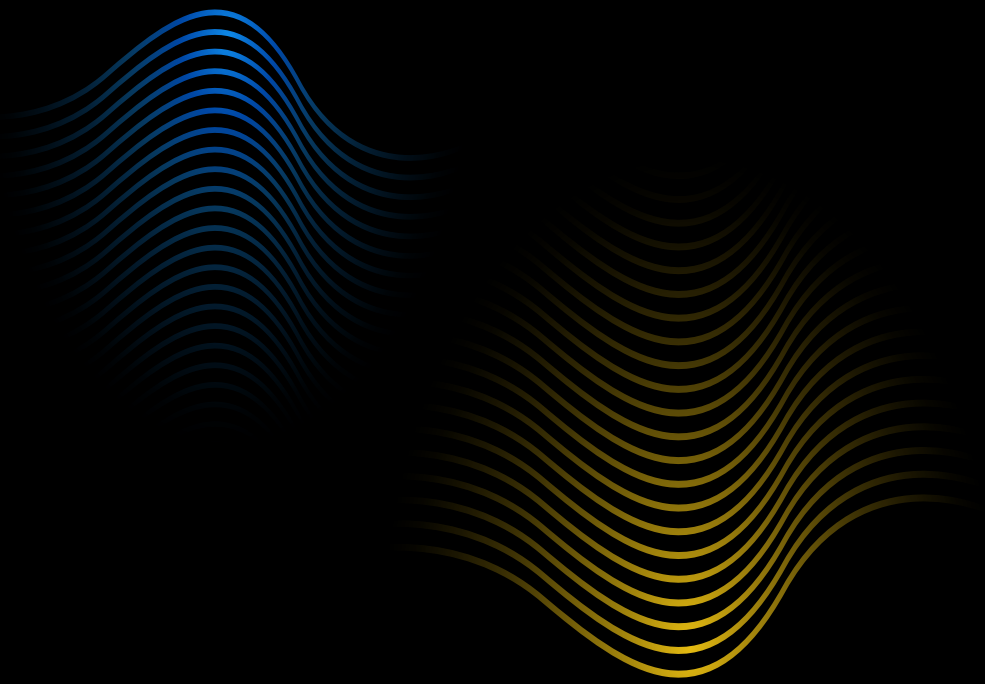
Samsung R&D Institute Poland

Python Tech Guide

Poszerzaj i utrwalaj znajomość zagadnień technicznych, korzystając z praktycznych wskazówek naszych ekspertów.



Samsung Tech Guides

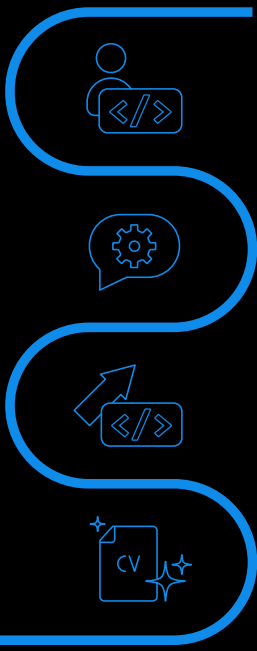
Drogi kandydacie!
Droga kandydatko!

Zapraszamy Cię do zapoznania się z Python Technical Guide - technicznym przewodnikiem poświęconym jednemu z najbardziej popularnych języków programowania!

Znajdziesz w nim tematy, które warto powtórzyć przed spotkaniem technicznym w rekrutacji do Samsung R&D!

Dodatkowo nasi Inżynierowie przygotowali przykładowe zadania testowe, które pomogą Ci sprawdzić swoją wiedzę.

Python Technical Guide - Dla kogo?



Dla kandydatów aplikujących na stanowiska, w których wymagana jest znajomość Pythona,

dla kandydatów zaproszonych już na spotkanie techniczne do Samsung R&D,

dla początkujących Inżynierów chcących poszerzyć/utrwalić swoją wiedzę z Pythona,

dla osób zainteresowanych w przyszłości rekrutacją do Samsung R&D.

Powodzenia!

Python - podstawy

Python jest ciągle rozwijającym się, wszechstronnym językiem programowania z rozbudowaną biblioteką standardową i wieloma dostępnymi bibliotekami zewnętrznymi. Warto znać podstawowe elementy budujące ten język.

- Czym są zmienne i jak ich używać?
- Proste typy wbudowane.
- Logika boolowska.
- Wbudowane typy kolekcji.
- Porównywanie obiektów między sobą dla różnych typów.
- Definiowanie i wywoływanie funkcji.
- Czym są moduły i pakiety?
- Czym są wyjątki?
- Programowanie z użyciem klas.

Zmienne

Czym są zmienne i jak ich używać?

- Wyjątek **NameError**.
- Instrukcja przypisania **=**.
- Rozszerzone przypisania **+=** itd.
- Zakresy nazw (lokalny, globalny i wbudowany).
- Powszechne konwencje nazewnicze w Pythonie.
- Jakie nazwy są „dobre”?
- Znaczenie podkreślnika na początku nazwy.

Proste typy wbudowane (int, float, str, bytes)

- Konwersje pomiędzy prostymi typami wbudowanymi.
- Osobliwości typu **float** (precyzja, porównywanie).
- Podstawowe metody napisów **(join()**, **lower()**, **split()**, **strip()**, **encode()**, **...).**
- *Escape sequences* i różne postaci literałów napisowych.
- Funkcje wbudowane: **abs()**, **min()**, **max()**, **sum().**
- Interpolacja napisów (formatowanie), *f-strings*.
- Operatory arytmetyczne: **+**, **-**, *****, **/**, **% i //.**
- Różnica między **str() a repr().**

Logika boolowska

- Stałe wbudowane: **None**, **True**, **False.**
- Pojęcia *"truthy"* i *"falsy".*
- Instrukcja **if-elif-else.**
- Operatory: **and**, **or**, **not.**
- Właściwość *short-circuit.*
- Funkcja wbudowana **bool().**

Wbudowane typy kolekcji (list, tuple, dict, set, str, bytes)

- Co je różni? Kiedy warto ich używać?
- Które operacje na kolekcjach są szybkie, a które wolne?
- Konwersje jednych kolekcji w inne, tworzenie kopii.
- Jak wyglądają literały (zwłaszcza pustych i jednoelementowych) kolekcji?
- Jak konstruować, modyfikować i odczytywać zawartość kolekcji?
- Które obiekty są "*hashable*" i kiedy ma to znaczenie?
- Czym są list/dict/set *comprehensions*?
- Operatory konkatenacji **(+**, ***)** i zawierania **(in**, **not in).**
- Operator **[]** (indeksowanie) oraz notacja slice **[:].**
- Czym jest *iterable* i *iterator*?
- Instrukcje: **while**, **for**, **break**, **continue.**
- Funkcje wbudowane: **enumerate()**, **range()**, **zip().**
- Sortowanie kolekcji; zmiana kryterium sortowania.

- Rozpakowywanie (unpacking) kolekcji w literałach i wywołaniach funkcji przy użyciu ***** i ******.
- Rozpakowywanie (destructuring) kolekcji do zmiennych.

Porównywanie obiektów między sobą dla różnych typów

- Operatory: **==**, **!=**, **<**, **>**, **<=**, **>=**.
- Mieszanie różnych typów w jednym porównaniu.

Definiowanie i wywoływanie funkcji

- ***args** i ****kwargs**.
- Instrukcje **def** i **return**.
- Argumenty pozycyjne i typu *keyword*.
- Domyślne wartości argumentów.
- Czym jest stos wywołań?
- Czym jest *callback*?
- Generatory i słowo kluczowe **yield**.

Czytanie z i pisanie do plików

- Otwieranie plików, różne tryby.
- Pliki tekstowe a binarne.
- Ścieżki względne i absolutne.
- Czym jest kodowanie znaków?
- Instrukcja **with**.
- Czym są standardowe wejście/wyjście?
- Funkcje wbudowane: **print()**, **input()**.

Moduły i pakiety

Tworzenie oprogramo-wania

- Czym są testy jednostkowe, jak napisać prosty test?
- Podstawy pracy z Gitem i GitHubem.
- Dobre praktyki w pisaniu kodu.

Czym są wyjątki? A także...

- Instrukcje: **raise** i **try-except-finally**.
- Wbudowane typy wyjątków i kiedy można się ich spodziewać: **SyntaxError**, **NameError**, **TypeError**, **ValueError**, **IndexError**, **KeyError**, **OSError.**
- Znaczenie typów **BaseException** i **Exception.**
- Czym jest hierarchia typów wyjątków i kiedy ma znaczenie?
- Czytanie *tracebacku.*
- Kiedy i które wyjątki można obsługiwać?
- Jak zdefiniować nowy typ wyjątku?

Programowanie z użyciem klas

- Pojęcia: obiekt/instancja, typ/klasa, enkapsulacja.
- Funkcje wbudowane: **type()**, **isinstance()**, **super().**
- Definiowanie metod i atrybutów.
- Argument **self.**
- Metoda **__init__()**,
- Czym są „magiczne" metody, np. **__str__()**?
- Gettery, settery i dekorator **@property.**
- Dziedziczenie i kompozycja.

1.
Rozważmy pliki zawierające liczby takie, jak ten:

```
  6.98
3.21

8.009
2.1
```

- Suma liczb w tym pliku to 20,299.
- Pliki zawierają najwyżej jedną liczbę w linijce i nie zawierają innych danych. Mogą zawierać puste linie i dodatkowe spacje.
- Napisz funkcję, która przyjmuje listę ścieżek do plików (napisów) i zwraca ścieżkę do tego pliku, w którym suma liczb jest najmniejsza.

2.
Napisz funkcję, która przekształca listę kwalifikowanych nazw (napisów) na słownik mapujący przestrzenie nazw na listy nazw.

Przestrzeń nazw jest oddzielona od nazwy dwoma dwukropkami (`::`). Brak `::` oznacza przestrzeń nazw opisana pustym napisem.
Na przykład dla poniższego argumentu:

```
[
    'stick',
    'animals::tiger',
    'birds::crow',
    'animals::lion',
    'animals::birds::penguin',
    'pebble'
]
```

Funkcja powinna zwrócić:

```
{
    'animals': ['tiger', 'lion'],
    'birds': ['crow'],
    'animals::birds': ['penguin'],
    '': ['stick', 'pebble']
}
```

3.
Rozważmy pliki tekstowe zawierające liczby. Oto przykładowy plik:

Klucze powinny być oddzielone od wartości znakiem =.
Pary klucz-wartość powinny być oddzielone od siebie pojedynczą spacją.
Jeżeli wartość nie jest napisem, funkcja powinna ją na niego
przekonwertować. Na przykład dla poniższego argumentu:

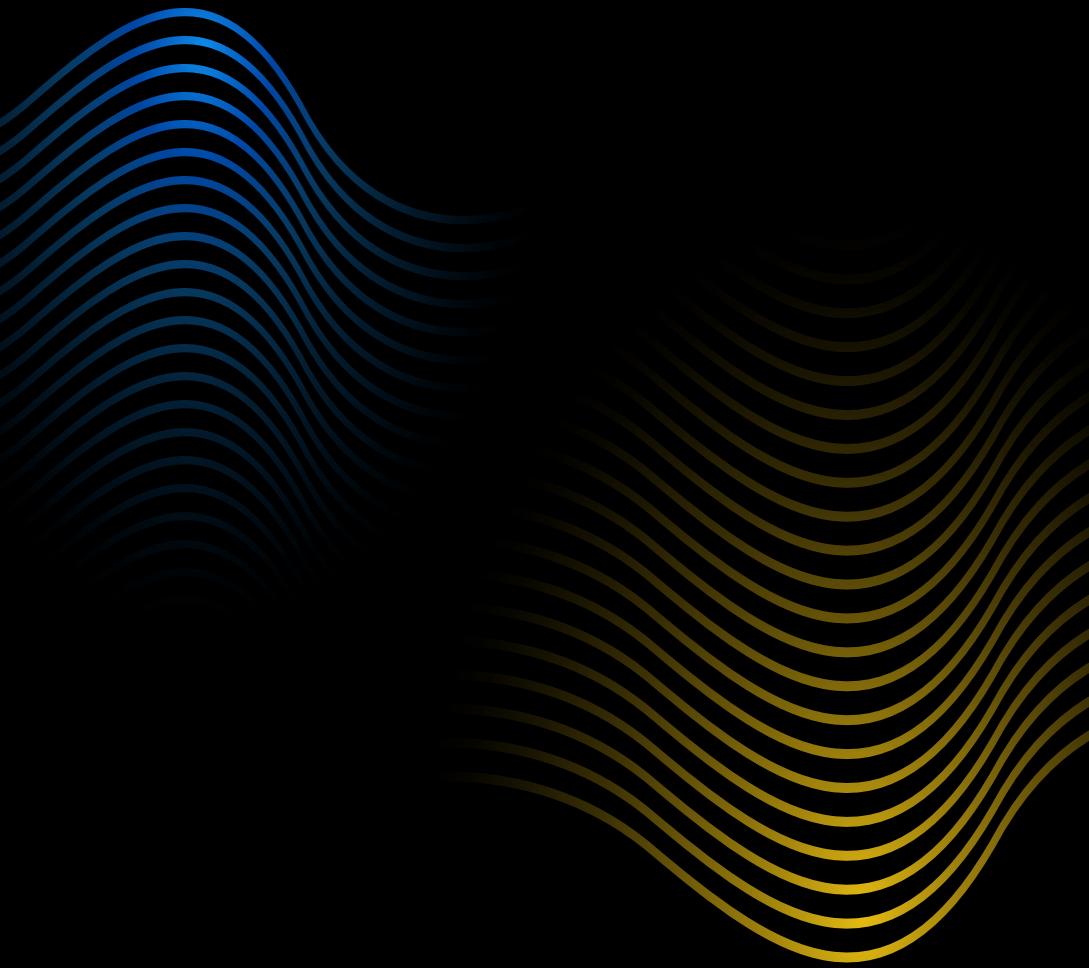
```
{'price': 7, 'currency': 'dollar'}
```

Funkcja powinna zwrócić:

```
'price=7 currency=dollar'
```

Funkcja powinna zgłosić wyjątek `ValueError`, jeżeli którykolwiek klucz zawiera znak `=` lub jeżeli którykolwiek klucz lub którakolwiek wartość zawiera spację.

Samsung R&D Institute Poland



Samsung Tech Guides