Python Cheat Sheet

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double-quotes.

Hello World

```
print("Hello, World!")
```

Hello world with a variable

```
output = "Hello, World!"
print(output)
```

f-string (using variables in strings)

```
firstName = 'Albert'
lastName = 'Einstein'
fullName = f"{firstName} {lastName}"
```

Date types

Data types are the way that data is stored in memory.

Numbers

```
int      #stores an Integer value i.e. 8
float    #stores a Floating-point value i.e. 3.14
```

Letters

```
string   #stores a set of characters
```

Boolean

```
bool     #stores either True or False
```

Operators

Way to perform numeric functions

Operator	Symbol in Python
Addition	+
Subtraction	-
Multiply	*
Divide	/
Power	**
Mod (Remainder)	%
Integer Divide	//

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

Equals	x == 42
Not equal	x != 42
Greater than	x > 42
Greater than or equal to	x >= 42
Less than	x < 42
Less than or equal to	x <=42

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning Boolean values

```
gameActive = True
canEdit = False
```

A simple if statement

```
if age >= 18:
    print("You can vote!")
```

If-else test statements

```
if age >= 18:
    print("You can vote!")
else:
    print("You can NOT vote!")
```

The if-elif-else chain

```
if age < 4:
    price = 0
elif age < 18:
    price = 5
else:
    price = 10
```

Check range

```
age = 16
if age >=15 and age <=65:
    print("You can work!")
```

While loops

If statements are used to test for particular conditions and respond appropriately.

Counting to 5

```
currentNumber = 1

while currentNumber <= 5:
    print(currentNumber)
    currentNumber += 1
```

Using a flag

```
active = True
while active:
    message = input("Do you want to continue?")

    if message == 'No':
        active = False
```

Using break

```
while True:
    city = input("Name a city to travel to?")

    if city == 'quit':
        break
    else:
        print("Your destination is:", city)
```

Using continue

```
bannedUsers = ['Eve', 'Fred', 'Gary', 'Helen']

players = []
while True:
    player = input(prompt)
    if player == 'quit':
        break
    elif player in bannedUsers:
        continue
    else:
        players.append(player)
```

For loops

Cycle through a list of values until they have all been inspected.

Counting to 5

```
for x in range (1,6):
    print(x)
```

for loop with list of strings

```
animals = ['cat', 'dog', 'horse']

for x in animals:
    print(x)
```

Standard for loop

```
for x in range(10):
    print(x)
```

Functions

Functions are named blocks of code, designed to do one specific job. Information to a function is called an argument, and data received by a function is called a parameter.

A simple function

```
def greetUser():
    print("Hello!")

greetUser()
```

Passing an argument

```
def greetUser(username):
    print("Hello!", username)

greetUser('Steve')
```

Returning a value

```
def addNumbers(x, y):
    return x + y

sum = addNumbers(3, 5)
print(sum)
```

Classes

A class defines the behaviour of an object and the kind of information an object can store. The information in a class is stored in attributes, and functions that belong to a class are called methods. A child class inherits the attributes and methods from its parent class.

Creating a dog class

```
Class Dog():
    """Represent a dog."""

    def __init__(self, name):
        """Initialize dog object."""
        self.name = name

    def sit(self):
        """Simulate sitting"""
        print(f"{self.name} is sitting.")

myDog = Dog('Fluffy')

print(f"{myDog.name} is a great dog!")
myDog.sit()
```

Inheritance

```
Class SearchDog(Dog):
    """Represent a Search dog."""

    def __init__(self, name):
        """Initialize Search dog object."""
        super().__init__(name)

    def search(self):
        """Simulate searching"""
        print(f"{self.name} is searching.")

mySearchDog = Dog('Tank)

print(f"{myDog.name} is a great dog!")
mySearchDog.sit()
mySearchDog.search()
```

Working with files

Your programs can read from files and write to files. Files are opened in read mode('r') by default but can also be opened in write mode ('w') and append mode ('a').

Reading a file and storing its lines

```
filename = 'LOTR.txt'
with open(filename) as fileObject:
    lines = fileObject.readlines()

for line in lines:
    print(line)
```

Writing to a file

```
filename = 'journal.txt'
with open(filename, 'w') as fileObject:
    fileObject.write("I love programming.")
```

Appending to a file

```
filename = 'journal.txt'
with open(filename, 'a') as fileObject:
    fileObject.write("\n I love making games.")
```

Exceptions

Exceptions help you respond appropriately to errors that are likely to occur. You can place code that might cause an error in the try block. Code that should run in response to an error goes in the except block. Code that should run only if the try block was successful goes in the else block

Catching an exception

```
numTickets = input("How many tickets?")
try:
    numTickets = int(numTickets)
except ValueError:
    print("Please try again.")
else:
    print("Your tickets are printing.")
```

Zen of Python

Simple is better than complex

If you have a choice between a simple and a complex solution, and both work, use the simple solution!