Funksionet

1. Sintaksa Bazë

Përcaktimi: Përdorni fjalën kyçe def, emrin e funksionit, parametrat në kllapa (), dhe dy pika :.

Trupi: Bllok kodi i identuar poshtë përcaktimit.

Thirrja: Përdorni emrin e funksionit me kllapa dhe argumente.

```
def pershendetje(emri):  # Përcakton një funksion
    print(f"Përshëndetje, {emri}!")

pershendetje("Ana")  # Thirr funksionin → Rezultati: Përshëndetje,
Ana!
```

2. Parametrat vs. Argumentet

Parametrat: Variablat e listuara në përcaktimin e funksionit.

Argumentet: Vlerat aktuale të dërguara gjatë thirrjes.

```
def mbledh(a, b):  # a dhe b janë parametra
    return a + b

rezultat = mbledh(2, 3)  # 2 dhe 3 janë argumente
print(rezultat)  # Rezultati: 5
```

3. Kthimi i Vlerave

Përdorni return për të kthyer një vlerë. Nëse mungon, funksioni kthen None.

```
def katrori(n):
    return n ** 2

print(katrori(4))  # Rezultati: 16
```

4. Parametra Default

Caktoni vlera default për parametra. Argumentet me default janë opsionale.

```
def fuqia(baza, eksponent=2):
    return baza ** eksponent

print(fuqia(3))     # Rezultati: 9 (përdor eksponent=2 si default)
print(fuqia(3, 4))  # Rezultati: 81
```

5. Argumente me Gjatësi të Ndryshueshme

*args: Merr një numër të ndryshueshëm argumentesh pozicionale si tuple.

**kwargs: Merr argumente me emër (keyword) si fjalor.

Për ta qartësuar këtë koncept, duhet kuptuar koncepti i argumenteve pozicionale, dhe atyre me fjalë kyçe

```
def shfaq_argumentet(*args, **kwargs):
    print("Pozicionale:", args)
    print("Me emër:", kwargs)

shfaq_argumentet(1, 2, emri="Ana", mosha=25)
```

6. Argumente me Gjatësi të Ndryshueshme

Specifikoni argumentet me emrin e parametrit për qartësi.

```
def pershkruaj_kafshe(emri, kafsha="qen"):
    print(f"{emri} është një {kafsha}.")

pershkruaj_kafshe(kafsha="mace", emri="Mia")  # Argumente me emër
```

7. Fushëveprimi (Scope)

Lokale: Variablat brenda funksionit.

Globale : Variabla jashtë funksioneve. Përdor global për t'i modifikuar.

```
x = 10  # Variabël globale

def ndrysho_x():
    global x
    x = 20  # Modifikon x globale

ndrysho_x()
print(x)  # Rezultati: 20
```

8. Funksionet Lambda

Funksione anonime me një shprehje, të përcaktuara me lambda.

Një funksion anonim nuk ka "emër", dhe deklarohet duke iu caktuar një variable

```
katrori = lambda x: x ** 2
print(katrori(5))  # Rezultati: 25
```

9. Rekursioni

Funksionet mund të thërrasin vetveten. Siguro një kusht bazë për të shmangur loop-et e pafundme.

```python
def faktorial(n):
    return 1 if n == 0 else n * faktorial(n-1)

print(faktorial(5))  # Rezultati: 120
```

10. Funksionet si Objekte

Funksionet janë objekte. Mund t'ju caktohen variablave, të kalohen si argumente ose të kthehen.

```python
def bertit(text):
    return text.upper()

def pershendet(func, mesazhi):
    return func(mesazhi)

print(pershendet(bertit, "Tungjatjeta"))  # Rezultati: TUNGJATJETA
```

11. Dekoratorët

Modifikoni funksione me sintaksën @decorator.

```python
def debug(func):
    def mbeshtjelles(*args, **kwargs):
        print(f"Thirrja e {func.__name__}")
        return func(*args, **kwargs)
    return mbeshtjelles

@debug
def thuaj_ciao():
    print("Ciao!")

thuaj_ciao()  # Output: Thirrja e thuaj_ciao → Ciao!
```

12. Dokumentimi

Dokumentoni funksionet me stringje me tre thonjëza. Mund të aksesohet me help() ose __ doc __.

```python
def mbledh(a, b):
    """Kthen shumën e a dhe b."""
    return a + b

print(help(mbledh))  # Shfaq dokumentimin
```

13. Sugjerime për Tipet (Type Hints)

Anotacione opsionale për kontroll të datatype.

```python
def shumezo(a: int, b: int) -> int:
    return a * b
```

14. Closures dhe Nested Functions

Funksionet "Nested" mund të "mbajnë mend" variabla nga fushëveprimi i jashtëm.

```
def jashtme():
    mesazhi = "Përshëndetje"
    def brendshme():
        print(mesazhi)  # Merr mesazhin nga jashtme()
    return brendshme

closure = jashtme()
closure()  # Rezultati: Përshëndetje
```

15. Argumente të Ndryshueshme vs. Jo-ndryshueshme

Objektet e ndryshueshme (p.sh., lista) mund të modifikohen brenda funksionit.

```
def shto_element(lst, elementi):
    lst.append(elementi)

lista_ime = [1, 2]
shto_element(lista_ime, 3)
print(lista_ime)  # Rezultati: [1, 2, 3]
```