Python Cheatsheet

www.appmillers.com

Elshad Karimov

www.appmillers.com

1. Data Types and Variables

```python
# Assigning different data types to variables

int_num = 42               # Integer

float_num = 3.14           # Float

string_var = "Hello, Python!"  # String

bool_var = True            # Boolean
```

2. Strings and String Manipulation

```
# Working with strings and applying various methods

str_var = "Hello, World!”

upper_str = str_var.upper()  # Converts string to uppercase

split_str = str_var.split(',')  # Splits string into list at commas

formatted_str = f"{int_num} is an integer"  # Formats string using f-string
```

3. Lists

```
# Creating and manipulating lists

my_list = [1, 2, 3, “Python"]

first_element = my_list[0]  # Accessing the first element

list_slice = my_list[1:3]  # Slicing list to get a sublist

my_list.append("New Element")  # Adding a new element to the list
```

4. Tuples

```
# Tuples are immutable sequences

my_tuple = (1, 2, 3, “Tuple")

first_element_of_tuple = my_tuple[0]  # Accessing elements

a, b, c, d = my_tuple  # Unpacking the tuple into variables
```

5. Dictionaries

```python
# Dictionaries hold key-value pairs

my_dict = {'name': 'John', 'age': 25}

name = my_dict['name']  # Accessing value by key

my_dict['country'] = 'USA'  # Adding a new key-value pair

for key, value in my_dict.items():  # Iterating over key-value pairs
    print(f"{key}: {value}")
```

6. Control Flow

```
# Using if-else statements and loops
if int_num > 10:
    print("Greater than 10")
elif int_num == 10:
    print("Equal to 10")
else:
    print("Less than 10")

for element in my_list:  # Looping over a list
    print(element)

count = 5
while count > 0:  # Looping with a while statement
    print(count)
    count -= 1
```

7. Functions

```python
# Defining and calling a function

def greet(name="User"):
    return f"Hello, {name}!”

print(greet("John"))  # Function call with an argument
```

www.appmillers.com

8. Classes and Objects

```python
# Defining a class and creating an instance

class Dog:
    def __init__(self, name):
        self.name = name  # Constructor for initializing the object

    def bark(self):
        print("Woof!")

my_dog = Dog("Buddy")  # Creating an instance of Dog
my_dog.bark()  # Calling a method on the Dog object
```

9. Modules and Libraries

```python
# Importing and using functions from modules

import math

from datetime import datetime

result = math.sqrt(25)  # Using the sqrt function from the math module

current_time = datetime.now()  # Getting the current time
```

10. File Handling

```
# Reading and writing files

with open("file.txt", "r") as file:  # Opening a file to read
    content = file.read()

with open("new_file.txt", "w") as new_file:  # Opening a file to write
    new_file.write("Hello, Python!")
```

11. Exception Handling

```python
# Handling exceptions using try-except

try:
    result = 10 / 0  # Code that might raise an exception
except ZeroDivisionError:
    print("Cannot divide by zero!")  # Handling a specific exception
finally:
    print("Execution completed.")  # This block is always executed
```

12. List Comprehensions

```python
# Creating a new list by applying an expression to each item in the list
squares = [x**2 for x in range(5)]  # List of squares of numbers 0-4
```

www.appmillers.com

13. Regular Expressions

```python
# Using regular expressions to find patterns in text

import re

pattern = r'\d+'  # Regular expression pattern for one or more digits

result = re.findall(pattern, "There are 42 apples and 123 oranges.")
```

14. Working with JSON

```python
# Converting between Python objects and JSON

import json

python_obj = {'name': 'John', 'age': 25}

json_data = json.dumps(python_obj)  # Python object to JSON string

new_python_obj = json.loads(json_data)  # JSON string to Python object
```

15. Concurrency and Threading

```
# Using threading for concurrent execution

import threading

def print_numbers():
    for i in range(5):
        print(i)

thread = threading.Thread(target=print_numbers)  # Creating a thread

thread.start()  # Starting the thread
```

www.appmillers.com

16. Working with Dates and Times

```python
# Handling dates and times

from datetime import datetime, timedelta

current_date = datetime.now()  # Current date and time

future_date = current_date + timedelta(days=7)  # Date 7 days from now

formatted_date = current_date.strftime('%Y-%m-%d %H:%M:%S')  # Formatting date
```

17. Advanced Topics - Decorators

```python
# Enhancing function behavior with decorators
def decorator(func):
    def wrapper():
        print("Before function execution")
        func()
        print("After function execution")
    return wrapper

@decorator
def say_hello():
    print("Hello, World!")

say_hello()  # Wrapped function call
```

18. Advanced Topics - Generators

```python
# Creating a generator function
def count_down(num):
    while num > 0:
        yield num  # Yielding values one by one
        num -= 1

for count in count_down(5):  # Using the generator
    print(count)
```

www.appmillers.com

19. Debugging and Testing

```python
# Writing unit tests for a function
import unittest

def add(x, y):
    return x + y

class TestAddition(unittest.TestCase):
    def test_add_positive_numbers(self):
        self.assertEqual(add(2, 3), 5)

if __name__ == '__main__':
    unittest.main()  # Running the tests
```

20. Web Development - Flask

```python
# Creating a basic web application using Flask
from flask import Flask
app = Flask(__name__)

@app.route('/')  # Route for the home page
def home():
    return "Hello, Flask!"

if __name__ == '__main__':
    app.run(debug=True)  # Running the Flask app
```

21. Database Interaction - SQLite

```python
# Basic SQLite database interaction

import sqlite3

conn = sqlite3.connect('example.db')  # Connecting to the SQLite database

cursor = conn.cursor()

cursor.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)')  # Creating a
table

conn.commit()  # Committing the transaction

conn.close()  # Closing the connection
```