Python Cheatsheet

Variables

```
# Examples

my_int = 5
my_float = 2.71828
my_bool_1 = True
my_bool_2 = False
my_string = "This is a string."
my_char = 'C'
# Characters and strings in Python are often treated as the same data type.
```

Basic I/O

```
# Use `input` to get a string.
s = input("Enter a string: ")
# Use `int` or `float` to cast a string input to an integer.
num = int(input("Enter an integer: "))
num2 = float(input("Enter a float: "))
```

Operators

Python Operator Precedence

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

Casting

```python
num = 3.14159
# Print the original `num`, but also casted to an integer.
print("num:", num, int(num))

# Casting to a String is a bit different:
num_string = str(num) # `str` is short for "string".
print("string:", num_string)

# Casting from a String to other data types:
num_string_2 = "123"
num2 = int(num_string_2)
print("num2:", num2)
```

Lists

```python
fruits = ["Apple", "Banana", "Cantaloupe"]

first_fruit = fruits[0] # "Apple"
second_fruit = fruits[1] # "Banana"

fruits_count = len(fruits) # `len` is short for "length".

# Multidimensional
matrix = [[2, 4, 6], [8, 10, 12], [14, 16, 18]]
```

Strings

```python
s = "Hello, world!"
c = s[7] # Get character at index 7.

# Get a substring starting at index 0 and stopping right before index 5.
s2 = s[0:5]

s1 = "Hello, "
s2 = "world!"
s3 = s1 + s2 # ⟶ s3 is "Hello, world!"

str_len = len(s)
```

Basic Escape Sequences

`\"`, `\'`, `\\`, `\n`, `\t`, ...

Functions

```
# Example. See Java cheatsheet for a more explicit outline.

pi = 3.14159

# All Python functions start with `def`.
def area(radius):
    return pi * radius * radius

def circumference(radius):
    return 2 * pi * radius

def output_circle_info(r, a, c):
    print("Circle Info:")
    print(f"Radius: {r}")
    print(f"Area: {a}")
    print(f"Circumference: {c}")

radius1 = 3
radius2 = 9

a1 = area(radius1)
a2 = area(radius2)

c1 = circumference(radius1)
c2 = circumference(radius2)

output_circle_info(radius1, a1, c1)
output_circle_info(radius2, a2, c2)
```

Comparison Operators

- == → equality operator, tests if two values are equal
- The single equals sign = is already used as the assignment operator!
- != → inequality operator, tests if two values are not equal
- < → less than
- > → greater than
- <= → less than or equal to
- >= → greater than or equal to

Logical Operators

- `and` → AND operator
- `or` → OR operator
- `not` → NOT operator

Conditionals

```
if boolean expression:
    // ...
elif boolean expression:
    // ...
else:
    // ...
```

Loops

```
num = 0
while num < 10:
    print(num, end=' ')
    num += 1 # Python does not have the prefix increment operator.
    # However, num += 1 is also the same thing as num = num + 1.
    # The += operator also works in Java.
print()
```

```
# for-loops in Python are weird. You can just pay attention to how to use them.
for i in range(21):
    if i % 2 == 0: # If i is even.
        print(i, end=' ')
print()
```

```
arr = ["Alice", "Bob", "Charlie"]
for s in arr:
    print(s, end=' ')
print()
```

Loop statements: `break`, `continue`

Other Data Structures

```python
# Lists in Python are already variable-length arrays.
# No need to import anything.
nums = [2, 4, 6, 8]

# Get an element at a specific index.
e = nums[2]

# Set an element at a specific index.
nums[1] = 9

# Add an element to the end of the list.
nums.append(5)

# Insert an element at index 3.
nums.insert(3, 6)

# Remove an element at index 1.
nums.pop(1)
```

```python
players = dict()

# Or, create a dictionary with some initial data.
# players = {
#     "Alice": 100,
#     "Bob": 200,
#     "Charlie": 150
# }

# Add / change a key-value pair.
players["Alice"] = 2000
players["Bob"] = 1900
players["Charlie"] = 1950

players["Alice"] = 3000 # Changes the Alice's existing balance.

# Check if the players has a specific key.
hasAlice = "Alice" in players
hasDave = "Dave" in players
print("Contains Alice:", hasAlice)
print("Contains Dave:", hasDave)

# Get a value associated with a key.
print("Alice's Balance: ", players.get("Alice"))
print("Dave's Balance: ", players.get("Dave")) # `None` since Dave is not in the players.
# `None` is the equivalent of `null` in Python.
```

```python
players = dict()

# Or, create a dictionary with some initial data.
# players = {
#     "Alice": 100,
#     "Bob": 200,
#     "Charlie": 150
# }

# Add / change a key-value pair.
players["Alice"] = 2000
players["Bob"] = 1900
players["Charlie"] = 1950

players["Alice"] = 3000 # Changes the Alice's existing balance.

# Check if the players has a specific key.
hasAlice = "Alice" in players
hasDave = "Dave" in players
print("Contains Alice:", hasAlice)
print("Contains Dave:", hasDave)

# Get a value associated with a key.
print("Alice's Balance: ", players.get("Alice"))
print("Dave's Balance: ", players.get("Dave")) # `None` since Dave is not in the players.
# `None` is the equivalent of `null` in Python.
```

Exception Handling

```python
# Start of the `try` block!
try:
    # Something that might throw an exception.
    n = 5 / 0
# End of the `try` block. Start of the `except` (catch) block!
except Exception as e:
    print("A problem occurred!");

    # You can also print the details of the exception.
    print(e)
```

Packages

Refer to slides 140 - 144 on day 5 for how to use packages.