Python CheatSheet

Basics

Basic syntax from the Python programming language

Showing Output To User

The `print` function is used to display or print output as follows:

```
print("Content that you wanna print on screen")
```

We can display the content present in an object using the `print` function as follows:

```
var1 = "Shruti"
print("Hi my name is: ", var1)
```

You can also use **f-strings** for cleaner output formatting:

```
name = "Shruti"
print(f"Hi my name is: {name}")
```

Taking Input From the User

The `input` function is used to take input as a string from the user:

```
var1 = input("Enter your name: ")
print("My name is: ", var1)
```

Typecasting allows us to convert input into other data types:

Integer input:

```
var1 = int(input("Enter the integer value: "))
print(var1)
```

Float input:

```
var1 = float(input("Enter the float value: "))
print(var1)
```

`range` Function

The `range` function returns a sequence of numbers, starting from `start`, up to but not including `stop`, with a default step of 1:

```
range(start, stop, step)
```

Example - display all even numbers between 1 to 100:

```
for i in range(0, 101, 2):
    print(i)
```

Comments

Single Line Comment

```
# This is a single line comment
```

Multi-line Comment (Docstring Style)

```
"""This is a
multi-line
comment"""
```

Escape Sequences

Common escape sequences:

- `\n` → Newline
- `\t` → Tab space
- `\\` → Backslash
- `\'` → Single quote
- `\"` → Double quote
- `\r` → Carriage return
- `\b` → Backspace

Example:

```
print("Hello\nWorld")
```

Strings

Creation

```
variable_name = "String Data"
```

Indexing & Slicing

```
str = "Shruti"
print(str[0])     # S
```

```
print(str[1:4])   # hru
print(str[::-1])  # reverse string
```

Useful String Methods

- `isalnum()` → Check alphanumeric
- `isalpha()` → Check alphabetic
- `isdigit()` → Check digits
- `islower()`, `isupper()` → Check case
- `isspace()` → Check for whitespace
- `lower()`, `upper()` → Convert case
- `strip()`, `lstrip()`, `rstrip()` → Remove spaces
- `startswith()`, `endswith()` → Check prefixes/suffixes
- `replace(old, new)` → Replace substring
- `split(delimiter)` → Split string
- `join(iterable)` → Join elements into string

Example:

```
name = " Shruti "
print(name.strip())
```

Lists

Creation

```
my_list = [1, 2, 3, "hello"]
```

Operations

```
my_list.append(5)
my_list.insert(1, "new")
my_list.remove("hello")
```

```
item = my_list.pop()  # removes last element
my_list.sort()
my_list.reverse()
```

List Comprehension

```
squares = [x**2 for x in range(10)]
```

Tuples

Immutable, ordered collection:

```
my_tuple = (1, 2, 3)
print(my_tuple.count(2))
print(my_tuple.index(3))
```

Sets

Unordered, unique elements:

```
my_set = {1, 2, 3}
my_set.add(4)
my_set.remove(2)
my_set.union({5, 6})
```

Other useful set methods: `intersection()`, `difference()`, `symmetric_difference()`

Dictionaries

Key-value pairs:

```
mydict = {"name": "Shruti", "age": 20}
print(mydict["name"])
mydict["age"] = 21
mydict.update({"city": "Delhi"})
```

Useful methods: `keys()`, `values()`, `items()`, `get()`, `pop(key)`, `clear()`

Indentation

Python uses indentation (usually 4 spaces) to define blocks.

Conditional Statements

```
if x > 0:
    print("Positive")
elif x < 0:
    print("Negative")
else:
    print("Zero")
```

Loops

For Loop

```
for i in range(5):
    print(i)
```

While Loop

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Loop Control

- `break` → exits loop
- `continue` → skips iteration
- `pass` → does nothing (placeholder)

Functions

```
def greet(name):
    return f"Hello {name}"

print(greet("Shruti"))
```

Supports default arguments, keyword arguments, `*args`, and `**kwargs`.

File Handling

```
with open("file.txt", "w") as f:
    f.write("Hello")
```

Modes: `r`, `w`, `a`, `r+`, `w+`, `a+`

Read methods: `read()`, `readline()`, `readlines()`

Exception Handling

```
try:
    x = 10 / 0
except ZeroDivisionError as e:
    print("Error:", e)
else:
    print("No error")
finally:
    print("Always runs")
```

Object Oriented Programming (OOP)

```
class Person:
    def __init__(self, name):
        self.name = name
    def greet(self):
        print(f"Hello, I am {self.name}")

p = Person("Shruti")
p.greet()
```

Supports inheritance, polymorphism, encapsulation, and abstraction.

Useful Built-in Functions

- `len()`, `type()`, `id()`, `dir()`, `help()`
- `sum()`, `max()`, `min()`, `sorted()`
- `enumerate()`, `zip()`, `map()`, `filter()`, `any()`, `all()`

Modules & Imports

```
import math
print(math.sqrt(16))

from datetime import datetime
print(datetime.now())
```

Virtual Environments (Best Practice)

```
python -m venv env
source env/bin/activate  # Linux/Mac
env\Scripts\activate     # Windows
```