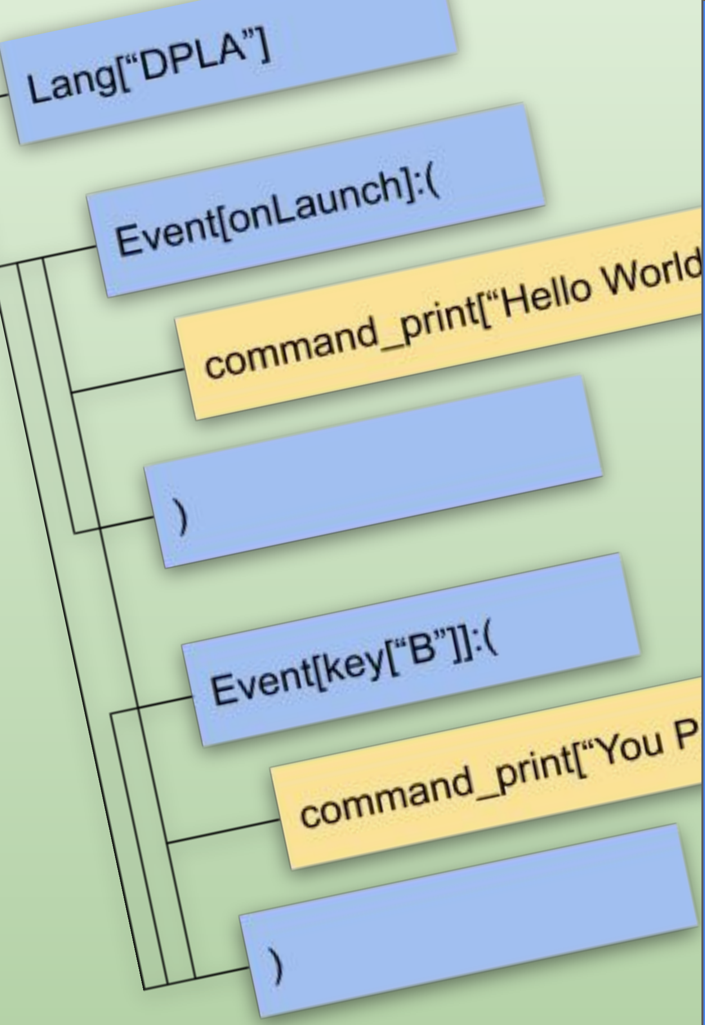
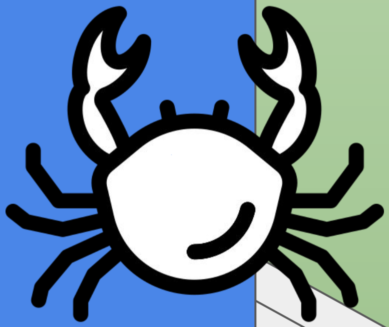


DPLA

Documentation 1.0

```
Lang["DPLA"]
Event[onLaunch]:(
command_print["Hello
World!"]
command_img["/dance.gif"]
)
```



Daniel Bird

```
ng["DPLA"]
Event[onLaunch]:(
Name="bob"
command_speak["Hello" +
name]
command_img["/wave.gif"]
)
```

DPLA/Cope Documentation

Written by Daniel Bird

Introduction to Cope……………………………………………….3
Hello World!……………...…………………………………….3
Maths Operations...……...……………………………………5
Joining Text...…………...………………………………………6
Comments…...…………...……………………………………7
Variables………………………………………………………….8
Joining Variables with Strings…………………………8
Increasing/Changing Variables……….……………………9
If, If else, elif……………..…………………………………10
Functions……………………..………………………………13

Tip!: The page guides are link, click em!

Introduction to Cope

DPLA/Cope is a language based off of python and is designed by Daniel Bird

Hello World!

Let’s make a simple DPLA file. First tell the file we are writing in DPLA by writing **`Lang[“DPLA”]`** . Now write **`Event[onLaunch]:()`**, this will run code when DPLA is launched. In Between the brackets write **`command_print[“Hello world!”]`**, this will print **Hello world**. Now here’s what the file should look like in blue, and the output in green

```
Lang[“DPLA”]
Event[onLaunch]:(
  command_print[“Hello world!”]
)
```

```
Hello World!
```

Check your code if that didn’t print **Hello world!**

So, we learnt how to print **Hello world!** In Cope. Now let’s change the ‘Hello world!’ to something else

```
Lang[“DPLA”]
Event[onLaunch]:(
  command_print[“The blue dog, is red.”]
)
```

```
The blue dog, is red.
```

Let’s make a new line in one line of code using **\n**.

```
Lang[“DPLA”]
Event[onLaunch]:(
  command_print[“Hello\nWorld!”]
)
```

```
Hello
World!
```

Today we learnt how to print words and create a new line in one line of code.

Maths Operations

Since you can print, we can print maths answers. We will need to remove the quotation marks, because we are not inputting strings/text

```
lang[“DPLA”]
event[onLaunch]:(
  command_print[2+2]
  command_print[2*2]
  command_print[2/2]
  command_print[2^2]
  command_print[2%2]
)
```

```
4
4
1
2
0
```

If the code does not print the right results, check the code and make sure it is all correct.

Joining Text

To join text, close off the quotation marks and write **+ `_join_` +** to join text.

```
lang[“DPLA”]
event[onLaunch]:(
    command_print[“Hello” +_join_+ ”World!”]
)
```

```
HelloWorld!
```

See that, there is no space. You will need to add a space in the text.

```
lang[“DPLA”]
event[onLaunch]:(
    command_print[“Hello” +_join_+ ” World!”]
)
```

```
Hello World!
```

Comments

Add **&&** to do a single line comment. Add **#&&#** to do a multiline comment.

```
lang[“DPLA”]
&& Hello, I’m a comment!

#&&#
I’m a multiline comment
Hi!
#&&#
```

User Input

Use **input[“Text:”]** to ask for text, use **input[int[“Number: ”]]** to get numbers

```
lang[“DPLA”]
Event[onLaunch]:(
  command_print[input[int[“Number:”]]]
)
```

```
Number: 1
1
```

Variables

Let’s store variable in Cope, just write the **variable name**, **equal sign** and **value**. IMPORTANT: No spaces int variable name only Letters (ABC), -’s, _’s and no other characters.

```
lang[“DPLA”]
Event[onLaunch]:(
  var=1
  Command_print[Var]
  var_two=”String”
  Command_print[var_two]
)
```

```
1
String
```

Joining Variables with Strings

To join variables you need to close off the quotation marks and write **`+ VarName +`**.

```
lang[“DPLA”]
Event[onLaunch]:(
  Name = input[“Name: ”]
  command_print[“Hello ”+ name +”!”]
)
```

```
Name: Bob
Hello Bob!
```

Increasing/Changing Variables

To increase a variable just add the plus sign and the var name while setting the variable to it.

```
lang[“DPLA”]
Event[onLaunch]:(
  int = 1
  Int = int+2
  command_print[int]
)
```

```
3
```

Or use a related maths symbol.

```
lang[“DPLA”]
Event[onLaunch]:(
  One = 1
  One = 1-1
  Two = 2
  && and so on..
)
```

If, If Else and elif

Let's add if, if else and if elif else statements to make the program make decisions. Add and == to ask if the input equals the other input

```
lang[“DPLA”]
Event[onLaunch]:(
  If [1 == 1]:(
    Command_print[1]
  )
)
```

```
1
```

Do =! To ask if one input does not equal another

```
lang[“DPLA”]
Event[onLaunch]:(
  If [1 =! 0]:(
    Command_print[1]
  )
)
```

```

```

In case this will print nothing but do you want it to print zero if it is not ture? Then add en else to it.

```
Event[onLaunch]:(
  If [1 =! 0]:(
    Command_print[1]
  )
  Else:(
    Command_print[0]
  )
)
```

```
0
```

Let’s and an elif to make sure it does equal anything else.

```
Event[onLaunch]:(
  If [1 =! 0]:(
    Command_print[1]
  )
  Elif[1 == 1]:(
    Command_print[2]
  )
  Else:(
    Command_print[0]
  )
)
```

```
2
```

Add as many Elifs as you want to!

Functions

Let’s make functions for our code by using functions:’s.

```
lang[“DPLA”]
functions:hi[]:(
  command _print[“hello”]
)
Event[onLaunch]:(
  hi[]
)
```

```
hello
```

Let’s add parameters to make your function more customizable.

```
lang[“DPLA”]
functions:say[text]:(
  command _print[text]
)
Event[onLaunch]:(
  say[“HELLO”]
)
```

```
HELLO
```