JUNIOR

PYTHON

INTERVIEW CHEATSHEETS

Mario Ruci | Senior Software Engineer

Learn more on
https://devmasters.pro/python

What is Python?

Python is a high-level, interpreted programming language known for its readability and simplicity. It supports multiple programming paradigms and is widely used in web development, data science, artificial intelligence, and more. Python 2 vs. Python 3, and the importance of Python in various industries.

Variables and Data Types

Python has various data types, including int, float, str, list, tuple, dict, etc. Understanding how to declare and use variables is fundamental. Explaining dynamic typing, type() function, and common data type conversion methods.

Variable Declaration

Examples of variable declarations for different data types.

```
# Variable declaration
name = 'John'
age = 25
height = 5.9
is_student = False
```

Dynamic Typing

The type of a variable is determined at runtime. Showing an example of changing the type of a variable.

```
# Dynamic typing
name = 42  # Valid, name can now hold an integer
```

Type Function

Using the type() function to determine the data type of a variable. Illustrating how to check the type of a variable.

```
# Type function
print(type(name))  # Output: <class 'int'>
```

Data Type Conversion

Common methods for converting data types in Python. Examples of converting between int, float, and str.

```
# Common data type conversions
float_age = float(age)
str_height = str(height)
int_string = int('123')
```

For Loop

Iterates through elements in an iterable using the Python for loop.

```python
# Syntax
for variable in iterable:
    # Code block

# Example
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```

While Loop

Executes a code block repeatedly as long as a specified condition is true with the Python while loop.

```python
# Syntax
while condition:
    # Code block

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

Ternary Operator

Provides a concise way to express conditional statements in Python.

```python
# Syntax
result = value_if_true if condition else value_if_false

# Example
x = 10
y = 20
max_value = x if x > y else y
```

range() Function

Generates a sequence of numbers in a specified range with optional start, stop, and step parameters.

```python
# Syntax
range(start, stop, step)

# Example
for i in range(1, 6, 2):
    print(i)
# Output: 1, 3, 5
```

Enumerate

Pairs each element in an iterable with its corresponding index using the Python enumerate() function.

```
# Syntax
for index, value in enumerate(iterable):
    # Code block

# Example
fruits = ['apple', 'banana', 'orange']
for index, fruit in enumerate(fruits):
    print(f'Index: {index}, Value: {fruit}')
```

Lambda Functions

Creates anonymous functions using the lambda keyword for short-lived operations in Python.

```
# Syntax
lambda arguments: expression

# Example
square = lambda x: x**2
result = square(4)
# Output: 16
```

Default Arguments

Assigns default values to function parameters, allowing for optional arguments in Python.

```
# Syntax
def greet(name, greeting='Hello'):
    # Code block

# Example
greet('John')
# Output: Hello, John!
```

Try-Except Block

Catches and handles exceptions in Python, preventing program crashes due to errors.

```
# Syntax
try:
    # Code block with potential error
except ExceptionType as e:
    # Code block to handle the exception
```

Finally Block

Defines a block of code that will be executed regardless of whether an exception occurs or not in Python.

```
# Syntax
try:
    # Code block with potential error
except ExceptionType as e:
    # Code block to handle the exception
finally:
    # Code block to execute always
```

Custom Exceptions

Creates user-defined exception classes to handle specific error conditions in Python.

```
# Syntax
class CustomError(Exception):
    # Code block

# Example
raise CustomError('This is a custom error')
```

Except-Else Block

Executes a block of code if no exceptions are raised in the try block in Python.

```
# Syntax
try:
    # Code block with potential error
except ExceptionType as e:
    # Code block to handle the exception
else:
    # Code block to execute if no exception
```

Handling Multiple Exceptions

Catches and handles multiple types of exceptions in a single except block in Python.

```
# Syntax
try:
    # Code block with potential error
except (ExceptionType1, ExceptionType2) as e:
    # Code block to handle the exceptions
```

DEV MASTERS

WANT MORE?

LIKE | COMMENT | FOLLOW

Mario Ruci | Senior Software Engineer

Learn more on
https://devmasters.pro