2/Jan/17

PYTHON Programming

Date / /
Page
STUDY BUDDIES

BASICS

- TPane

① # Variables & Maths

# variables

```
>>> <var nam> = <value>
    Ex    >>> x = 5
```

# Naming

```
>>> bigFoot = 4
```

↑ lowercase ↑ (Upper Case, <u>NO spaces</u>)

# Math · +, -, *, /, () Paranthesis, ** power, % remainder

# int & float

integer = 0, 15, 20 ...
float = 2.3, 6.4008, 7.0 ...

# float()

```
float (3) → 3.0
float (3)/2 → 1.5
float (3/2) → 1.0
```

# int ()

```
int (3.0) → 3
int (3.0) /2 → 1
```

# Other functions

abs()
sin(), cos(), floor()
ceil(), pow()

Ex pow() = power(a **

etc.

Date __/__/__
Page ____
STUDY BUDDIES

② Strings, Lists, tuples & Dictionaries

Strings

```
        x = 'Hello'     } both are acceptable
  or    x = "Hello"     }
```

Combining Strings

Use '+' to combine two strings together

```
Ex      x = "Hello"
        y = x + " World"
               ↑ space
   >>> y ⇒ ~~x~~ 'Hello World'   // output
```

Combining numbers + strings

str() : use to convert TO a STRING

```
Ex      x = "Hello"
        z = 10
        y = x + str(z)  →  Hello10
```

Special characters

placing numbers values within strings

syntax : "%d" % NUM = substitute INTEGER

```
        z = 10
Ex      y = "Hello %d" % z  →  Hello 10
```

Similarly, "%f" % NUM = substitute float

```
Ex      z = 10
        y = "Hello %f" % z  →  Hello 10.000000
```

∴ "%.3f" % NUM = substitute cut off FLOAT

```
Ex      y = "Hello %0.3f" % z  →  Hello 10.000
```

special characters

"\n" = newline character
"\t" = TAB character

Keyword 'IN'

used to CHECK if a value is written another value

Ex "Hello" in "Hello World"
↳ o/p True/false

[] LISTS — Its like vectors in C++

Created using '[]' square brackets

```
x = []
```

Ex
```
x = ["Hello", 4, 2.2]
```

Fun(CTIONS)

① append (VALUE) : Adds value to end of list

Ex
```
x = ["Hello", 4, 2.2]
x.append(5)
>>> x → ['Hello', 4, 2.2, 5]
```

② insert (location, VALUE) : insert value at a location

Ex
```
x = ["H", 4, 2.2, 5]
x.insert(1, 3.14)
>>> x → ~~['H', 4, 2.2, 5~~ ['H', 3.14, 4, 2.2, 5]
```

(3) pop (location) : remove and return the value at the location

```
           0      1       2  3 ~ 4
Ex  x =  ['H', 3.14 , 4, 2.2, 5 ]
     x.pop(1)
     >>> x  →  ['H', 4 , 2.2, 5]
```

(4) len (string or list) : return the TOTAL number of items within a string or list, short for length.

```
Ex   x = ["H', 4, 2.2, 5]
     len(x)   →    4

>>> len ("Hello")   →  5
```

(5) list (item) : Convert item to a list

```
Ex  y = list("Hello")
     >>> y  →  ['H', 'e', 'l', 'l', 'o']
```

(6) More 'IN' stuffs

```
Ex   y = ["Hello"]   →   f
        >>> y   →   ['H', 'e', 'l', 'l', 'o']

     >>> 's' in y   →  false
```

~~FE~~

Date / /
Page
STUDY BUDDIES

TUPLES ( )

just like lists but UNADJUSTABLE
i.e no add or delete fns. like in list

Ex x = ( )

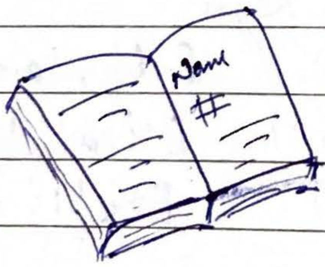
```
>>> x = ("Hello", 4, 5)
```

Tuples vs. Lists

Tuples	Lists
more memory efficient	Takes more memory
cannot be adjusted	Adjustable

DICTIONARIES { }

Used for binding KEYS to VALUES
(i.e just like a phone book!)



Ex
```
>>> sam = { }
>>> sam["Name"] = "Rajiv"
>>> sam["Age"] = 27
>>> sam
    {'Name': 'Rajiv', 'Age': 27 }
```

Retrieving the values in dictionary

dictionary[key] : GET and SET the value
del dict[key] : DELETE a value/key pair

Ex >>>sam ["Name"] → 'Rajiv'

Ex >>> del sam ['Age']
>>> sam → ['Name' : 'Rajiv']

EXERCISE-1

① a = str(int(2.23) + float(14)) + "tomatoes"
a = ?? 16.0tomates

② "ham Ham". upper() HAM HAM'
③ "HELLO world". lower() hello world
④ b = "I am Rajiv"
a) b.split() ['i', 'am', 'rajiv']
b) b.split("m") ['i a', 'rajiv']
c) b.join(a) ~~[illegible]~~ '1i am rajiv6i am rajiv. ...

⑤ "int : %.d, float %.0.5f" % (14.4, 55.2)
'int : 14, float 55.20000

⑥ L = [1, 6, 7, 26, 0, 3, 4, 5]

L[:]	[1, 6, 7, 26, 0, 3, 4, 5]
L[:2]	[1, 6]
L[2:]	[7, 26, 0, 3, 4, 5]
L[::2]	[1, 7, 0, 4]
L[1::2]	[6, 26, 3, 5]

Date__/__/__
Page ______
STUDY BUDDIES

EXERCISE - 2

①
```
L = [1, 2, 3]
x = [5, 6, L]
x = ???
```

②
```
y = [1, 2, 3, 1, 5, 2, 1, 3]
z = list ( set (y) )
z = ??
```

③
```
myDict = {14 : 'Ham', 20: 'sandwich' }
myDict.Keys( )
myDict.Values( )
len( myDict )
```

③ Conditionals: if, else, elif, operands, exampl

# IF = if (condition is true):
            do this!

Ex
```
if (5>2):
    print 'True'
```

# ELSE

Catches everything that does NOT meet prior condionals

Ex
```
if(5>2):
    print 'True'
else:
    print 'false'
```

ELIF (i.e else if)
- Comes AFTER 'if' statement
- Sets up another conditional

```
if (...):
    ....
elif (...):
    ....
else:
    ....
```

Ex

```
if (a>b):
    print 'a is big'
elif (b>a):
    print 'b is big'
else:
    print 'a equals to b'
```

Comparison Operators
<, <=, >, >=, ==, !=

'AND'
- Combines two conditionals
    - Only True if BOTH conditions are true!

Ex
```
if (7>5) and (5<7):
    print 'True'
```

'OR'
Ex No need ☺