report

June 7, 2023

1 Summary Python Adrien Cardinale

1.1 Data Structure

1.1.1 Scalar

[ ]:
```
int = 42
float = 3.14159
complex = 1 + 2j
string = "Hello World"
boolean = True
status = None
```

We don't have to declare the type of the variable, the type is automatically assigned.

1.1.2 Container

[ ]:
```
List = [1, 2, 3, 4, 5]
print(f"The last element of the list is {List[-1]}")
print(f"All the elements of the list are {List[:-1]}")
print(f"The second and third elements of the list are {list[1:3]}")
print(f"All the elements with a step of 2 are {List[::2]}")
# any and all examples
print(any([i > 3 for i in List]))
print(all([i > 3 for i in List]))
for i in List:
    print(i)
```

[ ]:
```
tuple = (1, 2, 3, 4, 5)
hash = tuple.__hash__()
print(4 in tuple)
# tuple[0] = 0 # Error
```

[ ]:
```
dictionary = {"key1": "value1", "key2": "value2"}
print(f"The dictionary {dictionary.items()}")
print(f"The key {dictionary.keys()}")
print(f"The value {dictionary.values()}")
print(f"The key is hashable {dictionary['key1'].__hash__()}")
```

[ ]:
```
set = {1, 2, 3, 4, 5}
print(f"The set {set}")
set.add(6)
print(f"The set {set}")
```

[ ]:
```
from collections import namedtuple
nTuple = namedtuple("namTuple", ["key1", "key2"])
nTuple = nTuple("value1", "value2")
print(f"The named tuple {nTuple}")
print(f"The named tuple {nTuple.key1}")
```

[ ]:
```
a = 1
b = 2
a, b = b, a
print(f"The value of a is {a} and the value of b is {b}")
```

[ ]:
```
for i, v in enumerate(List):
    print(f"The index is {i} and the value is {v}")

firstnames = ['John', 'Emmet', 'Luke']
lastnames = ['Doe', 'Brown', 'Skywalker']
print(list(zip(firstnames, lastnames)))
print([' '.join(x) for x in list(zip(firstnames, lastnames))])
print(list(map(lambda x: ' '.join(x), zip(firstnames, lastnames))))
print(list(filter(lambda x: x%3, [1,2,3,4,5,6,7])))
```

1.2 Operators of dereferencing

[ ]:
```
def operate(a, b, **kwargs):
        if 'add' in kwargs:
            print(f"{a}+{b}={a+b}")
        if 'sub' in kwargs:
            print(f"{a}-{b}={a-b}")
operate(23,42)
operate(23,42,add=True)
operate(23,42,add=True, sub=True)
```

1.3 Class

[ ]:
```
class MyClass:
    def __init__(self, value):
        """
        This is the initializer of the class
        """
        self.value = value
    def __next__(self):
```

```
        """
        This method return the next value
        """
        return self.value + 1
    def __iter__(self):
        """
        This method return the iterator
        """
        return self
    def __getitem__(self, index):
        """
        This method return the item at the index
        """
        return self.value[index]
    @property # property decorator to make the method a property of the class
↪(getter)
    def value(self):
        """
        This method return the value
        """
        return self._value
    @value.setter # setter decorator to make the method a setter of the class
    def value(self, value):
        """
        This method set the value
        """
        self._value = value

class oneClass(MyClass):
    def __init__(self, value):
        """
        This is the initializer of the class
        """
        super().__init__(value)
```

1.3.1 Singleton

[ ]:
```
# Singleton pattern (single)

## Manière (pas top)
class Singleton(object):
    _instance = None

    def __new__(cls, *args, **kwargs):
        if not cls._instance:
            print('Creating new instance')
            cls._instance = super(Singleton, cls).__new__(cls, *args, **kwargs)
```

```
        else:
            print('Instance already created')
        return cls._instance

## Manière Pythonique
class Singleton(type):
    _instances = {}
    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            cls._instances[cls] = super(Singleton, cls).__call__(*args, **kwargs)
        return cls._instances[cls]

class Logger(object):
    __metaclass__ = Singleton


def foo():
    logger = Logger()
    logger.error('This is an error message')
    logger.info('This is an info message')
    logger.debug('This is a debug message')

def bar():
    logger = Logger()
    logger.error('This is an error message')
```

1.4 Decorator

[35]:
```
import logging
import time

def heig(func):
    def wrapper(*args, **kwargs):
        logging.basicConfig(filename='heig.log', level=logging.INFO)
        logging.info('Running function: {}'.format(func.__name__))
        logging.info('Arguments values were: {}'.format(args))
        start = time.time()
        try:
            logging.info('Function returned: {}'.format(func(*args, **kwargs)))
        except Exception as e:
            logging.error('Error: {}'.format(e))

        end = time.time()
        logging.info('Execution time: {} seconds'.format(end - start))
        logging.info('Execution date and time: {}'.format(time.
 ↪strftime('%Y-%m-%d %H:%M:%S', time.localtime())))
    return wrapper
```

```
@heig
def myFunc(a, b):
    return [(a / b) for i in range(100)]

myFunc(2, 3)
```

1.5 Exception

[36]:
```
try:
    print(1/0)
except ZeroDivisionError as e:
    print(f"Error: {e}")
```

```
Error: division by zero
```

1.6 Open file

[ ]:
```
fp = open("file.txt", "r") # w: write, r: read, a: append
fp.read()
fp.readline()
fp.readlines()
fp.close()

with open("file.txt", "w") as fp: # with open automatically close the file
    fp.write("Hello World")
```

1.7 NumPy

[ ]:
```
import numpy as np
a = np.arange(1, 10)
print(a)
b = np.arange(1, 10, 2)
print(b)
c = np.linspace(1, 10, 10)
print(c)
d = np.zeros((2, 3))
print(d)
e = np.ones((2, 3))
print(e)
a2 = a[:, np.newaxis]
print(a2)
print(a2 * b)
```

1.8 Click

[ ]:
```
import click

@click.group()
def cli():
    pass

@cli.command()
@click.argument('a', type=float, nargs=1, required=True)
@click.argument('b', type=float, default=0.0)
@click.argument('c', type=float)
def quad(a, b, c):
    delta = b**2 - 4*a*c
    # Display x1 and x2 even if complex
    x1 = (-b + delta**0.5) / (2*a)
    x2 = (-b - delta**0.5) / (2*a)
    click.echo(f'x1 = {x1}')
    click.echo(f'x2 = {x2}')

if __name__ == '__main__':
    quad()
```

1.9 Pandas

[29]:
```
import pandas as pd
df = pd.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]})
df.describe()
```

[29]:

	a	b
count	3.0	3.0
mean	2.0	5.0
std	1.0	1.0
min	1.0	4.0
25%	1.5	4.5
50%	2.0	5.0
75%	2.5	5.5
max	3.0	6.0

[28]:
```
df.head()
```

[28]:

	a	b
0	1	4
1	2	5
2	3	6

[30]:
```
df.loc[df['a'].isin([1, 3]), ['a', 'b']]
```

[30]:

	a	b
0	1	4
2	3	6

1.10 Flask

[ ]:
```python
from flask import Flask, render_template, request

navs = [
    {'name': 'Home', 'url': '/'},
    {'name': 'Transfer', 'url': '/add'},
    {'name': 'Transactions', 'url': '/transactions'},
]

app = Flask(__name__)

@app.context_processor
def inject_user():
    return dict(navs=navs)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/add', methods=['GET', 'POST'])
def add():
    if request.method == 'POST':
        a = request.form['a']
        return 'Transfer done'
    if request.method == 'GET':
        a = request.args.get('a')
    return render_template('add.html')
```

1.10.1 index.html

```
{% extends "base.html" %}
{% block content %}
<h1 class="mt-5">Bonjour {{user}}</h1>
{% endblock %}
```

1.10.2 base.html

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

```
    <!-- Bootstrap CSS -->
    <!-- <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/boo
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css"
    <title>PyBanking</title>
  </head>
  <body>
    {% include "nav.html" %}
    <div class="container">
        {% block content %}
        {% endblock %}
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" inte
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.mi
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integri
  </body>
</html>
```