Cryptanalysis (암호분석)

– Python (Part 1) –

Caesar Cipher

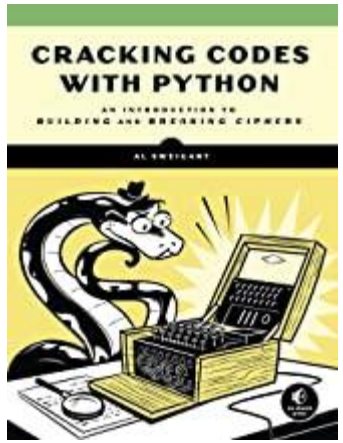
2020. 3

목차

1. Python 설치 및 환경 설정

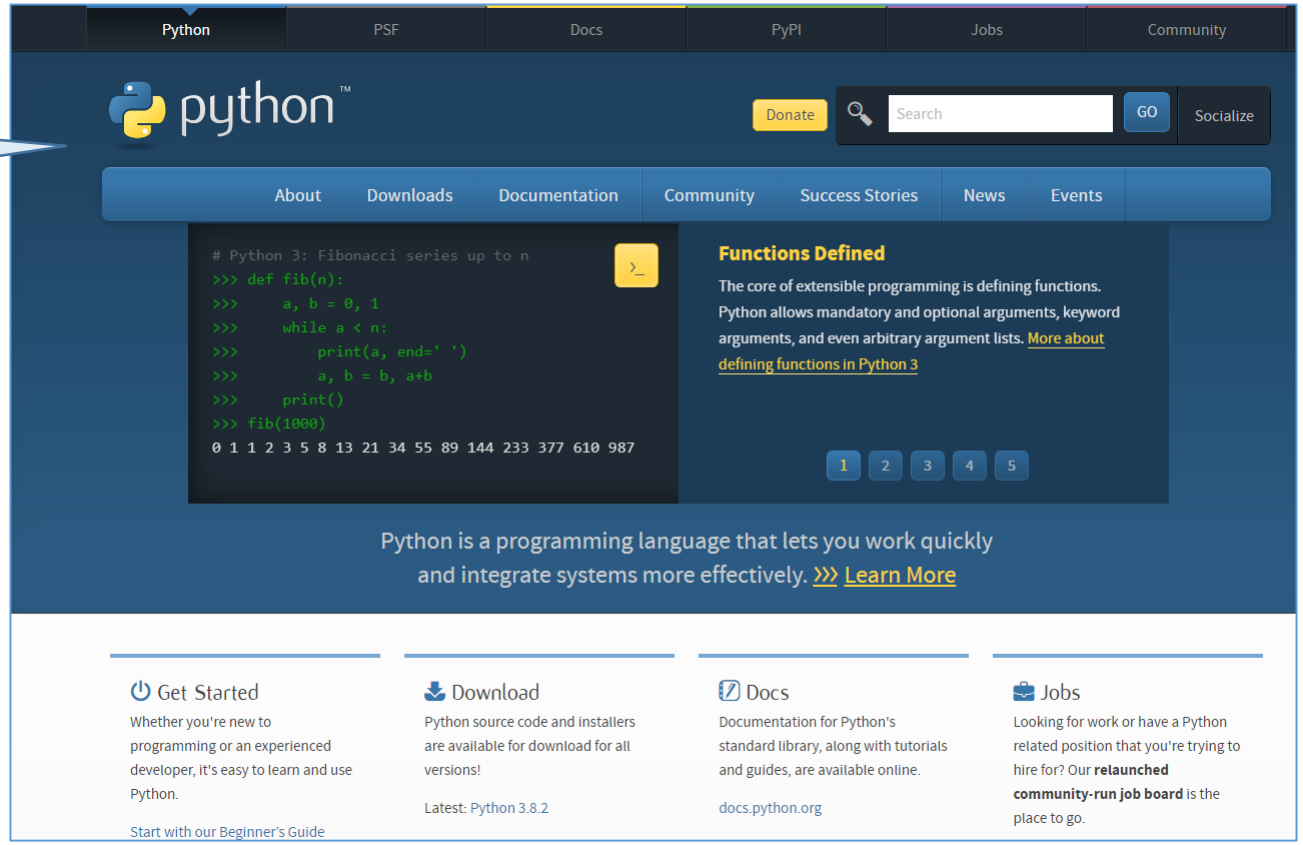
2. 문자열(string) 다루기

3. Caesar cipher



Python 설치하기

https://www.python.org/



IDE 개발 환경

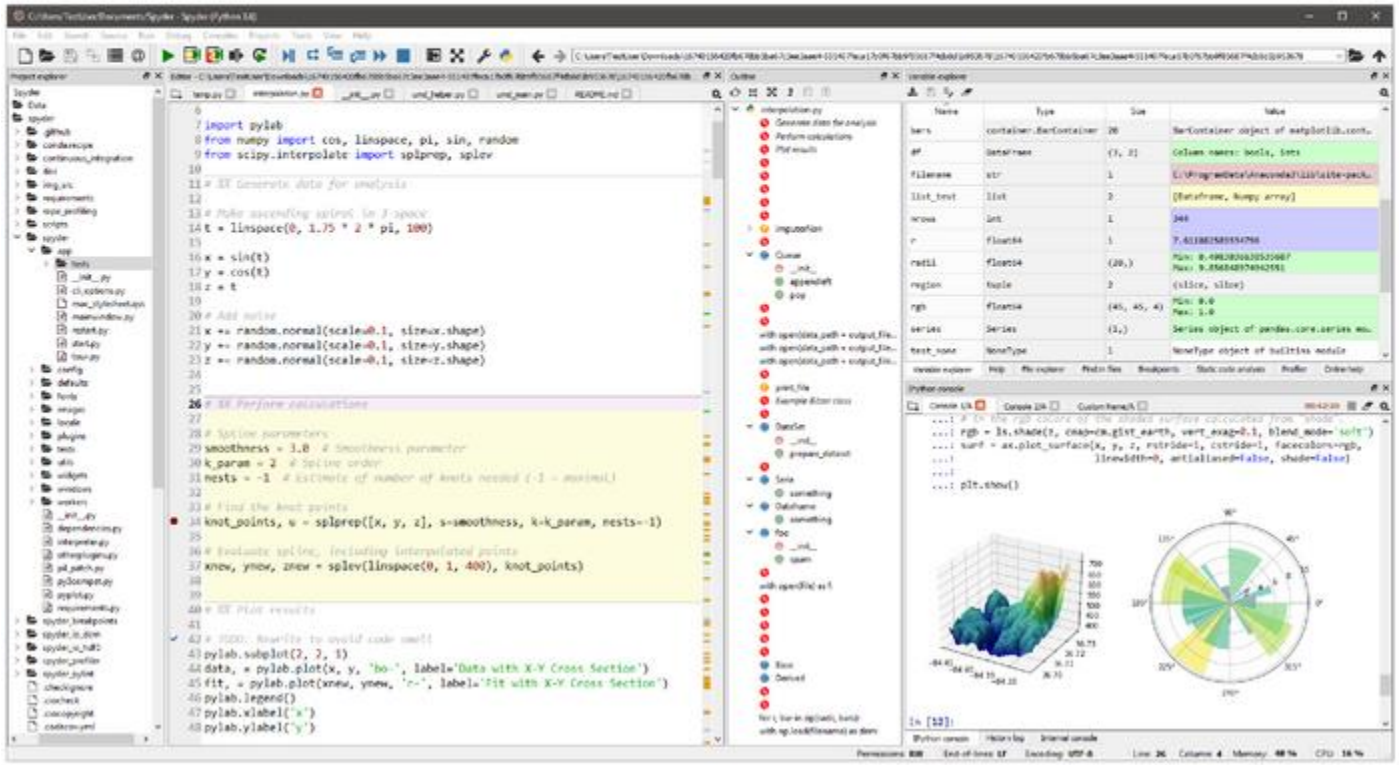
Spyder 설치

- 통합 개발환경 구축
  - 소스코드 저장/재사용
  - 디버깅

  - 예: Spyder, Eclipse
  - 한가지 환경만 선택하면 됨

SPYDER

The Scientific Python Development Environment



IDE for all languages

Eclipse | Microsoft Visual Studio | Komodo Edit | NetBeans | Xcode | PyCharm

문자열(String) 사용 하기

- String concatenation with + operator
- String replication with * operator
- Indexing with []
  - positive, negative index: [n] [-n]
- Slicing with [:] (range)
  - two indexes: [n:m]
  - blank index: [:m] [n:]
- Repeated index [:][]

```
# 문자열 합치기
print('Hello, ' + 'Python')


# 문자열 인덱싱
print('Hello'[0])
print('Hello'[-1])
print('Hello'[-2])
print('Hello'[2])


# 문자열 슬라이싱
print('Hello, world'[0:4])
print('Hello, world'[-5:-1])
print('Hello, world'[-5:])
```

입출력 함수

- 출력 함수 print()
- 입력 함수 input()
- 주석(comments) #

```
# 사용자 입력 받기
print("What is your name?")
myName = input()
print("Nice to meet you, " + myName + ".")
```

Sample Code: Reverse Cipher

- 반복문 while
- 문자열 길이: len()

```
# 문자열 거꾸로 만들기

message = 'This is a sample text.'
translated = ''
i = len(message)-1
while i >= 0:
    translated = translated + message[i]
    print('translated message = ', translated)
    i = i - 1
print('\n Final Result = ', translated)
```

- 문자열 찾기: find()
- 나머지 연산자 %
  - 연산의 우선 순위에 주의

```
#-- find()
msg = "abcdefghijklmnopqrstuvwxyz"
print(msg.find("def"))
print(msg.find("aa"))
print(msg[msg.find("j"):])

#--- % 연산자
print(5 % 3)
print( (10 + 10) % 3)
print( 10 + 10 % 3)
```

Caesar Cipher (Encryption)

```
plain_msg = 'This is a plaintext message to be encrypted.'
key = 3 # select from (0-25)

UpAlphabet    = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
LowerAlphabet = 'abcdefghijklmnopqrstuvwxyz'


##--- 암호화 과정
cipher_msg = ''
for symbol in plain_msg :
    if symbol in UpAlphabet:
        symbol_idx = UpAlphabet.find(symbol)
        trans_idx = (symbol_idx + key) % len(UpAlphabet)
        cipher_msg = cipher_msg + UpAlphabet[trans_idx]
    elif symbol in LowerAlphabet:
        symbol_idx = LowerAlphabet.find(symbol)
        trans_idx = (symbol_idx + key) % len(LowerAlphabet)
        cipher_msg = cipher_msg + LowerAlphabet[trans_idx]
    else:
        cipher_msg = cipher_msg + symbol

print('PLAINTEXT  = ', plain_msg)
print('CIPHERTEXT = ', cipher_msg)
```

Caesar Cipher (Decryption)

```
key = 3 # select from (0-25)

UpAlphabet    = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
LowerAlphabet = 'abcdefghijklmnopqrstuvwxyz'

##--- 복호화 과정
recovered_msg = ''
for symbol in cipher_msg :
    if symbol in UpAlphabet:
        symbol_idx = UpAlphabet.find(symbol)
        trans_idx = (symbol_idx - key) % len(UpAlphabet)
        recovered_msg = recovered_msg + UpAlphabet[trans_idx]
    elif symbol in LowerAlphabet:
        symbol_idx = LowerAlphabet.find(symbol)
        trans_idx = (symbol_idx - key) % len(LowerAlphabet)
        recovered_msg = recovered_msg + LowerAlphabet[trans_idx]
    else:
        recovered_msg = recovered_msg + symbol

print('CIPHERTEXT = ', cipher_msg)
print('PLAINTEXT  = ', recovered_msg)
```