PIFFIN

Python Cheatsheet

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Important data types

- An int (short for integer) stores an integer number, eg. 5
- A float (short for floating point) stores a non-integer eg. 5.111
- A string stores some text eg. “hello”

Operators

Basic mathematical operators:

```
a + b
a - b
a * b
a / c
```

Conversions

Converting from one data type to another:

```
#converts a string to a float
float("1.54")

#converts a string to an int
int("3")

#converts a float or an int to a string
str(2.5)
```

Variables

We can store some data in a variable using the = operator:

```
a = 10
b = "my stuff"
```

Then we can recall them by using their names:

```
print(a+5)
print(b)
```

And also update them:

```
a = a + 5
b = "hands off " + b
```

User Input

We can ask the user for a string (and store it in a variable called `answer`):

```
answer = raw_input("what is your name?")
```

Conditionals

This is how we write a program that can respond differently depending on the conditions in which it runs:

```
time = 12
if time < 12:
    print("morning!")
elif time >= 12 and time < 18:
    print("afternoon!")
else:
    print("evening!")
```

We have these to choose from:

- == exactly the same as. 2 equals signs for comparison, 1 for variable assignment.
- >= more than or equal to
- > more than
- <= less than or equal to
- < less than
- != not equal to

Loops

To loop forever:

```
while True:
    print("hello!!")
```

To loop a certain number of times we can use `while`. `while` will only loop the code after the `:` while its condition is True:

```
#make a variable to keep count
loops = 0

#keep running the code while the loops variable is less than 10
while loops < 10:
    print(loops)

    #increase the loops variable by 1
    loops = loops + 1
```

Libraries

We can use libraries to get extra functionality in our programs. For example, to sleep for some time:

```
import time
time.sleep(5)
```

Or to get a random number between 1 and 10:

```
import random
random.randint(1,10)
```

Functions

If we are copying and pasting the same code over and over, we can use a function to save time and improve readability. In this example the function is called `my_func` and it needs 2 arguments (arg1, and arg2).

- When we call the function we need to provide the right number of arguments
- We can only call a function after it's been defined with the `def` keyword

```
#define the function
def my_func(arg1,arg2):
    print(arg1 * arg2)

#call it
my_func(10,100)
```

Raspberry Pi GPIOs

Import the library, set the pin numbering to board mode and turn off distracting warnings:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
```

Outputs

```
#store the pin number in a variable
led_pin = 8

#set the GPIO up to be an output
GPIO.setup(led_pin, GPIO.OUT)

#turn on
GPIO.output(led_pin, True)
#turn off
GPIO.output(led_pin, False)
```

Inputs

```
#store the pin number in a variable
button_pin = 16

#set the GPIO to be an input that's high normally, low when pressed
GPIO.setup(button_pin,GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    #check if it's pressed
    if GPIO.input(button_pin) == False:
        print("button pressed")
```