coursera

Python Basics Cheat Sheet

Getting Started with Basics

Hello world:

- Prints a message to the screen.

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

Arithmetic:

- Performs basic arithmetic operations.

```python
# Addition
result = 2 + 3
```

Data types:

- Defines the type of values.

```python
num = 10   # Integer
text = "Python"   # String
```

File handling:

- Opens, reads, writes, and closes files.

```python
with open('file.txt', 'r') as file:
    content = file.read()
```

Functions:

- Defines reusable blocks of code.

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

Strings:

- Manages and manipulates text data.

```python
s = "Python"
print(s.upper())
```

If else:

- Implements conditional logic.

```python
if x > 0:
    print("Positive")
else:
    print("Non-positive")
```

Loops:

- Repeats a block of code multiple times.

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

Lists:

- Stores an ordered collection of items.

```python
fruits = ["apple", "banana", "cherry"]
```

Variables:

- Stores values that can be reused and manipulated.

```python
age = 25
```

f-Strings:

- Formats strings using expressions inside curly braces.

```python
name = "Alice"
print(f"Hello, {name}")
```

Plus-equals:

- Adds a value to a variable and assigns the result to that variable.

```python
count = 0
count += 1
```

Python Built-in Data Types

Strings:

- Text data enclosed in single or double quotes.

```python
s = "Hello"
```

Lists:

- Ordered and mutable sequence of elements.

```python
lst = [1, 2, 3]
```

Dictionary:

- Key-value pairs enclosed in curly braces.

```python
d = {"one": 1, "two": 2}
```

Booleans:

- Represents True or False values.

```python
is_active = True
```

Set:

- Unordered collection of unique elements.

```python
s = {1, 2, 3}
```

Tuple:

- Ordered and immutable sequence of elements.

```python
t = (1, 2, 3)
```

Numbers:

- Numeric data types like int, float, and complex.

```python
num = 10   # Integer
float_num = 10.5   # Float
```

Casting:

- Converts a variable from one type to another.

```python
x = int(3.8)   # 3
```

Python Modules

Import modules:

- Imports a module to use its functions and attributes.

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

Shorten module:

- Uses an alias for a module.

```python
import numpy as np
```

From a module import all:

- Imports all functions and attributes from a module.

```python
from math import *
```

Functions and attributes:

- Uses specific functions and attributes from a module.

```python
from math import pow, pi
print(pow(2, 3))
print(pi)
```

Python Strings

Concatenates:

- Combines two or more strings.

```python
full = "Hello" + " " + "World"
```

Slicing string:

- Extracts part of a string.

```python
substr = "Python"[2:5]   # 'tho'
```

Multiple copies:

- Repeats a string multiple times.

```python
repeat = "Hi! " * 3   # 'Hi! Hi! Hi! '
```

Formatting:

- Formats strings using {} placeholders.

```python
s = "Hello, {}. Welcome to {}.".format(name, place)
```

Input:

- Takes user input as a string.

```python
user_input = input("Enter something: ")
```

Join:

- Joins elements of an iterable into a single string.

```python
join_str = ", ".join(["apple", "banana", "cherry"])
```

Endswith:

- Checks if a string ends with a specific suffix.

```python
ends = "python.org".endswit
```

Looping:

- Iterates through characters of a string.

```python
for char in "Python":
    print(char)
```

Array-like:

- Access characters using indexes.

```python
s = "Python"
print(s[0])   # 'P'
```

String length:

- Gets the length of a string.

```python
length = len("Python")
```

Check string:

- Checks if a substring exists within a string.

```python
if "Py" in "Python":
    print("Found!")
```

Python Lists Commands

Generate:

- Creates a list with specified elements.

```python
fruits = ["apple", "banana", "cherry"]
```

List slicing:

- Extracts parts of a list.

```python
sublist = fruits[1:3]   # ['banana', 'cherry']
```

Omitting index:

- Omits starting or ending index for slicing.

```python
sublist_from_start = fruits[:2]   # ['apple', 'banana']
sublist_to_end = fruits[1:]   # ['banana', 'cherry']
```

With a stride:

- Extracts elements with step size.

```python
stride_slicing = fruits[::2]   # ['apple
```

Count:

- Counts occurrences of an element in a list.

```python
count = fruits.count("apple")
```

Repeating:

- Repeats elements in a list.

```python
repeat_list = [0] * 5   # [0, 0, 0, 0, 0]
```

Sort & reverse:

- Sorts and reverses a list.

```python
fruits.sort()   # ['apple', 'banana', 'cherry']
fruits.reverse()   # ['cherry', 'banana', 'apple']
```

Access:

- Accesses elements by index.

```python
first_fruit = fruits[0]   # 'apple'
```

Concatenating:

- Combines two or more lists.

```python
combined = fruits + ["kiwi", "mango"]
```

Python Loops

Basic:

- Repeats a block of code for a fixed number of times.

```python
for i in range(3):
    print(i)
```

Break:

- Exits the loop prematurely.

```python
for i in range(5):
    if i == 3:
        break
    print(i)
```

With zip():

- Iterates over multiple iterables in parallel.

```python
names = ["Alice", "Bob"]
scores = [85, 92]
for name, score in zip(names, scores):
    print(name, score)
```

While:

- Continues until a condition is met.

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

Python Loops (continued)

Range:

- Generates a sequence of numbers.

```python
for i in range(1, 10, 2):
    print(i)
```

Continue:

- Skips the current iteration and proceeds to the next.

```python
for i in range(5):
    if i == 3:
        continue
    print(i)
```

With index:

- Uses range and len to loop with an index.

```python
fruits = ["apple", "banana", "cherry"]
for i in range(len(fruits)):
    print(fruits[i])
```

for/else:

- Executes else block if no break occurs.

```python
for i in range(5):
    if i == 6:
        break
else:
    print("Completed")
```

Python Functions

Basic:

- Defines a function with a specific task.

```python
def greet():
    print("Hello, World!")
```

Keyword arguments:

- Passes arguments by name.

```python
def introduce(name, age):
    print(f"Name: {name}, Age: {age}")
introduce(age=30, name="Alice")
```

Anonymous functions:

- Uses lambda for short, unnamed functions.

```python
square = lambda x: x ** 2
print(square(5))
```

Return:

- Returns a value from a function.

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

Returning multiples:

- Returns multiple values from a function.

```python
def get_name_age():
    return "Alice", 25
name, age = get_name_age()
```

Default value:

- Uses default values for parameters.

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

Positional arguments:

- Passes arguments in the order of parameters.

```python
def add(a, b):
    print(a + b)
add(3, 5)
```

Python Classes

User-defined exceptions:

- Creates custom exception classes.

```python
class CustomError(Exception):
    pass
```

repr() method:

- Provides a string representation of an object.

```python
class Dog:
    def __repr__(self):
        return f"Dog({self.name})"
```

Constructors:

- Special methods to initialize new objects.

```python
class Dog:
    def __init__(self, name):
        self.name = name
```

Method:

- Defines functions within a class.

```python
class Dog:
    def bark(self):
        print("Woof!")
```

Defining:

- Defines a blueprint for objects.

```python
class Dog:
    def __init__(self, name):
        self.name = name
```

Class variables:

- Variables shared among all instances.

```python
class Dog:
    species = "Canine"
```

Polymorphism:

- Uses a single interface for different types.

```python
class Cat:
    def sound(self):
        return "Meow"
class Dog:
    def sound(self):
        return "Bark"
for pet in (Cat(), Dog()):
    print(pet.sound())
```

Overriding:

- Redefines methods in a subclass.

```python
class Animal:
    def make_sound(self):
        print("Generic sound")
class Dog(Animal):
    def make_sound(self):
        print("Bark")
```

Inheritance:

- Derives a new class from an existing class.

```python
class Animal:
    def __init__(self, name):
        self.name = name
class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed
```