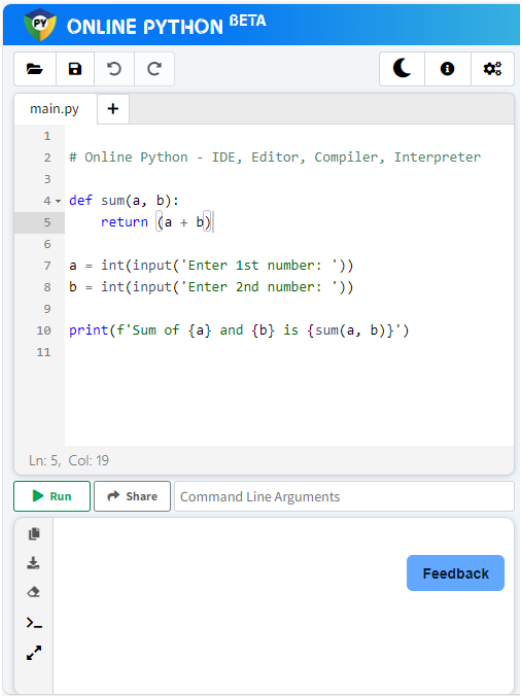
Python ohjelmointi

Python-ohjelma on lista erilaisista ohjeista, jotka tietokone suorittaa rivi kerrallaan. Python on yksi suosituimmista ohjelmointikielistä. Python kieli on kehitetty jo 1980 luvulla ja se on saanut nimensä sen ajan Monty Python sarjasta. Pythonia voidaan käyttää yksinkertaisesta numeraalisesta laskennasta aina vaativaan tieteelliseen laskentaan. Pythonissa ei itsessään ole mahdollisuutta ladata kuvia tai mediaa, mutta se onniustuu erilaisten modulien avulla.

Python on vakiintunut ohjelmistokieli data-analytiikassa. Python on myös erinomainen kieli erilaisten asioiden automatisointiin. Yksinkertaisten asioiden automatisointi ei vaadi ammattitason osaamista ja sillä on helppoa toteuttaa nopeasti erilaisia automatisointeja. Python on avoimen lähdekoodin tuote, eli sitä voi käyttää ilmaiseksi. Python on helppo ja tehokas ohjelmointikoodi teknisessä käytössä ja robotiikassa.

Verkkokehitysympäristö Online Python IDE
www.online-python.com

Ohjelmoi, suorita ja jaa Python-koodia verkossa ilmaiseksi Python-kehitysympäristön avulla (IDE - online-integrated python's development environment). Se on yksi luotettavimmista ja tehokkaimmista Python-ohjelmointikielen online-tulkkaajista. Sinun ei tarvitse huolehtia Python-ympäristön luomisesta paikallisesti tietokoneelle. Nyt voit suorittaa Python-koodia suoraan verkkoselaimessa. Tämän Python-editorin käyttö on helppoa ja nopeaa. Koodaa vain ohjelma ja paina RUN-painiketta!



Tietotyypit

Tietotyypit ovat keskeinen käsite ohjelmoinnissa, ja niitä käytetään kuvaamaan erilaisia tietoja, joita ohjelmat käsittelevät. Tietotyypit määrittävät, miten tietoa tallennetaan, käsitellään ja esitetään tietokonejärjestelmässä. Alla on joitakin yleisimpiä tietotyyppejä Python-ohjelmoinnissa:

- **Merkkijonot (strings)**: `'hello'`, `"world"`.

Lukuarvot:

- **Kokonaisluvut (integers)**: `3`, `-14`, `1000`.
- **Liukuluvut (floats)**: `3.14`, `2.71828`, `-0.001`.

Boolean-arvot (booleans): `True` tai `False`
Listat (lists): Listat ovat järjestettyjä tietorakenteita, jotka voivat sisältää erilaisia tietotyyppejä. Ne määritellään hakasuluilla `[ ]` ja niiden alkiot erotetaan pilkuilla. Esim: `[1, 2, 3, 4]`, `['apple', 'banana', 'orange']`.

Kommentit

Kommentointi auttaa selittämään koodin toimintaa ja tekee koodin ymmärrettävämmäksi muille ohjelmoijille sekä itselle. Kommentit näkyvät vain koodirakenteessa. Pythonissa on kaksi tapaa kommentoida koodia:

Yksiriviset kommentit: Näitä käytetään selittämään yhden rivin koodia. Ne alkavat #-merkillä ja jatkuvat rivin loppuun.

Moniriviset kommentit: Näitä käytetään kun kommenttirivejä on useita. Ne alkavat ja loppuvat kolmeen lainausmerkkiin.

Esimerkki:

```
# Tämä on yksirivinen kommentti
x = 5  # Alustetaan x arvoon 5

'''
Tämä on
monirivinen kommentti.
Käytämme tätä yleensä pidempiin selityksiin.
'''
```

Muuttuja

Muuttujat ovat nimiä, joille voidaan antaa arvoja ja jotka voidaan sitten käyttää viittaamaan näihin arvoihin ohjelmoinnissa. Ne ovat olennainen osa Python-ohjelmointia.

- **Nimenanto**: Muuttujille annetaan nimiä, jotta niihin voidaan viitata koodin eri osissa. Nimen on oltava validi, mikä tarkoittaa, että se voi sisältää kirjaimia, numeroita ja alaviivoja, mutta se ei saa alkaa numerolla eikä sisältää välilyöntejä.
- **Arvon tallentaminen**: Muuttujaan voidaan tallentaa tietty arvo, joka voi olla kokonaisluku, liukuluku, merkkijono, totuusarvo (boolean), lista, jne.

- **Dynaaminen tyypitys**: Pythonissa muuttujan tietotyyppiä ei tarvitse määrittää etukäteen, vaan se määrittyy automaattisesti sen perusteella, millainen arvo siihen asetetaan. Tämä tunnetaan nimellä dynaaminen tyypitys.
- **Arvon päivittäminen**: Muuttujan arvoa voidaan päivittää milloin tahansa asettamalla siihen uusi arvo. Tämä tarkoittaa, että muuttujan arvo voi vaihdella ohjelman suorituksen aikana.
- **Käyttö**: Muuttujia käytetään erilaisten tietojen tallentamiseen ja käsittelemiseen ohjelmassa. Ne mahdollistavat koodin modulaarisuuden ja joustavuuden, koska ne sallivat saman arvon käytön

Funktio

Funktiot ovat nimettyjä koodilohkoja Pythonissa, jotka suorittavat tiettyjä tehtäviä, kun niitä kutsutaan. Funktion avulla voidaan ryhmitellä koodia, jotta samaa toiminnallisuutta voidaan käyttää uudelleen monissa eri kohdissa ohjelmaa. Tässä on lisätietoja funktioista Pythonissa:

```python
# Funktion määrittely
def greet(name):
    return "Hello, " + name + "!"

# Funktion kutsu ja tuloksen tulostaminen
message = greet("John")
print(message)  # tulostaa: "Hello, John!"
```

Ehtorakenteet

- **Ehdollinen lauseke (if)**: Ehdollinen lauseke **`if`** mahdollistaa tietyjen koodirivien suorittamisen vain, jos tietyt ehdot ovat tosia. Jos ehto on tosi, lausekkeen sisällä oleva koodi suoritetaan. Muussa tapauksessa se ohitetaan.
- **Vaihtoehtoiset ehdot (elif)**: Lisäksi **`if`**-lausekkeiden kanssa voidaan käyttää **`elif`**-lausekkeita, joita käytetään tarkistamaan useita ehtoja peräkkäin.
- **Ehtojen vertailu**: Ehdolliset lausekkeet käyttävät ehtoja, jotka voivat olla vertailuja kahden arvon välillä. Yleisimpiä vertailuoperaattoreita ovat **`==`** (yhtäsuuruus), **`!=`** (epäyhtäsuuruus), **`<`** (pienempi kuin), **`>`** (suurempi kuin), **`<=`** (pienempi tai yhtä suuri kuin), **`>=`** (suurempi tai yhtä suuri kuin).
- **Ehtolausekkeiden yhdistäminen**: Ehtolausekkeita voidaan myös yhdistää logiikan avulla käyttämällä **`and`**, **`or`**, **`not`** -operaattoreita.

- **Sulkeutuvuus**: Pythonissa lohkojen (koodin, joka kuuluu tiettyyn ehtorakenteeseen) sisältö määritellään lohkon sisentämisen avulla (**tab**)

```
print("Tervetuloa tietokilpailuun!")
print("Vastaa seuraaviin kysymyksiin oikein voittaaksesi.")

score = 0  # Alustetaan pistemäärä

# Kysymykset
question1 = "Mikä on maailman suurin valtameri? "
question2 = "Mikä on pääkaupunki Suomessa? "
question3 = "Mikä on maailman korkein vuori? "

# Oikeat vastaukset
answer1 = "Tyyni valtameri"
answer2 = "Helsinki"
answer3 = "Mount Everest"

# Kysytään ensimmäinen kysymys
guess1 = input(question1)
if guess1.capitalize() == answer1:
    print("Oikein!")
    score += 1
else:
    print("Väärin!")

# Kysytään toinen kysymys
guess2 = input(question2)
if guess2.capitalize() == answer2:
    print("Oikein!")
      score += 1
else:
    print("Väärin!")

# Kysytään kolmas kysymys
guess3 = input(question3)
if guess3.capitalize() == answer3:
    print("Oikein!")
    score += 1
else:
    print("Väärin!")

# Lopputuloksen tulostus
print("Pistemääräsi on:", score, "/ 3")
if score == 3:
    print("Onnittelut! Voitit tietokilpailun!")
else:
    print("Kiitos osallistumisesta.")
```

Toistorakenteet

Pythonissa toistorakenteet/silmukat ovat rakenteita, joita käytetään toistamaan tiettyjä ohjeita tai lohkoja tietyn määrän kertoja tai kun tietty ehto on voimassa.

while-silmukka

While-silmukka on tehokas tapa suorittaa koodia toistuvasti niin kauan kuin tietty ehto on voimassa. On tärkeää varmistaa, että ehto muuttuu jossain vaiheessa, jotta silmukka ei jää ikuisesti toistamaan samaa koodia. Lisäksi on tärkeää välttää äärettömiä silmukoita, jotka voivat johtaa ohjelman toimintahäiriöihin.

Esimerkki:
Kuvitellaan tilanne, jossa haluamme tulostaa numerot 1-5.

```
1  numero = 1
2  while numero <= 5:
3      print(numero)
4      numero += 1
```

for-silmukka

Tiettyjä toimintoja suoritettaessa on usein tarpeen käydä läpi kokoelma alkioita, kuten listan jäseniä tai tietyllä välillä olevia numeroita. Tähän tarkoitukseen Pythonissa käytetään usein for-silmukkaa. for-silmukan voi myös yhdistää muihin Pythonin rakenteisiin, kuten if-lauseisiin, jolloin voidaan tehdä tarkistuksia jokaiselle iteraatiolle:

Tämä tulostaa luvut 0, 1, 2, 3 ja 4. **`range(5)`** luo lukujonon 0:sta 4:ään.

```
1  for i in range(5):
2      print(i)
3
```

Keskeytys

"break" komento Pythonissa on käytetty ohjauslauseke, joka keskeyttää lähimmän silmukan suorituksen, kuten for-loopin tai while-loopin. Kun "break" käskyä suoritetaan, ohjelman suoritus jatkuu välittömästi silmukan jälkeiseltä riviltä, ohittaen loput silmukan iteraatioista.

```
1  for i in range(10):
2      print(i)
3      if i == 5:
4          break
```

Harjoituksia: Luvun arvauspeli

```python
import random

def arvaa_luku():
    oikea_luku = random.randint(1, 100) # Arvotaan satunnainen luku väliltä 1-100
    arvauskerrat = 0

    while True:
        arvaus = int(input("Arvaa luku väliltä 1-100: "))
        arvauskerrat += 1

        if arvaus < oikea_luku:
            print("Luku on suurempi, yritä uudelleen.")
        elif arvaus > oikea_luku:
            print("Luku on pienempi, yritä uudelleen.")
        else:
            print(f"Oikein! Arvasit luvun {oikea_luku} {arvauskerrat} yrityksellä.")
            break

arvaa_luku()
```

Arvaa sana

```python
def arvaa_sana():
    sana = "suklaa"
    oikein_arvatut = []
    yritykset = 0
    max_yritykset = 7

    while True:
        salattu_sana = ""
        for kirjain in sana:
            if kirjain in oikein_arvatut:
                salattu_sana += kirjain
            else:
                salattu_sana += "_"

        if salattu_sana == sana:
            print(f"Oikein! Sana oli '{sana}'. Onneksi olkoon!")
            break

        print(f"Salattu sana: {salattu_sana}")
        arvaus = input("Arvaa kirjain: ").lower()

        if arvaus in sana:
            if arvaus not in oikein_arvatut:
                oikein_arvatut.append(arvaus)
                print("Oikein! Jatka samaan malliin.")
            else:
                print("Olet jo arvannut tämän kirjaimen.")
        else:
            print("Väärin. Yritä uudelleen.")
            yritykset += 1
            if yritykset >= max_yritykset:
                print("Hävisit pelin! Liikaa yrityksiä.")
                break

arvaa_sana()
```