Python - LiveCode Cheat Sheet

Comments

Comments allow you to add explanations and annotations to your code.

Python

```
# this is
commented
out
```

LiveCode

```
-- these
# are
// all
/*
commented
 out */
```

Variables

Variables are used to to store information, the stored value can be changed or accessed when you need it.

Python LiveCode

0/language-comparison/javascript-livecode-cheat-sheet/)
Python - LiveCode Builder Cheat Sheet (https://livecode.com/docs/9-5-0/language-comparison/python-livecode-builder-cheat-sheet/)
JavaScript - LiveCode Builder Cheat Sheet (https://livecode.com/docs/9-5-0/language-comparison/javascript-livecode-builder-cheat-sheet/)

Extending LiveCode (https://livecode.com/docs/9-5-0/extending-livecode/)

Whats New? (https://livecode.com/docs/9-5-0/whats-new/)

```
var =
"str"
var =
1
```

```
local
tVar
put
"str"
into
tVar
put 1
into
tVar
```

```
var["key"]
= "val"
```

```
put "val"
into
tVar["key"]
```

Control Structures

Control structures are used to control what code is executed and how many times.

Python LiveCode

```
for x in tVar:
    # do things
for x in
range(10):
```

do things

```
while x > 1:
x -= 1
```

```
if tVar:
elif tOther:
else:
```

```
repeat
for each
char
tChar in
tVar
end
repeat
repeat
10
end
repeat
```

```
repeat
with x =
1 to 10
end
repeat
```

```
repeat
while x >
1
subtract
1 from x
end
repeat
```

```
if true
then ...
else ...
```

```
if tVar
then
else if
tOther
then
else
end if
```

```
switch tVar
case "a"
break
default
break
end switch
```

Operators

Operators are ways of combining values such as boolean values, numbers or strings, to produce other values.

Python

LiveCode

```
// Logical
true and false is false
true or false is true
not false is true
// String
"foo" & "bar" is "foobar"
"foo" && "bar" is "foo bar"
"string" begins with "st"
"string" ends with "g"

// Chunks
char 5 of "string" is "n"
item 3 of "a,b,c" is "c"
word 1 of "hi there"
```

```
# Logical
true and false == false
true or false == true
!false == true
```

String

```
"foo" + "bar" == "foobar"
strs = ['foo','bar']
' '.join(strs) == "foo
bar"
"string".startswith("st")
"string".endswith("g")
```

Chunks

```
"string"[4:5] == "n"


items =
"a,b,c".split(",")
items[2] == "c"


words = "hi
there".split(" ")
words[0] == "hi"


lines = "anb".split("n")
lines[1] == "b"
```

```
lines = "a,b,c".split("n")
items = lines[1].split(",")
items[1][0:1] == "a"
```

String Processing

```
is "hi"
line 2
of "a" &
return
& "b" is
"b"
```

```
// Compound
chunks
char 1 of
item 1 of
line 1 of
"a,b,c" is
"a"
```

These examples show how string values can be manipulated.

Python

```
# General
var = 'a' + var
var = var[1:]
var.replace("_",
"-")
```

Regex

```
found =
re.match('([0-
9])', '1')
num =
tMatch.group(1)
```

```
for line in var:
if re.match(pattern,
line):
filtered.push(line)
var =
'n'.join(filtered)
```

LiveCode

```
// General
put "a"
before tVar
delete char 1
of tVar
replace "_"
with "-" in
tVar
// Regex
matchText("1",
"([0-9])", tN)
is true
tN is 1
```

```
filter lines of
tVar with regex
pattern tPattern
```

Array Processing

These examples show how array values can be manipulated.

Python

LiveCode

```
# Split / combine
var = "a,b,c".split(",")
var[1] is "b"
','.join(var)
var == "a,b,c"
```

Iteration

```
for key in array:

    do something with array[key]
```

Length

```
len(array)
```

```
// Spli
combin
put "a
into tV
split
by ","
tVar[2]
"b"
combin
with ",
tVar is
"a,b,c"
// Ite
repeat
each ke
in tArr
-- Do
someth
with
tArray
end re

repeat
each el
tElemen
tArray
end re
```

```
// Length
the numbe
elements i
```

Sorting

These examples show how to sort items and lists.

Python LiveCode

```
list = [5, 2,
3, 1, 4]
sorted(list)
== [1, 2, 3,
4, 5]
sorted(list,
reverse=True)
== [5, 4, 3,
2, 1]
```

```
data = [(6, 1),
(8, 3), (2, 2)]
sorted(data,
key=itemgetter(2))
== [(6, 1), (2,
2), (8, 3)]
```

```
local
tList
put
"5,2,3,1,4"
into tList
sort items
of tList
ascending
numeric
 -> tList
is
"1,2,3,4,5"
sort items
of tList
descending
numeric
 -> tList
is
"5,4,3,2,1"
```

```
local tData
put
"6,1:8,3:2,2"
into tData
set the
lineDelimiter to
":"
sort lines of
tData ascending
numeric by item
2 of each
-> tData is
"6,1:2,2:8,3"
```

Files & Processes

These examples show how to read from and write to files and processes.

Python LiveCode

```
open(tPath).read()
open(tPath).write("")
```

```
get
url("file:/"
& tPath)
put "" into
url("file:/"
& tPath)
```

```
process =
subprocess.Popen([tProc],
stdout=subprocess.PIPE)
while True:
process.wait()
data =
process.stdout.read(5)
if data:
break
```

```
open process
tProc
read from
process tProc for
5
close process
tProc
```

User Input / Notification

These examples show how to pop up information dialogs, or prompts for user input.

Python

```
dlg =
wx.TextEntryDialog(None,
"What is your name?",
defaultValue=default_value)
dlg.ShowModal()
name = dlg.GetValue()
dlg.Destroy()
```

```
dlg = wx.MessageDialog(None,
"Something", caption, wx.OK)
result = dlg.ShowModal()
dlg.Destroy()
```

LiveCode

```
ask
"What
is
your
name?"
put
it
into
tName
```

```
answer
"Something"
```

Custom Handlers

A custom handler is a function or command that you define yourself.

Python

```
def foo(param):
# return something
#
foo(var)
```

LiveCode

```
function foo pParam
end foo
// get foo(tVar)
```

```
command bar pParam
end bar
// bar 5
```