FIPU: rs-mid - Službeni šalabahter

Moduli `random` i `datetime`

```
random.random() # vraća slučajni float između [0.0 i 1.0)
random.randint(a, b) # vraća slučajni cijeli broj između a i b (uključivo)
random.choice(iterable) # vraća slučajni element iz iterable
random.choices(iterable, weights=None, k=1) # vraća listu od k slučajnih elemenata iz
iterable, s opcionalnim weightovima
random.uniform(a, b) # vraća slučajni float između a i b (može uključivati oba kraja)
datetime.now() # vraća trenutni datum i vrijeme
datetime.fromisoformat(date_string) # parsira ISO formatirani string u datetime objekt
datetime.timestamp() # convert u UNIX timestamp
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0,
weeks=0) # predstavlja vremenski interval
datetime.datetime +- datetime.timedelta # zbrajanje/oduzimanje datuma i vremenskog
intervala
```

Funkcije višeg reda i druge korisne funkcije

```
map(fn, iterables) # primjenjuje 'fn' na svaki element iz 'iterables'.
filter(fn, iterables) # filtrira elemente 'iterables' prema predikatu 'fn'.
any(iterables) # vraća True ako je bilo koji element 'iterables' True
all(iterables) # vraća True ako su svi elementi 'iterables' True
sorted(iterables, key=fn, reverse=False) # sortira 'iterables' prema 'fn'; reverse=True
za obrnut redoslijed
sum(iterables) # zbraja sve elemente iz 'iterables'
min/max(iterables, key=fn) # vraća minimalni/maksimalni element iz 'iterables' prema 'fn'
(opcionalno)
```

```
lambda argument_1, argument_2, ...: izraz # anonimna funkcija
lambda argument: izraz_if_true if condition else izraz_if_false # lambda s if-else
```

Comprehension sintaksa

```
[expression for item in iterable] # list comp
[expression for item in iterable if condition] # list comp s if
{key_expression: value_expression for item in iterable} # dict comp
{key_expression: value_expression1 for item in iterable for item2 in iterable2} # s više
iterables
{key_expression: value_expression for item in iterable if condition} # dict comp s if
[expression1 if condition else expression2 for element in iterable] # list comp s if-else
{key_expression: value_expression1 if condition else value_expression2 for item in
iterable} # dict comp s if-else
[expression for item_1, item_2 in zip(list1, list2, list3, ...)] # list comp s više
iterables
```

Klase i objekti

```
class SomeClass:
  def __init__(self, argument_1_value, argument_2_value):
    self.argument_1 = argument_1_value
    self.argument_2 = argument_2_value
  def some_method(self):
    return self.argument_1 + self.argument_2
instance = SomeClass(argument_1_value, argument_2_value)
instance.some_method()

class SubClass(SuperClass):
  def __init__(self, argument_1_value, argument_2_value, argument_3_value):
    super().__init__(argument_1_value, argument_2_value) # poziv konstruktora nadklase
    self.argument_3 = argument_3_value
```

asyncio

```
async def main(): # main korutina
  pass
asyncio.run(main()) # pokretanje korutine
asyncio.sleep(sec) # čekanje u korutini
await coroutine() # čekanje na završetak korutine
asyncio.gather(coroutine1, coroutine2, coroutine3...) # schedule, run and wait
asyncio.gather(*lista, return_exceptions=Bool) # schedule, run and wait s listom
korutina/taskova
asyncio.create_task(coroutine) # izrada/schedule Taska
await task # čekanje na završetak Taska
asyncio.wait_for(corutine/task, timeout) # čekanje na Task s timeoutom
asyncio.TimeoutError # Timeout iznimka

try: ... except Exception as e: ... finally: ... # rukovanje iznimkama
```

aiohttp.ClientSession

```
with neki_resurs as alias: # context manager sintaksa
async with aiohttp.ClientSession() as session: # izrada klijentske sesije kroz context
manager
  response = await session.get(url) # slanje GET zahtjeva
  response = await session.post(url, json=data) # slanje POST zahtjeva s JSON podacima u
tijelu zahtjeva
  response_data = await response.json() # deserijalizacija JSON odgovora
  response.status # statusni kod odgovora
# mjerenje vremena izvršavanja programa
start = time.perf_counter() # ili time.time()
end = time.perf_counter() # ili time.time()
print(f"Izvršavanje je trajalo {end - start:.2f} sekundi.")
```