Cheatography

Python Basics Cheat Sheet

by davidhemer via cheatography.com/132946/cs/26989/

Data types

Strings	Represent text	`"Hello"`
Integers	Represent whole numbers	`3`
Floating point numbers	Represent decimal values	`3.258`
Scientific notation	Used to represent very large or very small numbers	`3.798e17`

Mathematical operators

+	addition	`3+2`
-	subtraction	`72.2-83.7`
*	multiplication	`42.8*21.3`
/	division	`22.2/3.78`
//	integer division	`25//12`
%	modulus (remainder)	`35%2`

Comparing numbers

`==`	True if two numbers are equal
`!=`	True if two numbers are not equal
`<=`	True if the first number is less than or equal to the second
`<`	True if the first number is strictly less than the second
`>=`	True if the first number is greater than or equal to the second
`>`	True if the first number is strictly greater than the second

Importing modules

```
import math
from sigfig import *
from sigfig import round
```

The first import statement requires all calls to functions from this module to be prefixed with the module name, e.g. `math.gcd(4,8)`
The second import statement allows functions from the sigfig module to be called without prefixes. Name clashes can provide unexpected results.
The third import statement allows us to select the module functions that we want to use.

Math module functions

square root	math.sqrt	`math.sqrt{25.2}`
greatest common divisor	math.gcd	`math.gcd(25,8)`
power	math.pow	`math.pow(3,5)`
pi	math.pi	`math.pi`

sigfig module

```
from sigfig import *
# round to a given number of decimal places
round(number, decimals=3)
# round to the given number of significant figures
round(number, sigfigs=4)
```

Variables

```
import math
# assign values to variables
r=8.2
c=2*math.pi*r
print(c)
# update the value of a variable
r=r+5
c=2*math.pi*r
print(c)
```

Strings

`"Fred"`	`'Fred'`	`"The cat sat on the mat"`
`'The answer is 3.24'`	`"$5-6.85"`	`"Beware of the llama!"`
`'###IMPORTANT###'`	`"46*-*2"`	`"()$@!\#"`

Strings are enclosed in double quotes or single quotes.

Special characters in strings

`\n`	New line
`\t`	Horizontal tab
`\\`	Backslash
`\'`	Single quote
`\"`	Double quote

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Python Basics Cheat Sheet

by davidhemer via cheatography.com/132946/cs/26989/

Print statements

```
print("Hello World")
print("\"Hello World\"") # Enclosed in quotes
print("Hello\nWorld") # New line between words
print("Hello\tWorld") # Tab between words
```

Formatting strings

```
name="Clarence"
print("Hello my name is {}".format(name))
pet="cat"
place="mat"
print("The {0} sat on the {1}. The {0} is very lazy".format(pet, place))
```

Replacement fields are replaced by the corresponding string from the list of arguments in the format function.
The indexes start at 0, so {0} is replaced by the value of the first argument.

Format specifiers

d	`{0:d}`	Represents a number as a whole number
f	`{0:.2f}`	Represents a number with a fixed number of decimal places
e	`{0:.3e}`	Represents a number using scientific notation.
%	`{0:.1%}`	Represent a decimal as a percentage with the specified number of decimal places
b	`{0:b}`	Represent a number in binary form

Format specifiers are used in print statements to describe how a number is represented.

Formatting numbers in strings

```
aNum=3234.374
print("{0} with 0 dps is {0:.0f}".format(aNum))
print("{0} with 1 dp is {0:.1f}".format(aNum))
print("{0} with 2 dps is {0:.2f}".format(aNum))
print("{0} in scientific notation is {0:e}".format(aNum))
print("{0} in scientific notation with one decimal place is {0:.1e}".format(aNum))
print("{0} in scientific notation with two decimal places is {0:.2e}".format(aNum))
```

Formatting percentages

```
student1="Beryl"
score1=0.828
student2="Marvin"
score2=0.738
print("{0} got {1:.0%} in her Maths test".format(student1, score1))
print("{0} got {1:.1%} in his Maths test".format(student2, score2))
```

The format specifier `{0:.2%}` formats the first argument in the format list as a percentage with two decimal places.
Note that the number being formatted should be represented as a decimal.

Input statements

```
name=input("Enter your name: ")
favFood=input("Enter your favourite food: ")
print("Hi {0}, seems like your favourite food is {1}".format(name, favFood))
```

Input statements store the resulting value as a string. These strings can then be used in a print statement.

Converting strings

int	Converts a string to an integer	`int("32")`
float	Converts a string to a floating point number	`float("56.78")`

String conversions can be used with input statements, where values are always entered as strings.

Conditional statements

```
age=int(input("How old are you? "))
if (age >= 18):
    print("Don't forgot to vote.")
```

Check whether a person is 18 years. Remember to convert the age into an integer before doing the check.

Sponsored by CrosswordCheats.com
Learn to solve cryptic crosswords!
http://crosswordcheats.com

If then else statement

```
import random
target=random.randint(1,10)
guess=int(input("Enter your guess between 1 and
10: " ))
if (guess==target):
    print("Correct! Well done.")
else:
    print("Wrong. The correct answer was {:d}.".f-
ormat(target))
print("Thanks for playing.")
```

A lucky number guessing game. Uses the random module to generate a random number between 1 and 10.

Else if statements

```
import random
age=random.randint(1,99)
print("Age is {:d}.".format(int(age)))
if (age <= 12):
    print("You are a child, tickets cost $10.50")
elif (age <60):
    print("You are an adult, tickets cost $14.00")
else:
    print("You are a senior, tickets cost $12.00")
```

Random number generator used to generate a random age between 1 and 99.
Elif branch used to include an additional case.
Multiple

While loops

```
import random
target=random.randint(1,10)
guess=input("Enter your guess between 1 and 10: " )
num_guesses=1
while (int(guess) != target):
    print("That was wrong. Try again.")
    guess=input("Enter your guess between 1 and 10:
" )
    num_guesses=num_guesses+1
print("Correct! Well done. You took {0:d} guesse-
s.".format(num_guesses))
```

While loops (cont)

```
print("Thanks for playing.")
```

The while loop repeats until the condition is no longer true.
In this case the while loop repeats while the guess is not correct.

For loops

```
print("The first ten perfect square numbers are:")
for i in range(1,11):
    print("{:d}".format(i**2))
```

This loop is repeated 10 times. The values of the loop variable i range from 1 up to, but not including, 11.