Introduction to Computer Science:

Python programming

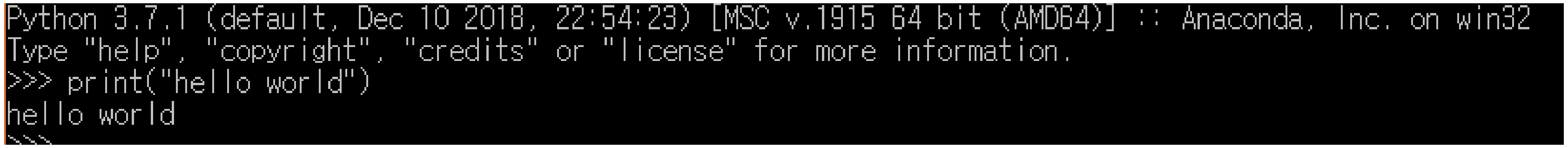
Mar. 2020

Honguk Woo

Contents

“hello world”

- Let’s write and execute the first code “hello world” as usual; this is a long-standing custom in the field of computer science^^
- After installing the python v3.7 (www.python.org/downloads)
    - When checking the web site recently, its latest version is 3.8.2 (but, any version will be fine as long as it’s about version 3.x, not 2.x)
- Start the python interpreter, and then type
- **print(“hello world“)**

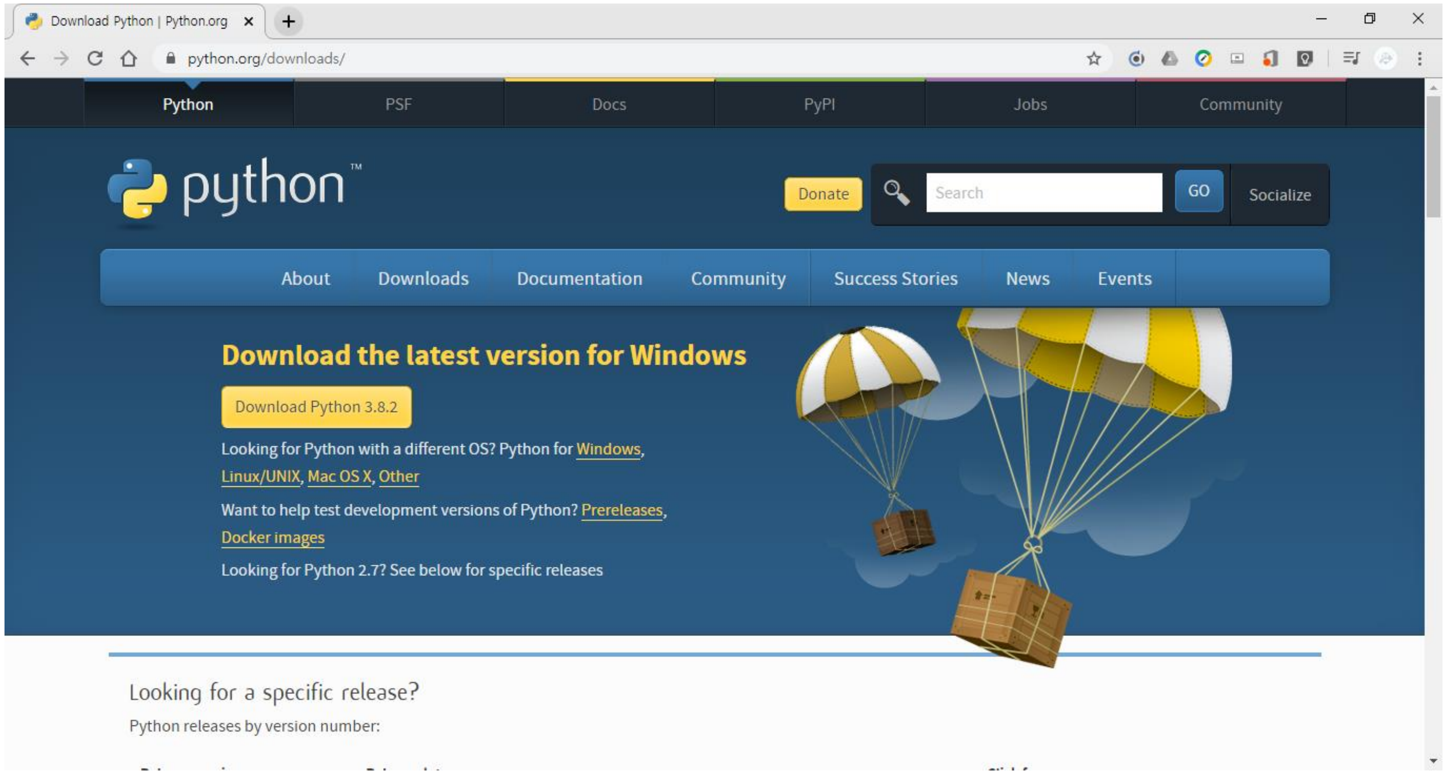


```
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>>
```

- “>>>” indicate where the python interpreter is waiting for a command
- You can use IDE (e.g., IDLE, jupyter notebook - https://jupyter.org/) that includes interpreter and editor functionalities

python.org

- You can visit this *python community site* for installation and studying



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

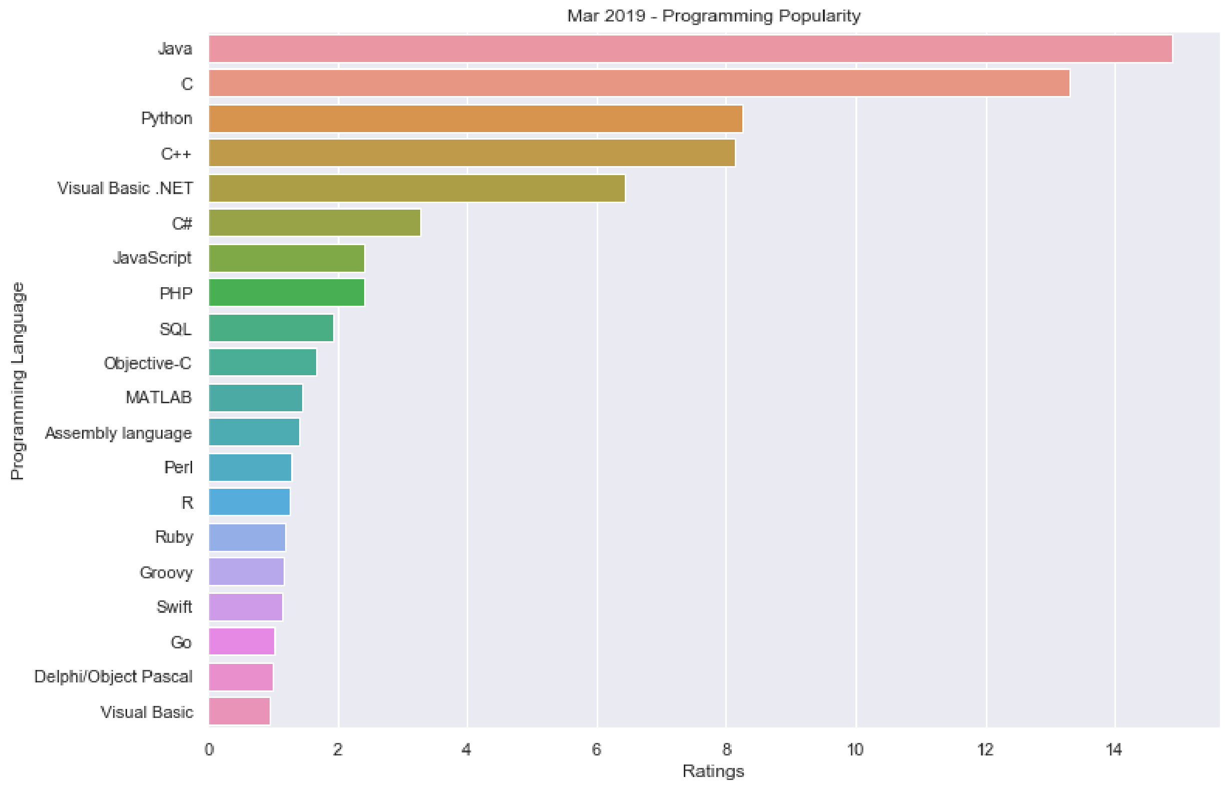
Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

Looking for a specific release?

Python releases by version number:

Programming language popularity



https://towardsdatascience.com/visualize-programming-language-popularity-using-tiobeindexpy-f82c5a96400d

Reference sites

- python.org
- www.w3schools.com/python/
- www.py4e.com
- wikidocs.net/book/1 (in Korean)

Data type - Numbers

- Python variables
  - no need to explicit declaration
  - **automatically declared** when **assigning** (=) a value to a variable

- Assigning (=) means :

  <variable> = <object>

  variable (name) is bound to object

- int

```
>>> a = 3
>>> b = 123456789
>>> a
3
>>> b
123456789
```

- float

```
>>> e = -0.123456789
>>> f = 3.1
>>> e
-0.123456789
>>> f
3.1
```

- type

```
>>> type(f)
<class 'float'>
```

overflow ?

- We learned from the previous “data representation” class that each integer value in C programming is represented by “4-bytes”, and has the specific limit on its range.
- Python ***int*** data is different from int in C : no limit to how long an integer value can be.
- *sys.getsizeof()* returns the size of an object in bytes

```
>>> a = 10000000000000000000000000000000000000000000000000000000000000000
>>> print(a)
10000000000000000000000000000000000000000000000000000000000000000
>>> print(a+1)
10000000000000000000000000000000000000000000000000000000000000001
>>> type(a)
<class 'int’>
>>> import sys
>>> sys.getsizeof(a)
?
```

- *What about float ?*

float follows IEEE standard

- Floating-point numbers

```
>>> b = 1.79e399
>>> print(b)
inf
>>> b = 1.79e300
>>> print(b)
1.79e+300
>>> type(b)
<class 'float'>
>>>
>>> c = 1.111111111111111111111111111111111111111111111111
>>> c
1.1111111111111112
>>> sys.getsizeof(c)
16
```

Data type - String

- Characters with single quotes or double quotes
- + concatenation

```
>>> a = ‘Sungkyunkwan’
>>> b = ‘University’
>>> c = a + ‘ ’ + b
>>> c
‘Sungkyunkwan University’
```

- slice

```
>>> c[0:4]
‘Sung’
```

String index

```
>>> a = ‘Sungkyunkwan’
>>> b = ‘University’
>>> c = a + ‘ ’ + b
>>> c
‘Sungkyunkwan University’
>>> c[1]
?

>>> c[-1]
?

>>> c[-2]
?
```

- The index -1 refers to the last item, -2 refers to the second last item

In python, everything is an object

```
>>> a = ‘xx’ + 3
TypeError: unsupported operand type(s) for +: 'int' and 'str’

>>> a = ‘xx’,  3
?
>>> a = 'xx', 'yy'
>>> a
?
>>> a = ‘xx’  3
?
>>> a = 'xx' 'yy‘
>>> a
?
```

input & print

- input() :
  - Placeholder {} for variables in a string and format() method

```
print(“input a number : ")
a = input()
print(“input a number : ")
b = input()
result = int(a) * int(b)
print("{0} * {1} = {2}".format(a, b, result))
```

- Cast : int(), float(), complex()

```
>>> float('3.4567')
3.4567
>>> complex('1+2j')
(1+2j)
```

input & print

- Placeholder with numbers or names (curly brackets)

```
print(“input a number : ")
a = input()
print(“input a number : ")
b = input()
result = int(a) * int(b)

print("{0} * {1} = {2}".format(a, b, result))

print("{op1} * {op2} = {res}".format(op2=b, op1=a, res=result))
```

Data type - list

- Ordered collection : sequence of objects (comma, square brackets)

```
>>> a = [1, 2, 3, 4]
>>> a
[1, 2, 3, 4]
>>> a[1]   #indexing
2
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> a[0:5]  #slicing
[1, 2, 3, 4, 5]
>>> a[5:]
[6, 7, 8, 9, 10]
>>> a[:3]
[1, 2, 3]
```

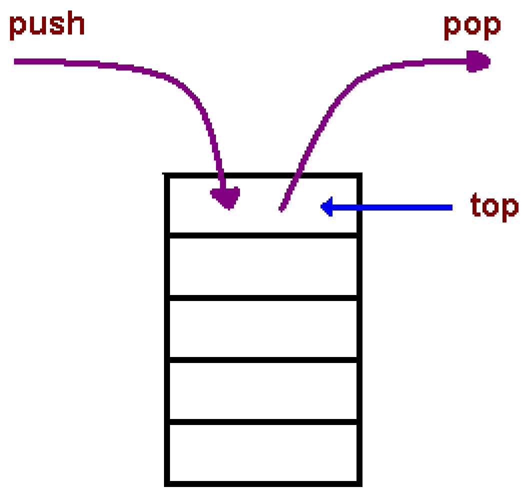
list

```
>>> a = [2, 4, 6, 8]
>>> b = [10, 12, 14]
>>> a + b
[2, 4, 6, 8, 10, 12, 14]

>>> a = [2, 4, 5, 8]
>>> a[2] = 6
>>> a
# ?

>>> a[3] = 10
>>> a[-1]
# ?
```

list & stack

- List can be used for Stack (Last-in, Frist-out)

```
>>> stack = [3, 4, 5]
>>> stack.append(6)  #push
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```



Ref. https://docs.python.org/3/tutorial/datastructures.html

Data type - tuple

- Ordered collection but Immutable (**comma**, parenthesis (sometimes, optional))

```
>>> a = (1, 2, 3)
>>> a
(1, 2, 3)
>>> a = (1, 2, 3, 4, 5, 6)
>>> a[:3]
(1, 2, 3)
>>> a[4:6]
(5, 6)
```

```
# Which one is a tutple ?  Packing values in a tuple

>>> a = (10)
>>> a = (10, 2)
>>> a = (10, )
>>> a = 10, 2
```

tuple - Immutable

```
>>> a = (1, 2, 3)
>>> b = (4, 5, 6)
>>> c = a + b
>>> c
(1, 2, 3, 4, 5, 6)
```

```
>>> a = (1, 2, 3)
>>> a[0]
1
>>> a[0] = 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Mutable vs. immutable

- Python objects are either mutable or immutable

- Mutable : objects can be manipulated and changed without the need to create a new copy (e.g., list)

- Immutable : objects can't be changed; assigning to immutable objects will produce an error (TypeError) (e.g., tuple)

- Mutable objects : list, dic, set
- Immutable objects : int, float, str, tuple

immutable

- **pyhon id**() function returns the "**identity**" of the **object**. The **identity** of an **object** is an integer, which is guaranteed to be unique and constant for this **object** during its lifetime

Immutable int

```
>>> a = 42
>>> id(a)
271178560
>>> a = 21
>>> id(a)
271178224
>>> a += 3
>>> id(a)
271178272
>>> a
24
```

mutable list

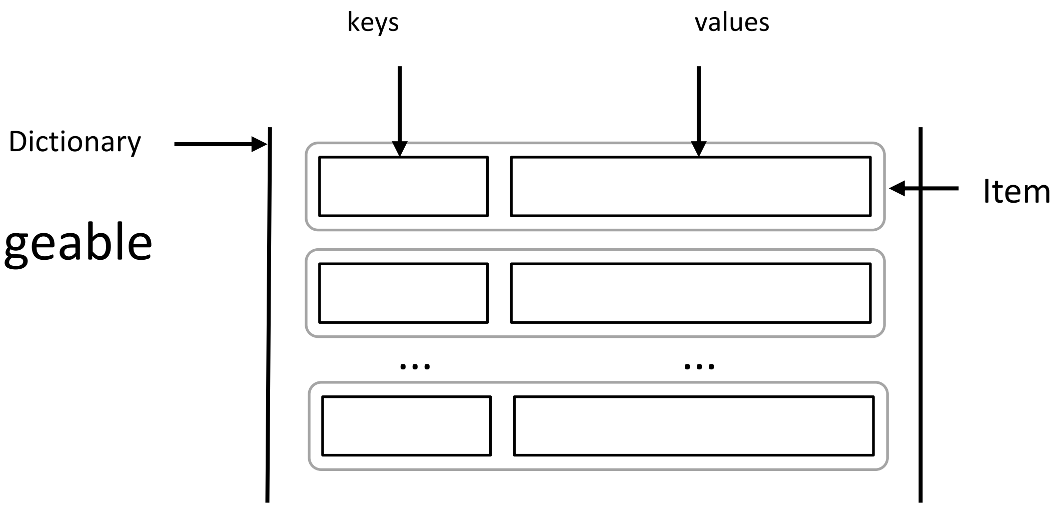
```
>>> b = [1, 2, 3]
>>> id(b)
59024528
>>> b[2]= 5
>>> id(b)
59024528
>>> b
[1, 2, 5]
>>> b += [6]
>>> b
[1, 2, 5, 6]
>>> id(b)
59024528
```

Discussion : Elements of collection

```
:
:
>>> d = [a, b, c]
>>> d
[5, 3, 6]
>>> a
5
>>> d[0] = -1

>>> a
?

>>> d
?
```

- *Is id(a) same as id(d[0]) ?*

Data type - dictionary



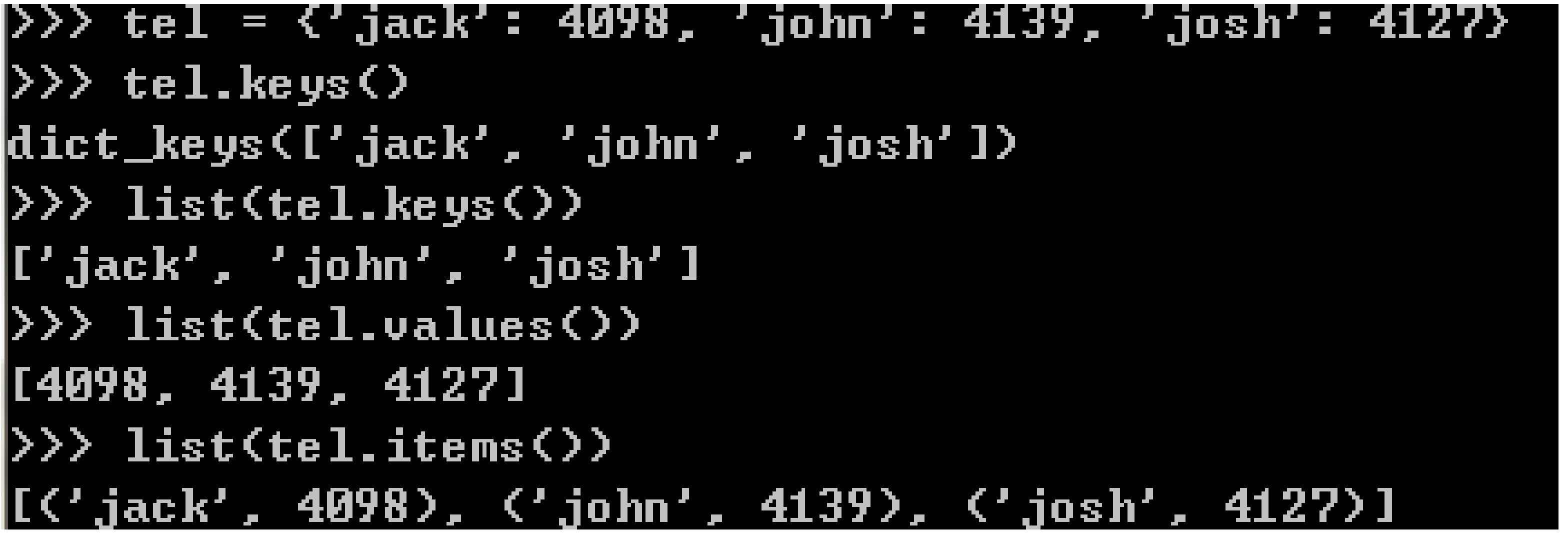
- Collection, unordered, changeable
- Indexed by Key
  - key, value pairs

```
>>> tel = {'jack': 4098, ‘john': 4139}
>>> tel[‘josh'] = 4127
>>> tel
{‘jack': 4098, ‘john': 4139, ‘josh': 4127}
>>> tel['jack']
4098
```

dictionary

- Keys
- Values
- items

```
>>> tel
{‘jack': 4098, ‘john': 4139, ‘josh': 4127}
>>> list(tel.keys())
[‘jack’, ‘john', 'josh']
>>> list(dic.values())
?

>>> list(tel.items())
?

>>> for k, v in tel.items():
        print(k, v)
```



```
>>> tel = {'jack': 4098, 'john': 4139, 'josh': 4127}
>>> tel.keys()
dict_keys(['jack', 'john', 'josh'])
>>> list(tel.keys())
['jack', 'john', 'josh']
>>> list(tel.values())
[4098, 4139, 4127]
>>> list(tel.items())
[('jack', 4098), ('john', 4139), ('josh', 4127)]
```

Data type - set

- unordered and unindexed collection of unique elements (comma, curly bracket)

```
a1 = set()

a2 = {1, 3, 5, 7}

a3 = set([1, 3, 5, 7])

a4 = set([1, 3, 3, 7])

a5 = set([x * 2 for x in range(1, 10)])

a6 = set("abac")
```

set

```
>>> a = {1, 2, 4}
>>> b = {1, 3, 5}
>>> a.intersection(b)
{1}
>>> a & b
{1}
>>> a.difference(b)
{2, 4}
>>> a - b
{2, 4}
>>> a.union(b)



>>> a.symmetric_difference(b)
{2, 3, 4, 5}
>>>
>>> a ^ b
{2, 3, 4, 5}
```

Python Condition

```
if condition :
        indentedStatementBlockForTrueCondition
else:
        indentedStatementBlockForFalseCondition
```

```
x = int(input("Please enter an integer: "))

if x < 0:
        x = 0
        print('Negative changed to zero')
elif x == 0:
        print('Zero')
elif x == 1:
        print('Single')
else:
```

Python loop

- for, while loop
  - for : iterate over a sequence
  - while : iterate while the condition is true

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

```
i = 1
while i < 6:
    print(i)
    i += 1
```

Python function

- Function : a block of code that only runs when it is called
  - the function is declared first and then executed when called

```
def hello():
       print(“Sungkyunkwan University”)

>>> hello()
Sungkyunkwan University
```

```
def abs(arg) :
    if (arg>=0) :
           result = arg
    else :
        result = arg*-1
    return result
```

Discussion : a function is an object

```
def print_something(a):
       print(a)

p = print_something

>>> p(123)
123
>>> p('abc')
abc
```

```
def plus(a, b):
       return a+b
def minus(a, b):
    return a-b

flist = [plus, minus]
>>> flist[0](1, 2)
3
>>> flist[1](1, 2)
-1
```