SCA
Student Computing Association

Py Reference

PYTHON CHEAT SHEET

presented by the Student Computing Association

VARIABLES & STRINGS

Variables store values. A string is a series of characters in single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

f-strings (using variables in strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = f"{first_name} {last_name}"
print(full_name)
```

LISTS

A list stores a series of items in order. Access items using an index or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item

```
first_bike = bikes[0]
```

Get the last item

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada',
'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

TUPLES

Tuples are similar to lists, but items can't be modified.

Making a tuple

```
dimensions = (1920, 1080)
```

IF STATEMENTS

If statements test for particular conditions and respond appropriately.

Conditional tests

```
equals           x == 42
not equal        x != 42
greater than     x > 42
or equal to      x >= 42
less than        x < 42
or equal to      x <= 42
```

Conditional test with lists

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

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

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

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

USER INPUT

Programs can prompt the user for input. All input is stored as a string.

Prompting for a value

```
name = input("What's your name? ")
print(f"Hello, {name}!")
```

Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("Value of pi? ")
pi = float(pi)
```

DICTIONARIES

Dictionaries store connections between pieces of information as key-value pairs.

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print(f"Color is {alien['color']}")
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav = {'eric': 17, 'ever': 4}
for name, number in fav.items():
    print(f"{name} loves {number}")
```

Looping through all keys

```
for name in fav.keys():
    print(f"{name} loves a number")
```

Looping through all values

```
for number in fav.values():
    print(f"{number} is a favorite")
```

FUNCTIONS

Defining a function

```
def greet(name):
    print(f"Hello, {name}!")

greet('alice')
```

Default values

```
def greet(name, msg='Hello'):
    print(f"{msg}, {name}!")
```

Returning a value

```
def add(a, b):
    return a + b

result = add(3, 5)
```

CLASSES

Defining a class

```
class Dog:
    def __init__(self, name):
        self.name = name
    def sit(self):
        print(f"{self.name} is
sitting.")
```

Creating an instance

```
my_dog = Dog('Buddy')
my_dog.sit()
```