2. Nekaj osnovnih operacij v Pythonu, Objekti, Tipi, Razredi, Nizi

V Pythonu lahko realiziramo vse elemente srukturiranega programiranja.

2.1. **Zaporedje**. V Pythonu preprosto realiziramo zaporedje. Stavke pišemo vsakega v svojo vrstico.

2.2. **Spremenljivke in prireditveni stavek**
Vrednost lahko shranimo v spremenljivko s prireditvenim stavkom:

- `a = 5`
- `obseg = 3.548`

Spremenljivke lahko uporabimo pri izračunu izrazov:

- `a = 5`
- `b = 6`
- `ploscina = a * b`

Če bi kasneje spremenili vrednost spremenljivke *a*, to ne bi imelo vpliva na vrednost spremenljivke *ploscina*. Pogosti so prireditveni stavki, kjer novo vrednost spremenljivke izračunamo iz njene stare vrednosti. S stavkom

- `a = a + 5`

smo stari vrednosti spremenljivke *a* prišteli 5 in dobljeno vrednost shranili nazaj v spremenljivko *a*.

2.3. **Odločitev in enostavni stavek if.** (Bomo spoznali v eni naslednjih lekcij.)

2.4. **Zanka in zanka while.** (Bomo spoznali v eni naslednjih lekcij.)

2.5. **Stavek def in return**. (Bomo spoznali v eni naslednjih lekcij.)

Branje in pisanje (vhod in izhod) v Pythonu.

2.6. Funkcija `input`
Funkcija input izpiše "prompt" in vrne niz, ki ga vtipkamo.

- `niz = input(prompt)`

Zgled: `niz = input("Kako ti je ime? ")`

Če želimo prebrati cello število lahko vtipkani niz spremenimo v število z vgrajeno funkcijo int.

Zgled: `n = int(input(“Koliko je n? ”))`

Če se med tekom programa pokaže na zaslonu vprašanje:
`Koliko je n?`
in vtipkamo dogovor 17, bo spremenljivka n dobila vrednost 17.

2.7 Funkcija (stavek) `print`

Do različice 3 je bil print stavek, zdaj pa je funkcija. To pomeni, da moramo uporabljati oklepaje.

- Zdaj: `print(n)`
- Včasih: `print n`

Zgled:

```
niz = input("Kako ti je ime? ")
print("Ime mi je ",niz)
```

2.8 Izrazi: Python kot računalo

Osnovni aritmetični operatorji:

- + (vsota)
- - (razlika)
- * (produkt)
- ** (potenca)
- / (kvocient)
- // (celoštevilski kvocient)
- % (ostanek pri deljenju)

Zgled: Izračunajmo dolžino hipotenuze c, če sta znani kateti a in b.

```
a = 3
b = 4
c = (a**2 + b**2)**(1/2)
```

```
Zgled: Napišite del programa, ki zamenja desetice in
stotice števila n.

n = 2012
d = (n//10)%10
s = (n//100)%10
nn = n - 10*d - 100*s + 100*d + 10*s
print(n,nn)
```

Zgled: Izpišimo kvadratni koren števila 3 na 6 decimalnih mest.

```
# kvadratni koren na k decimalk
x = 3 # kvadratni koren iz x
k = 6  # na k decimalnih mest.
a = x**(1/2)
p = 10**k
b = a*p
c = int(b)
d = c/p
print(x,d)
```

To se je izšlo z rezanjem decimal. Zdaj pa si poglejmo malo bolj splošno rešitev, ki deluje tudi z zaokrožanjem. Razliko med rezanjem in zaokrožanjem vidimo npr. pri kvadratnem korenu števila 2 na 7 decimalnih mest.

```
print("Števila in njihovi kvadratni koreni")
print("Zaokrožanje decimalk")
n = int(input("n = ? "))
k = int(input("Število decimalnih mest? "))
kv = n**(1/2)
kvnovi = int(kv * 10**k)/10**k
kvnoviz = int(0.5 + kv * 10**k)/10**k
print("sqrt(",n,") =",kv)
print("na ",k,"decimalk z rezanjem pa", kvnovi)
print("na ",k,"decimalk z zaokrožanjem pa", kvnoviz)
```

Ostane pa problem izpisa končnih ničel. Ta problem bomo najlažje rešili, ko bomo spoznali metodo `format.`

2.9. Vgrajene funkcije:

- `abs(x)` - absolutna vrednost števila x
- `int(x)` - celi del števila x
- `min(a, b, c, ...)` - najmanjša vrednost
- `max(a, b, c, ...)` - največja vrednost

Zgled: Napišite del programa, ki za naravno število `n` poišče število `m` tako, da je `m**2 <= n < (m+1)**2` in pove, ali je `n` popolni kvadrat.

```
print("Ali je dano število n popolni kvadrat?")
n = int(input("Vtipkajte n " ))
m = int(n**(1/2))
m2 = m*m
m21 = (m+1)*(m+1)
print("Števili ",m2," in ", m21, "sta popolna kvadrata in
")
print(m2,"<=",n,"<",m21)
print("Kvadrat števila ",m," je ",m2)
print("Tole, kar piše spodaj je",n==m2)
print("Kvadrat števila ",m, "je ",n)
```

2.10. Okrajšave

Tak stavek lahko zapišemo krajše

- `a += 5`

Podobne okrajšave pozna Python tudi za druge aritmetične operatorje:

`a = a + 3`	`a += 3`
`a = a - 3`	`a -= 3`

`a = a * 3`	`a *= 3`
`a = a ** 3`	`a **= 3`
`a = a / 3`	`a /= 3`
`a = a // 3`	`a //= 3`
`a = a % 3`	`a %= 3`

2.11. Kompleksna števila

Kompleksno število zapišemo na enega izmed spodnjih načinov:

- `2 + 3j`
- `complex(2, 3)`

Če imamo kompleksno število shranjeno v spremenljivki *z*, potem je

- `z.real` - realni del kompleksnega števila
- `z.imag` - imaginarni del kompleksnega števila

2.12. Naloge

1. Sestavite del programa, ki izračuna in izpiše desetice celega števila (števka, ki je na predzadnjem mestu).
2. Sestavite del programa, ki izračuna in izpiše vrednost 1, če je podatek med 0 in 1, sicer pa naj bo vrednost 0.
3. Sestavite del programa, ki izračuna in izpiše obrat tromestnega celega števila.
4. Sestavi del programa, ki dano dolžino v yardih pretvori v metre, decimetre in centimetre. Ta razultat naj tudi izpiše.

Rešitve nalog

Vsako od teh nalog bomo rešili na dva načina:
a) Podatek bomo zapisali v program
b) Podatek bomo interaktivno prebrali iz tipkovnice med izvajanjem programa.

Kasneje bomo spoznali še tretji način, ki bo najbolj primeran, vendar moramo počakati, da bolje spoznamo stavka "def" in "return".

1a)

```
stevilo = 123456
stevka = stevilo//10%10
print("Desetice stevila ", stevilo," je stevka ", stevka)
```

1b)

```
stevilo = int(input("vtipkaj stevilo: "))
stevka = stevilo//10%10
print("Desetice stevila ", stevilo," je stevka ", stevka)
```

3a)

```
stevilo = 345
enica = stevilo%10
desetica = stevilo//10%10
stotica = stevilo//100%10
obrat = 100*enica + 10*desetica + stotica
print("Obrat stevila ", stevilo," je stevilo ", obrat)
```

3b)

```
stevilo = int(input("vtipkaj stevilo: "))
enica = stevilo%10
desetica = stevilo//10%10
stotica = stevilo//100%10
obrat = 100*enica + 10*desetica + stotica
print("Obrat stevila ", stevilo," je stevilo ", obrat)
```

4a)

```
y = 77
yard = 0.9144
dolzina = y * yard + 0.005
m = int(dolzina)
dolzina = 10 * (dolzina - m)
dm = int(dolzina)
dolzina = 10 * (dolzina - dm)
cm = int(dolzina)
print(y, 'yardov =', m, 'm', dm, 'dm', cm, 'cm')
```

4b)

```
y = int(input("vtipkaj dolzino v yardih: "))
yard = 0.9144
dolzina = y * yard + 0.005
m = int(dolzina)
dolzina = 10 * (dolzina - m)
dm = int(dolzina)
dolzina = 10 * (dolzina - dm)
```

```
cm = int(dolzina)
print(y, 'yardov =', m, 'm', dm, 'dm', cm, 'cm')
```

2.13. Objekti, Tipi, Razredi, Nizi

Vsak program deluje nad podatki, nad rečmi. V jeziku Python je vsaka reč objekt. V prvi aproksimaciji si lahko predstavljamo, da ima vsak objekt svoj tip, vse objekte danega tipa pa združimo v razred.

Tako sodi npr. 1 v razred `int`, 1.0 v razred `float`, 1 + 1j pa v razred `complex`.

Objekti danega tipa sestavljajo nekakšno množico, ki ji rečemo razred. Tip pa je ime razreda. Nekatere podatkovne tipe smo že spoznali. Običajno Python kar ugane tip spremenljivke. Npr.

a = 5

povzroči, da postane a celoštevilska spremenljivka. Tako za tem lahko zapišemo

a = "teta"

in a se prelevi v niz.

2.14. Funkcija `type`.

Zato je koristno, da lahko zvemo, kakšen je trenutni tip spremenljivke. oz. objekta. Tip objekta lahko zvemo s funkcijo `type`.

2.15. Nespremenljivi tipi

Podatkovni tipi so nespremememljivi ali spremenljivi. Vrednost podatka spremenljivega tipa se lahko med računanjem spreminja, vrednost podatka nespremenljivega tipa pa se ne more med računanjem spreminjati. Zdaj bomo zapisali vse nespremenljive tipe, ki jih premore Python.

2.16. Razred `bool`.

`True` in `False` sestavljata razred `bool`.

2.17. Število

Celo število - `int`
Realno število - `float`
Kompleksno število - `complex`

2.18. Niz - `str`

Ta pomembni razred si bomo pogledali podrobno malo kasneje.

2.19. Nabor - `tuple`

Nabor dobimo tako, a zapišemo več vrednosti, ločenih z vejico. Uporabljali ga bomo takorekoč mimogrede.

2.20. None - `NoneType`

Funkcija, ki ne vrne vrednosti, npr. print() vrne prazno vrednost None praznega tipa `NoneType`.

2.21. Atributi in metode

Vsak objekt nosi s seboj **atribute** (lastnosti) in **metode**. V prvi aproksimaciji si lahko atribute predstavljamo kot vrednosti. Tako ima kompleksno število atributa `real` in `imag`. Metoda pa je funkcija, ki je vezana na objekt. Do lastnosti objekta s pridemo tako, da za objektom zapišemo piko (.) in ime atributa. Če pa gre za metodo, zapišemo še v oklepajih vrednosti parametrov. Pri tem nam parametra, ki se nanaša na objekt sam ni potrebno pisati.

```
s.atribut

s.metoda()

s.metoda(x,y)
```

2.22. Zgled.

Npr. kompleksno število `complex` ima atributa `real` in `imag` ter metodo `conjugate()`.

Naj bo `z = 2 - 3j`, tedaj je `z.real` je `2.0` in realna komponenta kompleksnega števila. `z.imag` je `-3.0` ter je imaginarna komponenta kompleksnega števila. `z.conjugate()` je metoda, ki vrne `2 + 3j`.

Razliko med čistim atributiom in metodo vidimo pri oklepajih.

2.23. Ukaz `dir`.

Z ukazom `dir(z)` dobimo seznam (glej eno od naslednjih lekcij) vseh atributov ter metod, ki pripadajo objektu z danega tipa.

2.24. Nizi

- Nizi so zaporedja znakov. Zapišemo jih v enojnih ali dvojnih narekovajih: `'beseda'`, `"beseda"`

- Posebni znaki: **`\n`** (nova vrsta), **`\t`** (tabulator), **`\\\\`** (nagibnica), **`\\'`** (enojni narekovaj), **`\"`** (dvojni narekovaj), ...
- Dolgi nizi: zapišemo jih lahko čez več vrstic, pri čemer vsako vrstico zaključimo z znakom **`\`**
- Večvrstični nizi: zapišemo jih v trikratnih narekovajih (enojnih ali dvojnih): **`'''dolg niz'''`**, **`"""dolg niz"""`**
- Prazen niz: **`''`**, **`""`**
- Dolžina niza je število znakov v nizu: **`len(s)`**

2.25. Operacije na nizih

Nizi se obnašajo podobno kot seznami znakov, le da moramo paziti na nekatere omejitve. Niza ne moremo spremeniti. To pomeni, da ne moremo spremeniti ali odstraniti posameznega znaka ali podniza. V takem primeru moramo narediti nov niz brez izbranega poniza. Sezname bomo spoznali kasneje, vendar se jim v tem poglavju ne moremo popolnoma izogniti.

2.26. Znaki, indeksi in podnizi - rezine.

- **`s[i]`**
  - i-ti znak niza s.
  - Znaki so indeksirani od 0 naprej.
  - Če je indeks i negativen, štejemo od konca niza proti začetku
- **`s[i:j]`**
  - Podniz od i-tega do j-tega znaka (j-ti znak ni več zraven). Če spustimo i, se podniz prične na začetku, če spustimo j, pa konča na koncu.
  - **`s[i:j]`** je podniz, sestavljen iz znakov s_i, s_{i+1}, ..., s_{j-1}
  - **`s[:j]`** je podniz, sestavljen iz znakov s_0, s_1, ..., s_{j-1}
  - **`s[i:]`** je podniz, sestavljen iz znakov s_i, s_{i+1}, ..., s_{n-1}, kjer je n število znakov v nizu s
  - **`s[:]`** je podniz, sestavljen iz znakov s_0, s_1, ..., s_{n-1}, kjer je n število znakov v nizu s
- **`a[i:j:k]`**
  - Podniz od i-tega do j-tega znaka (j-ti znak ni več zraven), pri čemer vzamemo vsak k-ti znak
  - **`s[i:j:k]`** je podniz, sestavljen iz znakov s_i, s_{i+k}, s_{i+2k}, ... do j-tega znaka

2.27. Stikanje

- **`s + t`**
  - Novi niz, sestavljen iz znakov niza s, ki jim sledijo znaki niza t.
- **`s += t`**
  - Podobno kot prej, le da se novi niz shrani nazaj v s
- **`n * s`**
  - Novi niz, ki je enak stiku n kopij niza s (s + s + ... + s)
- **`s *= n`**
  - Podobno kot prej, le da novi niz shrani nazaj v s

2.28. Primerjanje

- Operatorji **`<`**, **`>`**, **`<=`**, **`>=`**, **`==`** in **`!=`**
  - Dva niza primerjamo leksikografsko (kot so urejene besede v leksikonih)
- Operatorja **`in`** in **`not in`**
  - Preverimo, ali se prvi niz nahaja kot podniz v drugem nizu.

- Operatorja **`is`** in **`is not`**
    - Preverimo, ali sta niza isti/različen objekt

2.29. Funkcije na nizih:

- **Vrne število:**
- **`len(s)`** ... število znakov niza s
- **`min(s)`** ... najmanjši znak niza s
- **`max(s)`** ... največji znak niza s
- **`ord(s)`** ... vrne kodo edinega znaka niza s
- **`int(s)`** ... vrne celo število, zapisano v nizu s
- **`int(s, b)`** ... vrne celo število, zapisano v nizu s pri osnovi b
- **Vrne niz:**
    - **`chr(n)`** ... vrne niz, ki vsebuje znak s kodo n
    - **`str(n)`** ... vrne niz števk števila n.
- **Vrne seznam (*):**
    - **`list(s)`** ... vrne seznam znakov niza s (sezname bomo spoznali v eni naslednji lekcij.

2.30. Naloge

Naslednje naloge lahko rešujemo interaktivno v okolju idle.

1. Oblikujte niz sestavljen iz znakov + in o v obliki šahovnice.
2. V spremenljivko niz shranite poljuben niz, npr. niz = "teta". Z enim ukazom v niz `noviniz` shranite znake prvotnega niza, tako da so na začetku prvi, tretji, ... znaki niza niz, potem pa drugi, četrti, ... znaki niza niz. Npr. v našem primeru mora dobiti noviniz vrednost "ttea".
3. Kakšen je tip niza? Uporabite funkcijo `type`.
4. V naravnem številu n poiščite največjo števko. Pri tem si lahko pomagate z nizi.

Rešitve:

1. primer

```
c = "+"
b = "o"
br = "\n"
cv = 4*(c+b)
bv = 4*(b+c)
sah = 4*(cv+br+bv+br)
print(sah)
```

2.primer

```
niz = "teta"
noviniz = niz[0::2]+niz[1::2]
print(noviniz)
```

2.31. Nekatere metode na nizih

- **Vrne število**
  - **`s.count(t)`** ... vrne število, kolikokrat se niz t pojavi kot podniz (brez prekrivanj) v nizu s
  - **`s.count(t, i)`** ... podobno kot prej, le da začne šteti pri indeksu i
  - **`s.count(t, i, j)`** ... podobno kot prej, le da konča šteti pri indeksu j
  - **`s.find(t)`** ... vrne najmanjši indeks v nizu s, kjer se niz t pojavi kot podniz (vrne -1, če ga ne najde)
  - **`s.find(t, i)`** ... podobno kot prej, le da začne iskati pri indeksu i
  - **`s.find(t, i, j)`** ... podobno kot prej, le da konča z iskanjem pri indeksu j
  - **`s.index(...)`** ... podobno kot **`s.find(...)`**, le da javi napako, če podniza ne najde
- **Vrne niz**
  - **`s.join(sez)`** ... vrne niz, ki je sestavljen iz nizov iz seznama sez. Kot ločilo med vrednostmi uporabi niz s
  - **`s.replace(t1, t2)`** ... vrne kopijo niza s, kjer vse pojavitve podniza t1 zamenja s podnizi t2
  - **`s.replace(t1, t2, n)`** ... podobno kot prej, le da zamenja samo prvih n pojavitev podniza t1
  - **`s.strip()`** ... vrne kopijo niza s, kjer so na začetku in na koncu odstranjeni vsi beli znaki
  - **`s.strip(t)`** ... vrne kopijo niza s, kjer so na začetku in na koncu odstranjeni vsi znaki iz niza t
  - **`s.format(...)`** ... vrne kopijo niza s, preoblikovanega v skladu z zahtevami
  - **`s.lower()`** ... vrne kopijo niza s, kjer so vse črke predelane v male črke
  - **`s.upper()`** ... vrne kopijo niza s, kjer so vse črke predelane v velike črke
  - **`s.title()`** ... vrne kopijo niza s, kjer se vse besede pričnejo z veliko začetnico, nadaljujejo pa z malimi črkami
  - **`s.capitalize()`** ... vrne kopijo niza s, kjer je prvi znak predelan v veliko črko
  - **`s.swapcase()`** ... vrne kopijo niza s, kjer so vse male črke predelane v velike in obratno
  - **`s.ljust(n)`** ... vrne kopijo niza s zapisano levo poravnano v nizu dolžine n
  - **`s.ljust(n, znak)`** ... podobno kot prej, le da dodatne presledke na koncu nadomesti z danim znakom
  - **`s.rjust(n)`** ... vrne kopijo niza s zapisano desno poravnano v nizu dolžine n
  - **`s.rjust(n, znak)`** ... podobno kot prej, le da dodatne presledke na začetku nadomesti z danim znakom
  - **`s.center(n)`** ... vrne kopijo niza s zapisano sredinsko poravnano v nizu dolžine n
  - **`s.center(n, znak)`** ... podobno kot prej, le da dodatne presledke na začetku in koncu nadomesti z danim znakom

- **Vrne seznam(*)**
    - **`s.split()`** ... vrne seznam besed v nizu s (besede so ločene z enim ali več belimi znaki)
    - **`s.split(None, n)`** ... vrne seznam prvih n besed v nizu s, skupaj z nizom, ki vsebuje vse ostale besede
    - **`s.split(t)`** ... vrne seznam besed v nizu s (pri tem so besede ločene s podnizom t)
    - **`s.split(t, n)`** ... vrne seznam prvih n besed v nizu s, skupaj z nizom, ki vsebuje vse ostale besede (pri tem so besede ločene s podnizom t)
- **Vrne logično vrednost (predikat)**
- **`s.isdigit()`** ... vrne True, če niz s vsebuje vsaj en znak in so vsi znaki številke
- **`s.isalpha()`** ... vrne True, če niz s vsebuje vsaj en znak in so vsi znaki črke
- **`s.islower()`** ... vrne True, če niz s vsebuje vsaj en znak in so vsi znaki male črke
- **`s.isupper()`** ... vrne True, če niz s vsebuje vsaj en znak in so vsi znaki velike črke
- **`s.isalnum()`** ... vrne True, če niz s vsebuje vsaj en znak in so vsi znaki alfanumerični (črke ali številke)
- **`s.isspace()`** ... vrne True, če niz s vsebuje vsaj en znak in so vsi znaki beli (presledki, tabulatorji, nove vrste)

2.32. Naloge

Naslednje naloge bomo rešili na več načinov. Najprej bomo uporabili le doslej pridobljeno znanje jezika Python. Ko pa spoznamo stavek **for** za nize, jih bomo rešili še enkrat.

1. Sestavite funkcijo, ki prešteje, koliko samoglasnikov je v nizu.
2. Sestavite funkcijo, ki prešteje, koliko besed je v nizu. Besede so zaporedja poljubnih znakov, ki so med seboj ločena z enim ali več presledki.
3. Sestavite funkcijo, ki preveri, ali je dani niz palindrom.
4. Sestavite funkcijo, ki iz danega niza izloči vse pojavitve prvega znaka.
5. Sestavite funkcijo, ki ugotovi, ali so vsi znaki danega niza med seboj enaki.
6. Sestavite funkcijo, ki za dano število n vrne niz dolžine n, sestavljen iz samih presledkov.
7. Sestavite funkcijo, ki za dani niz s in dano črko c angleške abecede vrne število pojavitev te črke v nizu (ne glede na to, ali se pojavlja kot velika ali mala črka.)

Rešitve:
1. primer

```
niz = "slovenija"
nsamog = niz.count("a") + niz.count("e") + niz.count("i") + niz.count("o") +
niz.count("u")
print(nsamog)
```