 , 7/2 , 428.3`

- Simple operators:

  `+ , - , * , /`

- Symbols:

  `a , pi , current_time`

2. Means of Combination

`5 + 3`	8
`((5 + 3)-(2 * 3))`	2

3. Means of Abstraction

```
7 + 6        13

a = 7

a + 6        13

□ + 6
```

Variables

- Start with ‘a’ - ‘z’ or ‘A’ - ‘Z’ or ‘_’
- Contain only alphanumeric characters or ‘_’
- Case sensitive

```
Coco_Lee != coco_lee
```

Variables

- Avoid reserved keywords e.g. **`if`**
- Python convention: lower case letters separated by ‘_’
  - e.g. **`count_change`**

Types

int	`8, 45, 1234`
float	`2.3, 3.141592653`
bool	`True, False`

Types

str	`“ijc”` `‘ijc’`
None	Absence of value, null

type(..)

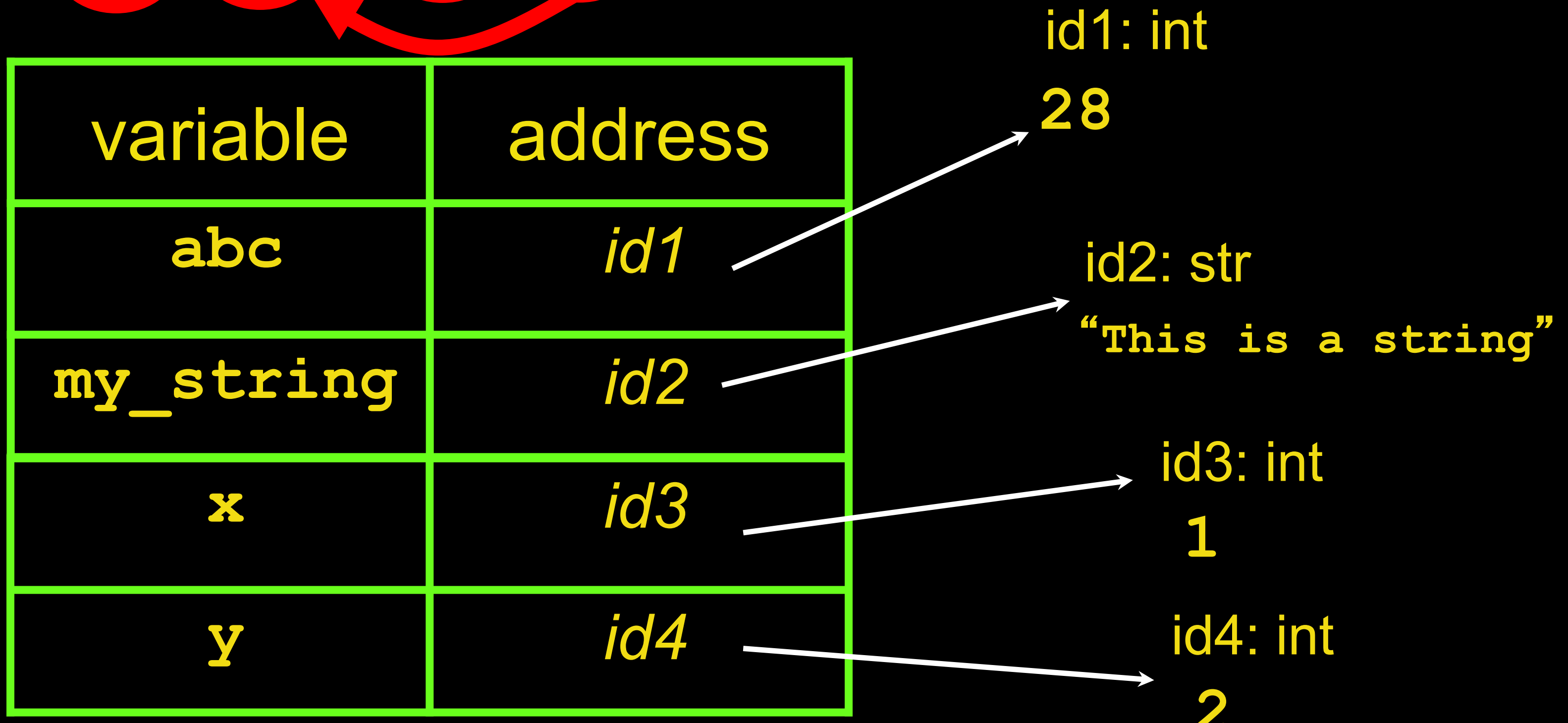
```
>>> type(123)
<class ‘int’>

>>> type(‘123’)
<class ‘str’>

>>> type(None)
<class ‘NoneType’>
```

Type conversion

```
>>> str(123)
’123’
>>> float(’45.2’)
45.2
>>> int(23.8)
23
>>> int(’ijc’)
ValueError!
```

Assignment

```
>>> abc = 28
>>> my_string = 'This is a string.'
>>> x, y = 1, 2
```



variable	address
abc	*id1*
my_string	*id2*
x	*id3*
y	*id4*

id1: int
28

id2: str
"This is a string"

id3: int
1

id4: int
2

Operators

Arithmetic: + - * / ** // %

```
>>> a = 2 * 3
>>> a
6
>>> 2 ** 3
8
```

Operators

Arithmetic: + - * / ** // %

```
>>> 11 / 3
3.6666666666666666

>>> 11 // 3
3


>>> 11 % 3
2
```

IDLE time with Jupyter

Your First Program

```
# This program says hello and asks for my name.

print('Hello World!')
print('What is your name?')        # ask for their name
myName = input()
print('It is good to meet you, ' + myName)
print('The length of your name is: ')
print(len(myName))
print()
print('What is your age?')         # ask for their age
myAge = input()
print('You will be ' + str(int(myAge)+1) + ' in a year.')
```

Dissecting Your Program

The *Comments*

The *print()* Function

The *input()* Function

The *len()* Function

The *str()*, *int()*, *float()* Functions

Dissecting Your Program

The *Comments*

```
# This program says hello and asks for my name.

# ask for their name

# ask for their age
```

Dissecting Your Program

The *print()* Function

```
print('Hello World!')
print('What is your name?')
print('The length of your name is: ')

print('It is good to meet you, ' + myName)

print()
```

Dissecting Your Program

The *input()* Function

```
myName = input()

myAge = input()
```

Dissecting Your Program

The *len()* Functions

```
myName = input()
print('The length of your name is: ')
print(len(myName))
```

Since we can type … ,

```
print('It is good to meet you, ' + myName)
```

Why don’t we type … ?

```
print('The length of your name is: ' + len(myName))
```

Dissecting Your Program

The *str(), int(), float()* Functions

```
myAge = input()
print('You will be ' + str(int(myAge)+1) + ' in a year.')
```

Dissecting Your Program

The *str()*, *int()*, *float()* Functions

Try This!

```
>>> str(0)          ‘0’
>>> str(-3.14)      ‘-3.14’
>>> int(’42’)        42
>>> int(’42.5’)      Error !
>>> int(‘-99’)       -99
>>> int(1.25)        1
>>> int(1.99)        1
>>> int(‘twelve’)    Error !
>>> float(‘3.14’)    3.14
>>> float(10)        10.0
```

Dissecting Your Program

myAge = ‘4’

The *str()*, *int()*, *float()* Functions

```
myAge = input()
print('You will be ' + str(int(myAge)+1) + ' in a year.')

print('You will be ' + str(int(‘4’)+1) + ' in a year.')
print('You will be ' + str(4 + 1) + ' in a year.')
print('You will be ' + str(5) + ' in a year.')
print('You will be ' + ‘5’ + ' in a year.')
print('You will be 5 in a year.')
```

Text and Number Equivalence

Although the string value of a number is considered a completely different value from the integer or floating-point version, an integer can be equal to a floating-point.

```
>>> 42 == ‘42’           False
>>> 42 == 42.0           True
>>> 42.0 == 0042.000     True
```

Python makes this distinction because strings are text, while integers and floats are both numbers.