CUHackit Python Workshop

Python-isms, dunders, and list comprehension

Alex Day

Ph.D. Student
School of Computing

January 29, 2022

Overview

1. Introductions
2. List Comprehension
3. Dunder Methods
4. Walrus Operator

Overview

1. **Introductions**
2. List Comprehension
3. Dunder Methods
4. Walrus Operator

Alex Day

- B.S. Computer Science Clarion in Pennsylvania
- Ph.D. Student under Dr. Ioannis Karamouzas
- Working on Social Robot Navigation in the Motion Planning Lab in Charleston
- Programming in Python for the last 7 years

Audience

- What year are you?
- How comfortable are you with programming in general, any language?
- How comfortable are you with python?

Audience

- What year are you?
- How comfortable are you with programming in general, any language?
- How comfortable are you with python?

Audience

- What year are you?
- How comfortable are you with programming in general, any language?
- How comfortable are you with python?

Python

- First released in 1991 by Guido van Rossum (BDFL)
- Python 3 was released in 2008
- Python Enhancement Proposals (PEPs)
- Scripting language popular in machine learning and for rapid prototyping
- Massive repository of third party libraries on PyPi

Python

Example

```
name = input("What is your name?")

if name.lower() == "alex":
    print("Hi!")
else:
    for _ in range(5):
        print("GO AWAY I DON'T KNOW YOU")
```

Input and Output

Example

```
print("Hello world!")
user_input = input("> ")
print(user_input)
```

Conditionals

Example

```
import random

number_to_guess = random.randint(0, 100)
guess = int(input("Please enter your guess... "))

if guess < number_to_guess:
    print("too small")
elif guess > number_to_guess:
    print("too big")
else:
    print("just right")
```

Iteration (While)

Example

```
import random

number_to_guess = random.randint(0, 100)
print(number_to_guess)

while True:
    guess = int(input("Please enter your guess... "))

    if guess < number_to_guess:
        print("too small")
    elif guess > number_to_guess:
        print("too big")
    else:
        print("just right!")
        break
```

Iteration (For)

Example

```
import random

number_to_guess = random.randint(0, 100)
print(number_to_guess)

for i in range(5):
    guess = int(input(f"Please enter your guess ({i}/5)... "))

    if guess < number_to_guess:
        print("too small")
    elif guess > number_to_guess:
        print("too big")
    else:
        print("just right!")
        break
else:
    print("You'll get em next time")
```

Functions

Example

```
def fib(n, a=0, b=1):
    """Return the first n numbers of the Fibonacci sequence

    Keyword arguments:
    n -- length of sequence
    a -- first number
    b -- second number
    """
    fib = []
    for i in range(n):
        fib.append(a)
        a, b = b, a + b

    return fib

print(fib(5))
# >>> [0, 1, 1, 2, 3]
```

Classes

Example

```
from random import choice

class Dog:
    def __init__(self, name: str, breed: str):
        self._name = name # type: str
        self._breed = breed # type: str

    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, name: str):
        self._name = name

    def bark(self):
        print("WOOF")

    @staticmethod
    def make_random_dog() -> "Dog":
        names = ["Frido", "Spot", "Kansas", "Lucky"] # type: List[str]
        breeds = ["German Shepard", "Border Collie", "Golden Doodle", "Labrador"] # type: List[str]
        return Dog(choice(names), choice(breeds))
```

Types?

- Python is duck typed
  - If it walks like a `float` and looks like a `float` then it's probably a `float`

Example

```
word = 'Hello'
word = 7
word = 3.63
word = {}
```

- The lack of type safety has pros and cons
- `typing` and `mypy` aim to make python statically typed

Typing in Python

Example

```
from typing import List

def fib(n: int, a: int=0, b: int=0) -> List[int]:
    """Return the first n numbers of the Fibonacci sequence

    Keyword arguments:
    n — length of sequence
    a — first number
    b — second number
    """
    fib = [] # type: List[int]
    for i in range(n):
        fib.append(a)
        a, b = b, a + b

    return fib

print(fib(5))
#>>> [0, 1, 1, 2, 3]
```

Typing in Python

Example

```
mypy functions_typed.py
# Success: no issues found in 1 source file
```

Overview

1. Introductions
2. List Comprehension
3. Dunder Methods
4. Walrus Operator

Lists

- Lists are reference types, similar to `ArrayLists` in Java or `Vectors` in C++
- Unlike other languages lists do not have a specific type

Example

```
numbers = [1, 2, 3, 4, 5, 6]

# Indexing
print(numbers[1])
>>> 2

# Slicing
print(numbers[1:])
>>> [2, 3, 4, 5, 6]

wacky = [1, 1.5, "one point five", {}]
```

List Comprehension

-

Overview

1. Introductions
2. List Comprehension
3. **Dunder Methods**
4. Walrus Operator

Overview

1. Introductions
2. List Comprehension
3. Dunder Methods
4. **Walrus Operator**

Walrus Operator

1.
2. Explanation
3. Example

Walrus Operator

Example

```
# Simple Assignment
num = 15

print(num)
>>> 15

print(num = 15)
>>> TypeError

# Walrus
print(num := 15)
>>> 15
```

The End