Tampereen ammattikorkeakoulu

Oppimispäiväkirja

Esko Rautiainen

Oppimispäiväkirja
Syyslukukausi 2025

Tutkinto-ohjelman nimi
Tradenomi (AMK), Tietojenkäsittelyn tutkinto-ohjelma

Kurssi
4A00HB81-3001 Python-ohjelmointi

Ryhmä: 24TIKOOT1

SISÄLLYS

ERITYISSANASTO

TAMK	Tampereen ammattikorkeakoulu
op	opintopiste

1 KT1

#KT1 Selitys opettajalle
Tässä ratkaisussa kysymme kaikki muuttujat käyttäjältä erikseen.
Käytämme printtaukseen f-strings toimintoa, sillä se helpottaa huomattavasti koodin kirjoittamista. Ehkä parempi ratkaisu olisi vastaanottaa käyttäjän syötteet Arrayyn tai Arraylistaan ja hakea ne sieltä indexi kerrallaan. Silloin print(f ” ” olisi huomattavasti lyhyempi ja koodi helpommin

```
aloitus = input("Kirjoita selityksen aloitus: ")
opettaja = input("Kirjoita opettajan nimi: ")
kotitehtävän_tyyppi = input("Kirjoita kotitehtävän tyyppi: ")
lemmikkieläimen_laji = input("Kirjoita lemmikkieläimesi laji: ")
lemmikkieläimen_nimi = input("Kirjoita lemmikkieläimesi nimi: ")
lemmikkieläimen_teko = input("Kirjoita mitä lemmikkieläimesi teki: ")

print(f"{aloitus}, {opettaja}, en tiedä mihin se {kotitehtävän_tyyppi} oikein joutui.
Lemmikki{lemmikkieläimen_laji}ni {lemmikkieläimen_nimi} varmaan
{lemmikkieläimen_teko} sen!")
```

1.1 KT1.1

```
#KT1 Selitys opettajalle (Ai avustettu ratkaisu)

user_inputs = []  # Create an empty list to store all inputs

user_input = input(f"Kirjoita selityksen aloitus: ")
user_inputs.append(user_input)  # Store each input in the list

user_input = input(f"Kirjoita opettajan nimi: ")
user_inputs.append(user_input)  # Store each input in the list

user_input = input(f"Kirjoita kotitehtävän tyyppi: ")
user_inputs.append(user_input)  # Store each input in the list

user_input = input(f"Kirjoita lemmikkieläimesi laji: ")
user_inputs.append(user_input)  # Store each input in the list

user_input = input(f"Kirjoita lemmikkieläimesi nimi: ")
user_inputs.append(user_input)  # Store each input in the list

user_input = input(f"Kirjoita mitä lemmikkieläimesi teki: ")
user_inputs.append(user_input)  # Store each input in the list

# Print the list of inputs
print(f"{user_inputs[0]}, {user_inputs[1]}, en tiedä mihin se {user_inputs[2]}
oikein joutui. Lemmikki{user_inputs[3]}ni {user_inputs[4]} varmaan {user_in-
puts[5]} sen!")
```

2 KT2

String tyyppinen arvonimi kirjoitetaan manuaalisesti. Lyhenne luodaan käyttämällä viittä muuttujaa, joille annetaan arvot merkkijonon indeksoinnilla. ”lainausmerkkejä” ei lasketa mukaan indeksointiin.

```
# Arvonimi String
arvonimi = "Raivokkaan Asiallinen, Utelias Nakemyksellinen Onnistuja"

# Poimitaan sanojen ensimmäiset kirjaimet manuaalisesti
r = arvonimi[0]          # Raivokkaan
a = arvonimi[11]          # Asiallinen
u = arvonimi[23]          # Utelias
n = arvonimi[31]          # Nakemyksellinen
o = arvonimi[47]          # Onnistuja

# Generoidaan lyhenne
lyhenne = f"{r}.{a}.{u}.{n}.{o}."

print(arvonimi)
print(lyhenne)
```

3 KT3

```
#Kopioin tämän rakenteen KT1 tehtävästä ja muutin nimet kuvaavammiksi.

kilometrimäärä = float(input("Autolla ajettava vuotuinen kilometrimäärä: "))
Bensiinin_litrahinta_euroina = float(input("Bensiinin litrahinta euroina: "))
Dieselöljyn_litrahinta_euroina = float(input("Dieselöljyn litrahinta euroina "))
Bensiinimoottorisen_auton_kulutus = float(input("Bensiinimoottorisen auton
kulutus: (litraa / 100 km) "))
Dieselmoottorisen_auton_kulutus = float(input("Dieselmoottorisen auton
kulutus: (litraa / 100 km) "))
Dieselin_käyttövoimaveron_määrä = float(input("Dieselin käyttövoimaveron
määrä: "))

#Lasketaan kulutus bensalle ja dieselille.

bensa_auton_matkakulut = (kilometrimäärä *
(Bensiinimoottorisen_auton_kulutus / 100) * Bensiinin_litrahinta_euroina)
print (f"Bensalla kulkevan auton kulut: {bensa_auton_matkakulut}€")

diesel_auton_matkakulut = (kilometrimäärä * (Dieselmoottorisen_auton_kulutus
/ 100) * Dieselöljyn_litrahinta_euroina + Dieselin_käyttövoimaveron_määrä)
print (f"Dieselillä kulkevan auton kulut: {diesel_auton_matkakulut}€")

# Lasketaan ero kulutuksissa. Olisin voinut hyödyntää abs()-funktiota, joka
muuttaa luvun aina positiiviseksi, jolloin eroavaisuutta varten tarvittaisiin vain
yksi muuttuja.
bensaero =  bensa_auton_matkakulut - diesel_auton_matkakulut
dieselero = diesel_auton_matkakulut - bensa_auton_matkakulut

#Yksinkertainen if – else lause
if diesel_auton_matkakulut > bensa_auton_matkakulut:
    print (f"Bensa on {round(dieselero, 2)}€ edukkaampi.")
else:
    print (f"Diesel on {round(bensaero, 2)}€ edukkaampi.")
```

Istunnon tulostus:

Autolla ajettava vuotuinen kilometrimäärä: 15000
Bensiinin litrahinta euroina: 2.1
Dieselöljyn litrahinta euroina 1.95
Bensiinimoottorisen auton kulutus: (litraa / 100 km) 6.5
Dieselmoottorisen auton kulutus: (litraa / 100 km) 5
Dieselin käyttövoimaveron määrä: 360
Bensalla kulkevan auton kulut: 2047.5€
Dieselillä kulkevan auton kulut: 1822.5€
Diesel on 225.0€ edukkaampi.

Ajatuksia:

Kotitehtävä oli aika helppo, sekä vaati samankaltaisia ratkaisuja, kuin aiemmat kotitehtävät. Opin abs() toiminnon uutena asiana. Pythonin tapa tehdä if – else lausekkeita on todella yksinkertainen Javaan verrattuna.

4 KT4_FOR

#KT4 Mäkihypyn pisteet

Seloste ohjelman toiminnasta:
Ohjelma kysyy käyttäjältä hypyn pituuden, sekä antaa virheilmoituksen, mikäli käyttäjä syöttää aakkosia. Tuomarien_maara muuttujalla voidaan muuttaa tuomarien määrää. Tuomarien pisteet kysytään näppäimistöltä, sekä annetaan virheilmoitus, mikäli syötteessä on kirjaimia. Nämä käyttäjän syötteet tallennetaan taulukkoon myöhempää käyttöä varten pisteet_array.append(piste) toiminnolla. Yhteispisteet muuttujaan lasketaan yhteispisteet ohjeistuksen mukaisesti. Lopuksi tulostetaan hypyn pituus, yhteispisteet ja tuomarien pisteet. Käyttäjän syötteet on lihavoitu.

For piste in pisteet_array: käy pisteet_array taulukon läpi ja antaa muuttujalle “piste” arvon taulukon indeksistä. Piste tulostetaan terminaaliin yksi kerrallaan. Tämä toistetaan, kunnes taulukko on tyhjä.

Ratkaisu on toimiva, sillä tuomarien määrään muokkaus vaatii vain yhden numeron muutoksen.

#KT4 Mäkihypyn pisteet

```
# ANSI escape code for bold
BOLD = '\033[1m'
END = '\033[0m'

# Loop until acceptable HypynPituus is given
while True:
    try:
        HypynPituus = float(input("Hypyn pituus: "))
        break  # Exit the loop if successful. Dont mix it with exit() or quit()
    except ValueError:
```

```
        print("Virheellinen syöte! Kirjoita pituus numeroina.")

# Ask user input for style points. Put style points into an Array.
i = 1
pisteet_array = []
# Change the amount of judges
tuomari_maara = 5

while i <= tuomari_maara:
    try:
        piste = float(input(f"Tuomarin {i} pisteet: {BOLD}» {END}"))
        pisteet_array.append(piste)
        i += 1

    except ValueError:
        print("Virheellinen syöte! Kirjoita pisteet numeroina.")

Yhteispisteet = float(HypynPituus * (0.9) + sum(pisteet_array))

# Print everything using For command>
print(f"Hypyn pituus: {BOLD}{HypynPituus}{END}")
t = 1

for piste in pisteet_array:
    print(f"Tuomarin {t} pisteet: {BOLD}{piste}{END}")
    t += 1

print(f"Yhteispisteet: {Yhteispisteet}")
```

Käyttöesimerkki:

Hypyn pituus: abc
Virheellinen syöte! Kirjoita pituus numeroina.
Hypyn pituus: 90
Tuomarin 1 pisteet: » 11
Tuomarin 2 pisteet: » 12
Tuomarin 3 pisteet: » abc
Virheellinen syöte! Kirjoita pisteet numeroina.
Tuomarin 3 pisteet: » 13
Tuomarin 4 pisteet: » 14
Tuomarin 5 pisteet: » 15
Hypyn pituus: **90.0**
Tuomarin 1 pisteet: **11.0**
Tuomarin 2 pisteet: **12.0**
Tuomarin 3 pisteet: **13.0**
Tuomarin 4 pisteet: **14.0**
Tuomarin 5 pisteet: **15.0**
Yhteispisteet: 146.0

4.1 KT4_WHILE

#KT4 Mäkihypyn pisteet

Seloste ohjelman toiminnasta:
Ohjelma kysyy käyttäjältä hypyn pituuden, sekä antaa virheilmoituksen, mikäli käyttäjä syöttää aakkosia. Tuomarien_maara muuttujalla voidaan muuttaa tuomarien määrää. Tuomarien pisteet kysytään näppäimistöltä, sekä annetaan virheilmoitus, mikäli syötteessä on kirjaimia. Nämä käyttäjän syötteet tallennetaan taulukkoon myöhempää käyttöä varten pisteet_array.append(piste) toiminnolla. Yhteispisteet muuttujaan lasketaan yhteispisteet ohjeistuksen mukaisesti. Lopuksi tulostetaan hypyn pituus, yhteispisteet ja tuomarien pisteet. Käyttäjän syötteet on lihavoitu.

While t < len(pisteet_array) looppi tulostaa tuomarien pisteitä, niin kauan kun muuttuja t on pienempi, kuin taulukon indexien määrä. Muuttuja t alkaa arvosta 0, sekä siihen lisätään yksi jokaisella toistolla. (f"Tuomarin {t+1} pisteet: {BOLD}{pisteet_array[t]}{END}") kohdassa t muuttujaan tulee lisä +1, sillä Tuomarit alkavat ykkösestä, mutta heidän pisteiden indexit alkavat nollasta.

Ratkaisu on toimiva, sillä tuomarien määrään muokkaus vaatii vain yhden numeron muutoksen.

#KT4 Mäkihypyn pisteet

```
# ANSI escape code for bold
BOLD = '\033[1m'
END = '\033[0m'

# Loop until acceptable HypynPituus is given
while True:
    try:
        HypynPituus = float(input("Hypyn pituus: "))
        break  # Exit the loop if successful. Dont mix it with exit() or quit()
```

```
    except ValueError:
        print("Virheellinen syöte! Kirjoita pituus numeroina.")

# Ask user input for style points. Put style points into an Array.
i = 1
pisteet_array = []
# Change the amount of judges
tuomari_maara = 5

while i <= tuomari_maara:
    try:
        piste = float(input(f"Tuomarin {i} pisteet: {BOLD}» {END}"))
        pisteet_array.append(piste)
        i += 1

    except ValueError:
        print("Virheellinen syöte! Kirjoita pisteet numeroina.")

Yhteispisteet = float(HypynPituus * (0.9) + sum(pisteet_array))

# Print everything using While command>
print(f"Hypyn pituus: {BOLD}{HypynPituus}{END}")
t = 0

while t < len(pisteet_array):
    print(f"Tuomarin {t+1} pisteet: {BOLD}{pisteet_array[t]}{END}")
    t += 1

print(f"Yhteispisteet: {Yhteispisteet}")
```

Terminaalin tulostus:

Hypyn pituus: abc
Virheellinen syöte! Kirjoita pituus numeroina.
Hypyn pituus: 90
Tuomarin 1 pisteet: » 11
Tuomarin 2 pisteet: » 12
Tuomarin 3 pisteet: » abc
Virheellinen syöte! Kirjoita pisteet numeroina.
Tuomarin 3 pisteet: » 13
Tuomarin 4 pisteet: » 14
Tuomarin 5 pisteet: » 15
Hypyn pituus: **90.0**
Tuomarin 1 pisteet: **11.0**
Tuomarin 2 pisteet: **12.0**
Tuomarin 3 pisteet: **13.0**
Tuomarin 4 pisteet: **14.0**
Tuomarin 5 pisteet: **15.0**
Yhteispisteet: 146.0

Ajatuksia:
Käytin paljon aikaa yrittäessäni saada käyttäjän syötettä lihavoitua siinä vaiheessa, kun sitä kysytään HypynPituus = float(input("Hypyn pituus: ")) komennolla. Myöhemmin selvisi, ettei tämä ole mahdollista.
Arrayden käyttö on Pythonilla hyvin helppoa ja array toiminnot kuten sum(pisteet_array) ja for piste in pisteet_array: olivat hyvin helppoja sisäistää.

5 KT5 LÄMPÖTILAMUUTOS

Seloste ohjelman toiminnasta:

Ohjelma alkaa main()-funktiosta, jossa luodaan symbolit-niminen dictionary. Tämä tehdään siksi, että Celsius- ja Fahrenheit-asteikot käyttävät ° merkkiä symboleissaan, mutta Kelvin-asteikko ei käytä tätä merkkiä.
Seuraavaksi käyttäjältä kysytään lämpötila (Asteet) näppäimistöltä while-loopin avulla. Asteet-muuttuja hyväksyy vain desimaaliluvun (float), ja pyytää uutta syötettä, jos käyttäjä antaa virheellisen arvon.
Tämän jälkeen kysytään muuttujien Mista Mihin arvot käyttäjältä. Nämäkin syötteet tarkistetaan while-loopeilla niin, että käyttäjä voi syöttää vain arvot 'c', 'f' tai 'k'.
Kun kelvolliset arvot on saatu, kutsutaan muunna()-funktiota, jolle annetaan parametreiksi Asteet, Mista ja Mihin. Funktiossa on useita if- ja elif-ehtoja, jotka sisältävät laskukaavat lämpötilojen muuntamiseen asteikosta toiseen.
Tuloksena saatu arvo palautetaan return-komennolla ja tallennetaan tulos-muuttujaan.
Lopuksi ohjelma tulostaa muunnetun lämpötilan kahden desimaalin tarkkuudella, käyttäen oikeaa lämpötilasymbolia.

```
# ANSI escape code for bold
BOLD = '\033[1m'
END = '\033[0m'

def muunna(asteet, mista, mihin):
    """Converts the temperature from one scale to another."""
    if mista == mihin:
        return asteet

    if mista == "c":
        if mihin == "f":
            return asteet * 9/5 + 32
        elif mihin == "k":
            return asteet + 273.15
```

```
    elif mista == "f":
        if mihin == "c":
            return (asteet - 32) * 5/9
        elif mihin == "k":
            return (asteet - 32) * 5/9 + 273.15
    elif mista == "k":
        if mihin == "c":
            return asteet - 273.15
        elif mihin == "f":
            return (asteet - 273.15) * 9/5 + 32

def main():
    symbolit = {
        "c": "°C",
        "f": "°F",
        "k": "K"
    }

    # Loop until acceptable Asteet is given
    while True:
        try:
            Asteet = float(input("Lämpötila asteina: "))
            break
        except ValueError:
            print("Virheellinen syöte! Kirjoita lämpötila numeroina.")

    # Loop until acceptable Mista is given
    while True:
        Mista = input("Mistä asteikosta muutetaan? Syötä 'f', 'k' tai 'c': ").lower()
        if Mista in ("f", "k", "c"):
            print("Syöte hyväksytty.")
            break
        else:
            print("Virheellinen syöte!")
```

```
    # Loop until acceptable Mihin is given
    while True:
        Mihin = input("Mihin asteikkoon muutetaan? Syötä 'f', 'k' tai 'c': ").lower()
        if Mihin in ("f", "k", "c"):
            print("Syöte hyväksytty.")
            break
        else:
            print("Virheellinen syöte!")

    tulos = muunna(Asteet, Mista, Mihin)
    print(f"{BOLD}Muunnettu lämpötila: {tulos:.2f}{symbolit[Mihin]}{END}")

if __name__ == "__main__":
    main()
```

Ajatuksia:

Tehtävä oli oikein mukava ja kompakti. Ei tarvinnut kirjoittaa valtavasti koodia, mutta oppi monta uutta asiaa.

Terminaalin tulostus:

```
Lämpötila asteina: 30
Mistä asteikosta muutetaan? Syötä 'f', 'k' tai 'c': c
Syöte hyväksytty.
Mihin asteikkoon muutetaan? Syötä 'f', 'k' tai 'c': f
Syöte hyväksytty.
Muunnettu lämpötila: 86.00°F
```

6 KT6 PALINDROMIVIRKE

Seloste ohjelman toiminnasta:

Koodi määrittelee function ja listan. Tämän jälkeen suoritetaan for e in ehdokkaat: aloittaa loopin ja käy läpi ehdokkaat listan yksi kerrallaan, sekä antaa muuttujalle e arvoksi merkkijonon. Merkkijonon tulokseksi alustetaan “Ei ole” heti kättelyssä, mutta tämä tulos arvo päällekirjoitetaan, mikäli on_palindromi(e) funktio palauttaa arvoksi true.

on_palindromi funktiota kutsutaan ja sen parametriksi annetaan merkkijono yksi kerrallaan. Ensiksi määritellään tyhjä lista.
Tämän jälkeen suoritetaan for merkki in e: toistolause, joka käy läpi merkit muuttujassa e ja lisää ne pieninä kirjaimina listaan, mikäli ne ovat aakkosia hyödyntäen .append toimintoa. Muuttujalle takaperin annetaan arvoksi lause[] arrayn merkkijono käännettynä takaperin. Mikäli lause on yhtä kuin takaperin, funktio palauttaa True.
Mikäli tulos on True, if on_palindromi(e): jatkaa toimintoaan, siirtyy alemmalle riville ja päällekirjoittaa tulos muuttujan arvoksi “On”.

Tämän jälkeen tulostetaan e muuttujan merkkijono f-stringinä ja todetaan, onko se palindromi.

```
def on_palindromi(e):
    """Palauttaa True jos sana on palindromi, muutoin False."""
    lause = [] #luodaan lista, johon kerätään vain kirjaimet.
    for merkki in e:
        if merkki.isalpha(): #tarkistaa, onko merkki aakkonen. Pilkkuja ja pisteitä ei lisätä listaan.
                lause.append(merkki.lower()) #jos merkki on kirjain, se muutetaan pieneksi.

    takaperin = list(reversed(lause)) # käännetään lista takaperin.
    return lause == takaperin #vertailuoperaatio palauttaa joko true tai false.
```

```
ehdokkaat = [   #array, jossa on merkkijonoja.
    'A man, a plan, a canal: Panama.',
    'Iso rikas sika sökösakissa kirosi.',
    '"Aa, viinaa sitruksilla", kallis kurtisaani ivaa.',
    'Joku satunnainen lause', # Ei ole palindromi
    'Innostunut sonni',
    'Tynnyri',  # Ei ole palindromi
    'Aadam, Im Adam.', # Ei ole palindromi
    'Never odd or even.',
    'Was it a car or a cat I saw',
    'Mr. Owl ate my metal worm.',
    'No lemon, no melon.',
    'Evil rats on no star live'
]


#käy loopilla läpi ehdokkaat, antaen muuttujalle e yksitellen merkkijonot.
for e in ehdokkaat:
    tulos = 'EI OLE' #annetaan kaikille ehdokkaille ensiksi vastaukseksi "ei ole"
    if on_palindromi(e):
            tulos = 'ON' #jos on_palindromi palauttaa true, tulos muuttuja päällekirjoitetaan.
    print(f'"{e}": {tulos} palindromi')
```

Terminaalin tulostus:

```
$ C:/Python313/python.exe c:/Users/Esko/OneDrive/Tamk/2025/Python-ohjelmointi/TT6.py
"A man, a plan, a canal: Panama.": ON palindromi
"Iso rikas sika sökösakissa kirosi.": ON palindromi
""Aa, viinaa sitruksilla", kallis kurtisaani ivaa.": ON palindromi
"Joku satunnainen lause": EI OLE palindromi
"Innostunut sonni": ON palindromi
"Tynnyri": EI OLE palindromi
"Aadam, Im Adam.": EI OLE palindromi
```

"Never odd or even.": ON palindromi
"Was it a car or a cat I saw": ON palindromi
"Mr. Owl ate my metal worm.": ON palindromi
"No lemon, no melon.": ON palindromi
"Evil rats on no star live": ON palindromi

Ajatuksia:

Tehtävä kanssa eniten ongelmia aiheutti koodin sisennys. JavaScriptin kanssa työskentelyn jälkeen Pythonin sisennysvaatimukset aiheuttivat vähän hämmennystä. merkki.lower() voi tulevaisuudessa olla hyödyksi kirjautumistietojen kanssa, kun haluamme hyväksyä niin pienet, kuin isotkin kirjaimet.

7 KT7 PIZZATILAUS

Seloste ohjelman toiminnasta:

Ohjelma tallentaa pizzojen nimet ja täytteet Pythonin sanakirjaan, jossa pizzanumero toimii avaimena ja arvona on toinen sanakirja “nimi” ja “täytteet”.

Pizzat hinnoittelee for pizza_id in pizzat: looppissa: pizzat 1-13 maksavat 12€ ja 14-26 maksavat 13€.

Kokonaissumma alustetaan ennen kyselyä, jotta siihen voidaan lisätä arvoja scopen sisäpuolella.

Pizzan numeron kysymme while True: loopilla, sekä jatkamme kysymistä, kunnes hyväksyttävä numero syötetään. Aakkoset, negatiiviset numerot ja puuttuvat IDt hylätään.

Seuraavaksi kysytään pizzan kappalemäärä ja syöte hylätään, mikäli se on negatiivinen, nolla, yli 50 tai sisältää kirjaimia.

Jokaisen pizzan lisäämisen jälkeen muuttuja Välisumma lisätään kokonaissummaan, sekä käyttäjälle viestitään välisummasta.

Kun While true: loopista poistutaan, ohjelma tulostaa kokonaissumman.

```
pizzat = {
    1: {'nimi': 'Bolognese',
        'taytteet': ['jauheliha']},
    2: {'nimi': 'Frutti di Mare',
        'taytteet': ['ananas', 'katkarapu', 'tonnikala']},
    3: {'nimi': 'Americano',
        'taytteet': ['ananas', 'aurajuusto', 'kinkku']},
    4: {'nimi': 'Opera Special',
        'taytteet': ['kinkku', 'salami', 'tonnikala']},
    5: {'nimi': 'Mexicana',
        'taytteet': ['ananas', 'pepperoni', 'chili',
        'tacokastike']},
    6: {'nimi': 'Julia',
        'taytteet': ['ananas', 'aurajuusto', 'katkarapu',
        'kinkku']},
    7: {'nimi': 'Empire Special',
        'taytteet': ['katkarapu', 'kinkku', 'mustapippuri',
```

```
    'salami', 'sipuli', 'tuplajuusto',
    'valkosipuli']},
8: {'nimi': 'Kummisetä',
    'taytteet': ['herkkusieni', 'katkarapu', 'kinkku',
    'valkosipuli']},
9: {'nimi': 'Chicken Hawaii',
    'taytteet': ['ananas', 'aurajuusto', 'kana']},
11: {'nimi': 'Romeo',
    'taytteet': ['ananas', 'aurajuusto', 'katkarapu',
    'salami']},
12: {'nimi': 'Vegetariana',
    'taytteet': ['herkkusieni', 'oliivi', 'paprika',
    'sipuli', 'tomaatti']},
13: {'nimi': 'Dillinger',
    'taytteet': ['jauheliha', 'kinkku', 'salami',
    'sipuli']},
14: {'nimi': 'Papa''s Special',
    'taytteet': ['aurajuusto', 'mustapippuri', 'oliivi',
    'paprika', 'salami', 'sipuli']},
15: {'nimi': 'Quattro Stagioni',
    'taytteet': ['herkkusieni', 'katkarapu', 'kinkku',
    'paprika']},
16: {'nimi': 'Cambretti',
    'taytteet': ['tuplajuusto', 'katkarapu',
    'valkosipuli']},
17: {'nimi': 'Pepperoni',
    'taytteet': ['paprika', 'pepperoni', 'tonnikala']},
19: {'nimi': 'Spicy Hot Crispy',
    'taytteet': ['mausteinen naudanliha', 'sipuli',
    'tomaatti', 'chili']},
21: {'nimi': 'Finlandia',
    'taytteet': ['kana', 'katkarapu', 'kinkku',
    'salami', 'tonnikala']},
23: {'nimi': 'Driver''s Special',
    'taytteet': ['pekoni', 'pepperoni', 'kinkku',
```

```
        'salami', 'aurajuusto']},
    26: {'nimi': 'Fantasia',
        'taytteet': [None, None, None, None]}
}

for pizza_id in pizzat:
    if 1 <= pizza_id <= 13:
        pizzat[pizza_id]['hinta'] = 12
    elif 14 <= pizza_id <= 26:
        pizzat[pizza_id]['hinta'] = 13

Kokonaissumma = 0
print(f'Tervetuloa Guido’s Pizza Palaceen, mitä haluaisitte?')
#Syötä pizzan numero
while True:
    try:
        Valinta = int(input("Anna pizzan numero (0 = lopetus): "))
        if Valinta == 0:
            break
        elif Valinta not in pizzat:
            print("Virheellinen pizzan numero!")
            continue
    except ValueError:
        print("Virhe: Anna numero, ei tekstiä.")
        continue
    #Syötä määrä
    try:
        maara = int(input(f'Anna pizzan {Valinta} kappalemäärä: '))
        if maara <= 0 or maara >50 :
            print("Virheellinen määrä!")
            continue
    except ValueError:
        print("Virhe: Anna numero, ei tekstiä.")
        continue
```

```
    Välisumma = pizzat[Valinta]['hinta'] * maara
     print(f"Välisumma: {maara} x {pizzat[Valinta]['hinta']:.2f} = {Välisumma:.2f} euroa")

    Kokonaissumma += Välisumma

print(f'Kiitos tilauksestanne! Kokonaishintanne on {Kokonaissumma:.2f} euroa.')
```

Ajatuksia:

Tehtävä yllätti vaikeudellaan. If – elif - try lauseiden kasaaminen while True toistolauseen sisälle osoittautui hankaksi sisennysten kanssa. Kokonaissumma aiheutti Bugeja while True toistolauseessa, mutta sen alustaminen ennen toistolausetta korjasi ongelmat.

Terminaalin tulostus:

```
Tervetuloa Guido’s Pizza Palaceen, mitä haluaisitte?
Anna pizzan numero (0 = lopetus): abc
Virhe: Anna numero, ei tekstiä.
Anna pizzan numero (0 = lopetus): -1
Virheellinen pizzan numero!
Anna pizzan numero (0 = lopetus): 10
Virheellinen pizzan numero!
Anna pizzan numero (0 = lopetus): 12
Anna pizzan 12 kappalemäärä: abc
Virhe: Anna numero, ei tekstiä.
Anna pizzan numero (0 = lopetus): -1
Virheellinen pizzan numero!
Anna pizzan numero (0 = lopetus): 19
Anna pizzan 19 kappalemäärä: 1
Välisumma: 1 x 13.00 = 13.00 euroa
Anna pizzan numero (0 = lopetus): 12
Anna pizzan 12 kappalemäärä: 2
Välisumma: 2 x 12.00 = 24.00 euroa
```

Anna pizzan numero (0 = lopetus): 0

Kiitos tilauksestanne! Kokonaishintanne on 37.00 euroa.

8 KT8 TIETOVISA

Seloste ohjelman toiminnasta:

Tarkistetaan, että tiedoston nimi on annettu komentorivillä
(if len(sys.argv) < 2)

Määritellään tiedoston nimeksi kysymys.text komentoriviltä.
tiedosto = sys.argv[1]

Listataan tiedosto muuttujaan arvoksi kysymys.txt tiedoston sisältö.
Käydään tiedoston rivit läpi yksitellen ja lisätään niiden merkkijonot listaan rivit.
with open(tiedosto, encoding="utf-8") as f:
rivit = [rivi.rstrip("\n") for rivi in f]

Lisätään expect FileNotFoundError virheiden ehkäisyyn.
except FileNotFoundError:
print(f"Tiedostoa '{tiedosto}' ei löydy.")

Kysymykset tallennetaan listan sisälle ja käytetään laskuria muistamaan sijaintimme rivit listalla.
kysymykset = []
i = 0

Toistetaan while looppia niin pitkään kun muuttuja i suurempi kuin rivien määrä tiedostossa. Skipataan tyhjien rivien yli.
while i < len(rivit):
if rivit[i] == "":
i += 1
continue

Luetaan kaikki vaihtoehdot riviltä, kunnes saavumme ehtoon, joka lopettaa lukemisen.
while i < len(rivit) and rivit[i] != "" and not (len(rivit[i]) == 1 and rivit[i].islower()):
vaihtoehdot.append(rivit[i])
i += 1

Lukemisen lopettavat:

#Lopetamme, jos saavumme tiedoston loppuun. (i < len(rivit))

#Lopetamme, jos saavumme tyhjään riviin "". (rivit[i] != "")

#Lopetamme, jos saavumme riviin, jonka merkkijonon pituus on 1 ja se on kirjoitettu pienellä. (not (len(rivit[i]) == 1 and rivit[i].islower()):)

Mikäli vastausta ei löydy, ohjelma printtaa erroring ja lopettaa koodin ajamisen.

```
if i >= len(rivit):
    print("Tiedostossa virhe: oikea vastaus puuttuu.")
    sys.exit(1)
```

Muuttuja oikea määrittelee arvoksi vastauksen a, b tai c.

```
oikea = rivit[i]
i += 1
```

Seuraavaksi tarkistetaan, onko oikean vastauksen numero korkeampi, kuin vastausten määrä. A = 0, B = 1 etc. Hyödynnetään ASCII toimintoa.

```
if ord(oikea) - ord('a') >= len(vaihtoehdot):
print(f"Tiedostossa virhe: oikea vastaus '{oikea}' ei sovi vaihtoehtoihin {vaihtoehdot}.")
sys.exit(1)
```

Kysymys, vaihtoehdot ja oikea muuttujille on annettu arvot ja ne lisätään sanakirjana kysymykset listaan. Koodi jatkaa suorittamista, kunnes kaikki rivien määrä on pienempi kuin muuttuja i.

```
kysymykset.append({"kysymys": kysymys, "vaihtoehdot": vaihtoehdot, "oikea": oikea})
```

Tarkistetaan, että kaikissa kysymyksissä on sama määrä vaihtoehtoja.

```
vaihtoehtojen_maara = len(kysymykset[0]["vaihtoehdot"])
for k in kysymykset:
if len(k["vaihtoehdot"]) != vaihtoehtojen_maara:
 print("Tiedostossa virhe: kaikissa kysymyksissä ei ole sama määrä vaihtoehtoja")
sys.exit(1)
```

Suoritetaan pelaajan kysely. Käydään kaikki kysymykset läpi ja hyödynnetään enumerate toimintoa listaamaan listan sisällä olevien rivien Id numerot.

```
pisteet = 0
for idx, k in enumerate(kysymykset):
```

Hyödynnetään while True looppia ja käydään kaikki kysymykset läpi.
Palautetaan kysymysten aakkoset, jotka aikaisemmin muutimme numeroiksi.

```
A = 0 > 0 = A
while True:
    print(f"\n{idx+1}. {k['kysymys']}")
for j, v in enumerate(k["vaihtoehdot"]):
    print(f" {chr(ord('a')+j)}. {v}")
vastaus = input("Vastauksesi: ").strip().lower()
if vastaus not in [chr(ord('a')+j) for j in range(len(k["vaihtoehdot"]))]:
    print("Virheellinen kirjain, yritä uudelleen.")
    continue
if vastaus == k["oikea"]:
        print("Oikein!")
        pisteet += 1 # Lisää piste vastauksen mennessä oikein.
    else:
        print(f"Väärin! Oikea vastaus oli '{k['oikea']}'.")
    break
```

Tulostetaan vastaukset.

```
print(f"\nSait {pisteet}/{len(kysymykset)} oikein.")
```

```
import sys

# Tarkistetaan, että tiedoston nimi on annettu komentorivillä
if len(sys.argv) < 2:   #sys.argv sisältää tekstin, jonka kirjoitit komentoriville
    print("Käyttö: python TT8.py kysymykset.txt")
    sys.exit(1)

tiedosto = sys.argv[1] # tiedosto muuttujaan määritellään string kysymykset.txt

# Luetaan tiedosto
try: # try:execpt Python yrittää suorittaa jotain ja suorittaa expect osion, mikäli try epäonnistuu.
    with open(tiedosto, encoding="utf-8") as f: # open avaa tiedoston lukemista varten ja sulkee sen käytön jälkeen (with)
        # as f luo file objektin. kysymykset.txt on ns. kirjan nimi ja f on itse kirja.

        rivit = [rivi.rstrip("\n") for rivi in f] # for rivi in f silmukka käy läpi kaikki tiedoston rivit yksi kerrallaan.
        # rnstrip("\n" poistaa ns. uuden rivin aloittamisen. \n ei näy lukijalle, mutta on siitä huolimatta löytyy jokaisen sanan lopusta.)
except FileNotFoundError:  #FileNotFoundError on tarkkaan määritelty. Pelkkä Error ei palauta mitään.
    print(f"Tiedostoa '{tiedosto}' ei löydy.")
    sys.exit(1)

# Kysymysten käsittely
kysymykset = [] #Kysymykset tallennetaan listan sisälle
i = 0 # Laskuri, joka muistaa, millä rivillä olemme parhaillaan.

while i < len(rivit): # Toista looppia, niin pitkään, kuin i on pienempi kuin rivien määrä tiedostossa.
    if rivit[i] == "":  # Jos rivi on tyhjä, siirrymme seuraavaan riviin.
        i += 1
        continue
```

```
    kysymys = rivit[i] # Mikäli rivi ei ole tyhjä, se on kysymys.
    i += 1

    vaihtoehdot = []
    # Luetaan kaikki vaihtoehdot (rivillä >1 merkkiä)
    while i < len(rivit) and rivit[i] != "" and not (len(rivit[i]) == 1 and rivit[i].islower()):
        vaihtoehdot.append(rivit[i])
        i += 1
    #Lopetamme, jos saavumme tiedoston loppuun. (i < len(rivit))
    #Lopetamme, jos saavumme tyhjään riviin "". (rivit[i] != "")
    #Lopetamme, jos saavumme riviin, jonka merkkijonon pituus on 1 ja se on
kirjoitettu pienellä. (not (len(rivit[i]) == 1 and rivit[i].islower()):)

    if i >= len(rivit): # Mikäli vastausta ei löydy, printtaa error ja poistu ohjelmasta.
        print("Tiedostossa virhe: oikea vastaus puuttuu.")
        sys.exit(1)

    oikea = rivit[i] # Määrittelee oikea muuttujalle arvoksi vastauksen a,b tai c
kysymykset.txt tiedoston riviltä.
    i += 1

    # Tarkistus oikeasta vastauksesta
    if ord(oikea) - ord('a') >= len(vaihtoehdot): #ord(oikea) muuttaa kirjaimen
ASCII numeroksi. a = 0, b = 1, c = 2 etc.
        # ord('a') vertailee, onko vastauksen määrä enemmän, kuin vaihtoehdoja
on tarjolla.
        # Tässä tapauksessa vaihtoehtoja on 0,1,2, joten a = 0 ei aiheuta erroria.
        print(f"Tiedostossa virhe: oikea vastaus '{oikea}' ei sovi vaihtoehtoihin
{vaihtoehdot}.")
        sys.exit(1)

    #Kysymys, vaihtoehdot ja oikea muuttujille on annettu arvot ja ne lisätään
sanakirjana kysymykset listaan. Koodi palaa riville 26 ja jatkaa suorittamista.
    kysymykset.append({"kysymys": kysymys, "vaihtoehdot": vaihtoehdot,
"oikea": oikea})
```

```
# Tarkistetaan, että kaikissa kysymyksissä on sama määrä vaihtoehtoja
vaihtoehtojen_maara = len(kysymykset[0]["vaihtoehdot"]) # Lasketaan ensimmäisen kysymyksen vastausten määrä.
for k in kysymykset: # Tarkastellaan jokaista kysymystä yksitellen
    if len(k["vaihtoehdot"]) != vaihtoehtojen_maara:  # Verrataan, onko muissa kysymyksissä sama määrä vastauksia, kuin ensimmäisessä.
        print("Tiedostossa virhe: kaikissa kysymyksissä ei ole sama määrä vaihtoehtoja")
        sys.exit(1)

# Pelaajan kysely
pisteet = 0
for idx, k in enumerate(kysymykset): #Käydään kaikki kysymykset läpi. Enumerate näyttää arrayn sisällä olevan asian index numeron.
    # Mikäli enumerate poistetaan, poistetaan myös idx.
    while True: # While-True loop jatkaa kysymyistä
        print(f"\n{idx+1}. {k['kysymys']}") # \n luo välin terminaaliin ja tekee tulostuksesta nätimpää.
        for j, v in enumerate(k["vaihtoehdot"]): # Enumerate antaa indexin ja arvon. Ei toimi ilman enumerate toimintoa.
            print(f" {chr(ord('a')+j)}. {v}") # ord('a')+j antaa arvoksi a. j:n arvo on 0. jos j:n arvo on 15, arvoksi tulee p.
        vastaus = input("Vastauksesi: ").strip().lower() #.strip() poistaa kaikki välit merkkijonosta. #.lower() muuttaa isot kirjaimet pieniksi.
        if vastaus not in [chr(ord('a')+j) for j in range(len(k["vaihtoehdot"]))]: # Mikäli kirjoittamasi merkkijono ei muutu ASCII numeroksi, joka löytyy vaihtoehdoista, kysy syöte uudestaan.
            print("Virheellinen kirjain, yritä uudelleen.")
            continue
        if vastaus == k["oikea"]:
            print("Oikein!")
            pisteet += 1 # Lisää piste vastauksen mennessä oikein.
        else:
            print(f"Väärin! Oikea vastaus oli '{k['oikea']}'.")
```

```
        break

print(f"\nSait {pisteet}/{len(kysymykset)} oikein.") # Tulosta vastaukset.
```

Ajatuksia:

Tehtävä oli aika haastava, ja jouduin turvautumaan tekoälyn käyttöön melko paljon. Kävin koodin läpi rivi riviltä ja lisäsin runsaasti kommentteja varmistaakseni, että ymmärrän, mitä koodissa tapahtuu.
Pythonin ASCII-funktioneiden käyttö oli aluksi vaikeaa, enkä heti keksinyt tilannetta, jossa tulisin niitä tarvitsemaan.
Koodi on hyvin hyödyllinen monivalintatehtävien harjoitteluun tulevaisuudessa.
Sanakirjojen lisääminen listan sisälle on hyvin monikäyttäinen ratkaisu.
Oppilaan kannalta on hyvä, että se käytiin läpi näin perusteellisesti.

Terminaalin tulostus:

```
Esko@ ~/OneDrive/Tamk/2025/Python-ohjelmointi
$ python TT8.py kysymykset.txt
Tiedostossa virhe: kaikissa kysymyksissä ei ole sama määrä vaihtoehtoja
#Korjasin kysymykset.txt tiedoston

Esko@ ~/OneDrive/Tamk/2025/Python-ohjelmointi
$ python TT8.py kysymykset.txt

1. Mikä planeetta on lähimpänä Aurinkoa?
 a. Merkurius
 b. Venus
 c. Maa
 d. Mars
 e. Jupiter
Vastauksesi: abc
Virheellinen kirjain, yritä uudelleen.
```

1. Mikä planeetta on lähimpänä Aurinkoa?

a. Merkurius

b. Venus

c. Maa

d. Mars

e. Jupiter

Vastauksesi: ööö

Virheellinen kirjain, yritä uudelleen.

1. Mikä planeetta on lähimpänä Aurinkoa?

a. Merkurius

b. Venus

c. Maa

d. Mars

e. Jupiter

Vastauksesi: a

Oikein!

2. Mikä on maailman suurin nisäkäs?

a. Afrikan norsu

b. Sinivalas

c. Kirahvi

d. Makoha

e. Virtahepo

Vastauksesi: b

Oikein!

3. Kuka kirjoitti näytelmän "Romeo ja Julia"?

a. Charles Dickens

b. Jane Austen

c. William Shakespeare

d. Mark Twain

e. Leo Tolstoi

Vastauksesi: c

Oikein!

4. Mikä on Australian pääkaupunki?

a. Sydney

b. Melbourne

c. Canberra

d. Brisbane

e. Perth

Vastauksesi: 111

Virheellinen kirjain, yritä uudelleen.

4. Mikä on Australian pääkaupunki?

a. Sydney

b. Melbourne

c. Canberra

d. Brisbane

e. Perth

Vastauksesi: c

Oikein!

5. Mikä alkuaineella on kemiallinen symboli 'O'?

a. Kulta

b. Happi

c. Hopea

d. Rauta

e. Helium

Vastauksesi: b

Oikein!

6. Mikä on maailman nopein maaeläin?

a. Leijona

b. Gepardi

c. Hevonen

d. Gaselli

e. Tiikeri

d. Brisbane

e. Perth

Vastauksesi: c

Oikein!

5. Mikä alkuaineella on kemiallinen symboli 'O'?

a. Kulta

b. Happi

c. Hopea

d. Rauta

e. Helium

Vastauksesi: b

Oikein!

6. Mikä on maailman nopein maaeläin?

a. Leijona

b. Gepardi

c. Hevonen

d. Gaselli

e. Tiikeri

Oikein!

5. Mikä alkuaineella on kemiallinen symboli 'O'?

a. Kulta

b. Happi

c. Hopea

d. Rauta

e. Helium

Vastauksesi: b

Oikein!

6. Mikä on maailman nopein maaeläin?

a. Leijona

b. Gepardi

c. Hevonen

d. Gaselli
e. Tiikeri
a. Kulta
b. Happi
c. Hopea
d. Rauta
e. Helium
Vastauksesi: b
Oikein!

6. Mikä on maailman nopein maaeläin?
a. Leijona
b. Gepardi
c. Hevonen
d. Gaselli
e. Tiikeri
e. Helium
Vastauksesi: b
Oikein!

6. Mikä on maailman nopein maaeläin?
a. Leijona
b. Gepardi
c. Hevonen
d. Gaselli
e. Tiikeri
Oikein!

6. Mikä on maailman nopein maaeläin?
a. Leijona
b. Gepardi
c. Hevonen
d. Gaselli
e. Tiikeri
b. Gepardi

c. Hevonen
d. Gaselli
e. Tiikeri
Vastauksesi: b
Oikein!

7. Kuka maalasi "Mona Lisan"?
d. Gaselli
e. Tiikeri
Vastauksesi: b
Oikein!

7. Kuka maalasi "Mona Lisan"?
Vastauksesi: b
Oikein!

7. Kuka maalasi "Mona Lisan"?
a. Vincent van Gogh
b. Pablo Picasso
c. Leonardo da Vinci
d. Claude Monet
a. Vincent van Gogh
b. Pablo Picasso
c. Leonardo da Vinci
d. Claude Monet
c. Leonardo da Vinci
d. Claude Monet
e. Michelangelo
Vastauksesi: c
Oikein!
d. Claude Monet
e. Michelangelo
Vastauksesi: c
Oikein!
e. Michelangelo

Vastauksesi: c

Oikein!

Sait 7/7 oikein.

9 KT9 PIZZALUOKKA

Selostus koodin toiminnasta:

Pizzat-sanakirjaan lisätään kolme pizzan aiheisia sanaparia. Pizza-luokan konstruktoria kutsutaan, kun luodaan uusi Pizza-objekti. Pizza-objektin konstruktori alustaa nimen, täytelistan ja tyhjät arvot hinnoille.
Pizza-objektien luomisen jälkeen niitä kutsutaan erilaisilla metodeilla.
Annamme muuttujalle p arvoksi kolmannen pizza-objektin ja haemme pitsasta tietoa:

print (p.nimi)
= Tulosta pizzan nimi.

print (p.hae_taytteet())
= Hae täytelista pizza-objektin sisältä.

Seuraavaksi määrittelemme pizzoille 1-3 hinnat. Annamme aseta_hinnat metodille kolme parametria, jotka aseta_hinnat Metodi tallentaa olion sisäiseen hinnat-attribuuttiin.

print (p.hae_hinnat())
= Palauttaa tuple-muodossa kaikki kolme hintaa. (normaali, perhe, pannu)

Lopuksi tulostamme kaikki pizzat, täytteet ja hinnat terminaaliin.
for nro, pizza in pizzat.items():
= Käy toistolauseella läpi kaikki pizzat pizza-sanakirjasta. Nro toimii avaimena.
    normaali, perhe, pannu = pizza.hae_hinnat()
= pizza.hae hinnat() hakee kolme muuttujaa yhdellä haulla.
Printtaamme haetut tiedot pizza-objektista ja teemme tulostuksesta paremmin luettavan hyödyntämällä f-merkkijonoa ja :>21 muotoilua sarakkeiden luomiseksi.

```
# Pizza luokka
class Pizza:
    def __init__(self, nimi, taytteet): # Konstruktori ajetaan, kun uusi Pizza objekti luodaan.
        self.nimi = nimi
        self.taytteet = taytteet # Säilyttää taytteet listan pizza-objektin sisällä.
        self.hinnat = (0.0, 0.0, 0.0) # Alustaa hinnat erikokoisille pizzoille.

    def hae_taytteet(self):
         return '', ''.join(self.taytteet) # Palauttaa täytteet merkkijonona. .join ottaa listan ja yhdistää sen elementit, sekä laittaa pilkun elementtien väliin.

    def onko_tayte(self, t):
        return t in self.taytteet # Tarkistaa, löytyykö listalta pizza.taytteet muuttuja t ("tomaatti"). Palauttaa True tai False.

    def aseta_hinnat(self, normaali_hinta, perhe_hinta, pannu_hinta): # Parametrit saavat arvonsa kutsun parametereistä.
        self.hinnat = (float(normaali_hinta), float(perhe_hinta), float(pannu_hinta))

    def hae_hinnat(self):
        return self.hinnat


# Pizzasanakirja
pizzat = {
    1: Pizza('Margherita', ['mozzarella', 'tomaatti']),
    2: Pizza('Bolognese', ['jauhelihakastike']),
    3: Pizza('Americana', ['kinkku', 'ananas', 'aurajuusto'])
}

# Valittu pizza
p = pizzat[3]
```

```
# Tulostetaan pizzan nimi
print (p.nimi)

# Hae täytteet
print (p.hae_taytteet())

# Tarkista, löytyykö täyte
Ainesosa = "tomaatti"
print(p.onko_tayte(Ainesosa))

# Asetetaan hinnat
pizzat[1].aseta_hinnat(10.5, 18, 15)
pizzat[2].aseta_hinnat(11, 18, 15)
pizzat[3].aseta_hinnat(11, 20, 16)

# Haetaan hinnat
print (p.hae_hinnat())

# Hinnaston tulostus  :<21 luo tyhjiä välejä terminaalin tulostukseen.
print(f"{'Nro':<3} {'Nimi':<21} {'Normaali':<8} {'Perhe':<7} {'Pannu':<7}")
for nro, pizza in pizzat.items():
    normaali, perhe, pannu = pizza.hae_hinnat()
    print(f"{nro:<3} {pizza.nimi:<21} {normaali:<8.2f} {perhe:<7.2f} {pannu:<7.2f}")
    print(f"    {pizza.hae_taytteet()}")
```

Ajatuksia:

Olio-ohjelmointi oli entuudestaan tuttu viime vuoden kurssilta Hamk:issa. Tehtävässä käytetyt Metodit olivat yleiskäytännöllisiä ja helpottivat objektin sisällön hahmottamista. Tuple-palautus pizzojen hinnoista oli vähän haastava hahmottaa, mutta tähän löytyi paljon apua netistä

Terminaalin tulostus:

```
Americana
kinkku, ananas, aurajuusto
False
(11.0, 20.0, 16.0)
Nro Nimi                Normaali Perhe   Pannu
1   Margherita          10.5     18.0    15.0
    mozzarella, tomaatti
2   Bolognese           11.0     18.0    15.0
    jauhelihakastike
3   Americana           11.0     20.0    16.0
    kinkku, ananas, aurajuusto
```

10 KT10 PAIKAT JSON-MUODOSSA

Selostus koodin toiminnasta:

Luodaan paikat.py tiedosto, jossa on Paikka luokka. Luokassa on konstruktori uusien Paikka-olioiden luomiseen.

Paikkojen nimet ja kuvaukset löytyvät paikat.json tiedostosta.

TT10.py tiedostoon importataan json ja Paikka-luokka paikat.py tiedostosta.

Main-funktiossa luodaan lista paikat – johon laitetaan konstruktorilla luodut Paikka-oliot.

Open “paikat.json” , “r”

avaa paikat.json tiedoston read modessa.

“With” sulkee tiedoston automaattisesti käytön jälkeen.

data = json.load(tiedosto) lataa paikat.json tiedoston sisällön ja tallentaa sen Python-objektiksi.

```
for paikka in data:
        nimi = paikka["nimi"]
        kuvaus = paikka["kuvaus"]
        paikat.append(Paikka(nimi, kuvaus))
```

Käy läpi data-objektin toistolauseella ja lisää uuden Paikka-luokalla luodun olion paikat – listaan.

```
print()
for paikka in paikat:
      print(paikka)
```

Tulostetaan terminaaliin paikat – listan sisältö.

```
if __name__ == "__main__":
    main()
```

Kutsuu main() – funktiota, jos tiedosto ajetaan suoraan. main() – funktiota ei kutsuta, mikäli TT10.py on tuotu moduulina.

TT10.py

```
import json # Tuo json moduuli.
from paikat import Paikka # Importtaa Paikka-luokka.

def main():
    paikat = [] # Tyhjä lista, johon tallennetaan Paikka-oliot.

    #Luetaan paikat.json
    #Avataan paikat.json read modessa. "With" sulkee automaattisesti tiedoston, kun koodiblokki ollaan ajettu.
    with open("paikat.json", "r", encoding="utf-8") as tiedosto:
        data = json.load(tiedosto) # Lataa paikat.json tiedoston sisällön ja muuttaa sen Python-objektiksi.

    # Käydään toistolauseella sanakirja läpi ja luodaan Paikka-oliot.
    for paikka in data:
        nimi = paikka["nimi"]
        kuvaus = paikka["kuvaus"]
        paikat.append(Paikka(nimi, kuvaus))

    #Printataan
    print()
    print() # Formatointi
    for paikka in paikat:
        print(paikka)

if __name__ == "__main__": # Tarkistetaan, ajetaanko tiedostoa suoraa, vai onko se tuotu moduulina.
    main()
```

paikat.py

```
class Paikka: # Luokka tarjoaa ns. "mallin" oliolle
    def __init__(self, nimi: str, kuvaus: str): # Konstruktori
        self.nimi = nimi
        self.kuvaus = kuvaus

    def __str__(self):
        return f"{self.nimi}: {self.kuvaus}"
```

paikat.json

```
[
    {"nimi": "Sisäpiha", "kuvaus": "TAMKin Teiskontien puoleinen sisäpiha."},
    {"nimi": "Teiskontien aula", "kuvaus": "TAMKin Teiskontien aula."},
      {"nimi": "Palvelukatu", "kuvaus": "Palvelukatu. Tämän käytävän varrella sijaitsevat opintopalvelut."},
      {"nimi": "Tietotalon aula", "kuvaus": "Aulatila, josta pääsee C-taloon ja juhlasaliin."},
    {"nimi": "Juhlasali", "kuvaus": "TAMKin juhlasali."},
    {"nimi": "Kuntokadu aula", "kuvaus": "Kuntokadun puoleinen eteisaula."}
]
```

Terminaalin tulostus:

```
Sisäpiha: TAMKin Teiskontien puoleinen sisäpiha.
Teiskontien aula: TAMKin Teiskontien aula.
Palvelukatu: Palvelukatu. Tämän käytävän varrella sijaitsevat opintopalvelut.
Tietotalon aula: Aulatila, josta pääsee C-taloon ja juhlasaliin.
Juhlasali: TAMKin juhlasali.
Kuntokadu aula: Kuntokadun puoleinen eteisaula.
```

Ajatuksia:

Tehtävän kanssa ongelmia aiheutti TT10.py tiedoston luominen. Loin ensiksi uuden rtf – tiedoston, sekä muutin sen .py tyyppiseksi. Visual studio codessa hälytyskellot eivät soineet, mutta ohjelma ei lukenut paikat.py tiedostoa odotetusti. Ongelmat poistuivat, kun TT10.py tiedoston loi alusta alkaen Python – tiedostona.

11 HARJOITUSTYÖ

SUUNNITELMA

Pelin suunnittelu perustuu **tekstipohjaiseen seikkailupeliin**, jossa pelaaja liikkuu eri huoneiden ja alueiden välillä, kerää esineitä, keskustelee NPC-hahmojen kanssa ja taistelee tai kerää Pokémon-hahmoja. Pelin keskeiset käsitteet ovat **huoneet, objektit, Pokémonit ja pelaaja**.

1. Huoneet (Room-luokka)

- Jokainen huone on **erillinen olio**, joka sisältää:
    - Nimen ja kuvauksen
    - Listan objekteista (objects), joita pelaaja voi tarkastella tai käyttää
    - Sanakirjan exits, joka yhdistää huoneen muihin huoneisiin suunnan avulla
- Huoneet yhdistetään toisiinsa metodilla connect(direction, room), jolloin pelaaja voi liikkua eteen- tai taaksepäin.

2. Pelaaja (Player-luokka)

- Pelaajalla on:
    - current_room: viittaus nykyiseen huoneeseen
    - inventory: lista esineistä, joita pelaaja kantaa
    - party: lista Pokémon-hahmoista (max 6)
- Pelaaja liikkuu huoneiden välillä metodilla move(direction), joka tarkistaa huoneen exits-sanakirjasta, onko liike sallittu, ja mahdolliset **vartijat** (guard) estämässä etenemistä.

3. Esineet ja objektit (GameObject, Item, HeavyItem)

- Kaikki pelin objektit periytyvät luokasta GameObject, jolla on:
    - Nimi, kuvaus, mahdollisuus ottaa mukaan (takeable) ja keskustella (talkable)
- Item on otettavissa oleva esine, esim. potion tai bait
- HeavyItem sisältää **toimintoja** (triggers) kuten "hit", "pull", "use", jotka voivat laukaista skriptejä pelaajan tekemien toimintojen perusteella

4. NPC-hahmot (NPC-luokka)

- NPC:llä on:
    - Nimi, kuvaus
    - Lista dialogirivejä, joissa voi olla **trigger-funktio** (esim. antaa potion tai avata polku)
    - repeatable-attribuutti, joka määrittää, voiko dialogia toistaa

5. Pokémon ja liikkeet (Pokemon, Move)

- Pokémonilla on:
    - Nimi, tyyppi, elämäpisteet (health) ja liikkeet
    - __str__()-metodi, joka muuntaa Pokémonin luettavaksi tekstiksi
- Move-luokka sisältää yksittäisen liikkeen nimen ja tyypin

6. Pulmat ja vuorovaikutukset

- Esineisiin ja Pokémoniin voidaan liittää **pulma-tiloja** tai trigger-funktioita:
    - Esim. Poliwag-pond puzzle, jossa pelaajan täytyy käyttää baitia Pokémonin houkuttelemiseksi
    - Sandshrew on jumissa ja pelaajan täytyy vetää (pull) sandshrew sen vapauttamiseksi
- Tämä toteutetaan funktioina triggers.py, joita kutsutaan HeavyItem- tai NPC-olioiden yhteydessä

Pythonin ominaisuuksien käyttö

Pelin toteutuksessa hyödynnettiin seuraavia Python-ominaisuuksia:

1. **Olio-ohjelmointi (OOP)**:
    - Luokat ja periytyminen (GameObject → Item, HeavyItem, Pokemon, NPC)
    - Instanssit huoneista, pelaajasta ja esineistä
    - Metodit objektien vuorovaikutuksille

2. **Sanakirjat ja listat**:
    - Huoneiden exits on sanakirja, jonka avaimet ovat suuntia (north, south, east, west)

- Pelaajan inventaario ja Pokémon-party ovat listoja
- Triggers on sanakirja ({"pull": pull_sandscrew_trigger})

3. **Funktiot ja korkeamman asteen funktiot**:
   - Trigger-funktiot NPC:llä tai HeavyItemillä
   - Mahdollistaa pelin tapahtumien käsittelyn dynaamisesti

4. **String-muokkaus ja värit**:
   - ANSI-koodit (\033[32m) käytetty Pokémonien ja tärkeiden objektien värjäämiseen

5. **Iteraatiot ja comprehensions**:
   - Etsitään objekteja huoneesta tai inventariosta
   - Etsitään Pokémon tiettyjen ehtojen perusteella (next((p for p in player.current_room.objects if ...)))

6. **Modulaarisuus**:
   - Eri osat jaetaan tiedostoihin:
     - game_objects.py (luokat)
     - triggers.py (pulmat ja tapahtumat)
     - guards.py (vartijat)
     - world.py (maailman rakentaminen)
   - Tämä helpottaa ylläpitoa ja laajentamista

PELIN KUVAUS

Pelin tarkoitus on seikkailla Pokemon-maailmassa, kerätä uusia pokemoneja ja haastaa muita kouluttajia Pokemon-otteluihin.
Peli perustuu vanhoihin Pokemon-peleihin, jossa päähenkilö aloittaa seikkailunsa Kanto-alueelta ja haastaa pokemon-saleja.

Pelin tarkoitus on seikkailla Pokémon-maailmassa, kerätä uusia Pokémoneja ja haastaa Brock Pokémon-otteluun.. Peli perustuu klassisiin Pokémon-peleihin, joissa päähenkilö aloittaa seikkailunsa Kanto-alueelta, tarkemmin aloituskylästä. Pelaaja liikkuu eri kaupungeissa, kylissä ja reiteillä, etsien uusia Pokémoneja ja esineitä, jotka auttavat otteluissa. Matkan varrella voi suorittaa puzzleja ja keskustella muiden hahmojen kanssa.
Pelissä on mahdollista kerätä useita Pokémoneja ja haastaa esimerkiksi Brock Pewter Cityn Pokémon-salilla. Otteluissa pelaaja valitsee millä liikkeellä hyökkää. Pelin päätavoite on voittaa Brock ja saada häneltä Boulderbadge.
Lisäksi pelaajan tavoitteena on kerätä kaikki saatavilla olevat Pokémoneit – tämä toimii pelin pistelaskutapana (Gotta catch them all!). Pelissä on myös strategisia elementtejä, kuten erilaisten Pokémon-tyyppien heikkoudet sekä liikekohtaiset valinnat otteluissa.

WALKTROUGH

Look around
Head down
Talk to mom x3
Go east
Walk east
Speak to professor oak x 4
Select your starter Pokemon
Charmander
Go west
Head north
Look around
Hit/Kick/Smack tree
Caterpie fell down!
Catch/pokeball Caterpie
Head west
Look around
Pull Sandshrew
Catch/pokeball Sandshrew
Head north

Look around
Pick rope
A wild Growlithe appeared!
Pokeball Growlithe
Pick rope
Pick rope
Go west
Head north
Look around
Pick berry
Go south
Go east
Use crafting table
Rope + Berry / Berry + Rope
YOU CREATED BAIT!
Walk west
Walk south
Look around
Catch Poliwag (fails)
Drop bait
Pull bait
Catch Poliwag
Head west
Head west
Talk to Ranger
Capture Pikatchu
Head north
Head north
Head west
Talk to Brock x 4
1 (Chosen skill number)
1 (Chosen skill number)
2 (Chosen skill number)
2 (Chosen skill number)

YOU RECIEVED THE BOULDERBADGE!

MAP LAYOUT